Innovative UNET-Based Steel Defect Detection Using 5 Pretrained Models

Yulianto, Rudy
Industrial Technology Faculty, Agricultural Institute of Bogor

Faqihudin
Industrial Technology Faculty, Agricultural Institute of Bogor

Meika Syahbana Rusli
Industrial Technology Faculty, Agricultural Institute of Bogor

Adhitio Satyo Bayangkari Karno
Department of Information System, Faculty of Engineering, Gunadarma University

他

KYUSHU UNIVERSITY

Innovative UNET-Based Steel Defect Detection Using 5 Pretrained Models

Rudy Yulianto[1], Faqihudin[2], Meika Syahbana Rusli[3],
Adhitio Satyo Bayangkari Karno[4], Widi Hastomo[5,*], Aqwam Rosadi Kardian[6],
Vany Terisia[7], and Tri Surawan[8]

[1,2,3]Industrial Technology Faculty, Agricultural Institute of Bogor, Indonesia

[2]Department of Information System, Faculty of Engineering, Gunadarma University, Depok, Indonesia

[5,7]Department of Information Technology, Ahmad Dahlan Institute of Technology and Business, Jakarta, Indonesia

[6]Sekolah Tinggi Manajemen Informatika dan Komputer Jakarta STI&K, Jakarta, Indonesia

[8]Department of Mechanical Engineering, Faculty of Technology Industry, Jayabaya University, Jakarta, Indonesia

*Author to whom correspondence should be addressed:
E-mail: Widie.has@gmail.com

Abstract: Checking the surface of steel is still primarily done manually and visually in the industrial business. Because of the high number and surface area of steel, inspection is time-intensive and of poor quality. Machine learning may be used to detect steel surfaces automatically by training a model with a large amount of data, and the model findings can then be used to detect other steel surfaces. Automatic detection will undoubtedly save control time, save prices, and improve checking quality. This research uses the core architecture of UNet and five versions of EfficientNet (B0 to B4) as the backbone (encoder). In this research, we use three types of training processes: the first is binary classification to predict the presence of defects; the second is multi-label classification to predict the type of defect; and the third is image segmentation to determine the location of defects according to their type. The results of this research show that the EfficientNet-B0 version can generally provide the best measurement results. Except for measuring recall in the binary classification process, it turns out that EfficientNet-B4 is the best, and measuring dice loss in the segmentation process for defect-4 turns out that EfficientNet-B2 gives the best results. Based on the accuracy value obtained from our study, it is still feasible to attempt employing a different type of architecture as a backbone (other than EfficientNet) to acquire even greater accuracy values in order to detect uncommon faults on the steel surface.

1. Introduction

Steel is a composite material that comprises numerous elements[1], as opposed to iron, which only contains one element[2]. Some contain a mix of steel-related components such as iron, carbon, nickel, silicon, manganese, aluminium, and others[3]. Because the amount of this substance varies, the qualities of steel vary, and steel serves several functions in human life[4]. Steel is frequently utilised in building construction, home fences, and other applications due to its high strength and inexpensive manufacturing costs[5]. Steel is lighter than regular iron due to its carbon component[6]. Because of its aluminium content, steel is less prone to corrosion and more resistant to acids, and it is frequently utilised in the automobile industry[7] (car engines and bodies, ships, and irplanes). Steel has a strong resistance to pressure[8], yet it is stiff (non-flexible) and difficult to shape[9]. Steel fractures and cracks due to this rigidity feature[10]. It is vital to know the status of the steel surface against defects in order to prevent the cracking process[11]. Once defects have been identified, additional treatment can be carried out to prevent the chance of more significant cracks[12]. Because of the enormous surface area of steel that must be detected (the hull or body of the ship), visual and manual detection will take a long time, effort, and cost, and have a low quality of observation[13]. The current computer vision technology is capable of identifying defect types and

locating defect locations in steel surface images[14].

The procedure for categorising and locating defects was initially done manually[15] by observing data in the form of pictures from the steel surface. Aside from the fact that this procedure cannot be applied to new types of problems, it also necessitates the use of skilled professionals. As machine learning technology has advanced, numerous neural network topologies with deeper layers, variations, and combinations have emerged as a continual endeavour to enhance accuracy[16]. The present deep learning technology has replaced human work that was previously performed manually with machine duties[17]. The combination of visual computing and the convolutional approach results in a new technology, CNN (Convolutional Neural Network), which can analyse images into small pieces to make obtaining the desired features easier[18].

Deep learning technology can not only do classification jobs but also discover (segmentation) defects on a steel surface by evaluating picture feature sections[19]. Furthermore, deep learning can detect other types of problems (new defects) that were not present throughout the machine learning training process[20]. The use of models that have been trained with huge amounts of data (transfer learning) speeds up the training process using other types of architectures [21] and results in improved classification accuracy[22]. Literature[23] has performed location detection of fine steel surface defects using the UNet architecture, although this method requires a large number of pixel annotations, which are time-consuming and expensive

In this research, we attempt to find the optimum model for categorising and estimating the location of defects using photographs of the steel surface as input data. The major architecture utilised for segmenting defects from the steel surface is UNet, which has shown good results in image segmentation[24]. We performed observations utilizing five variants of the EfficientNet architecture as the backbone and transferred learning data from ImageNet to select the best model.

2.   Related Work

Initially, a visual machine algorithm was used to detect steel surfaces[25]. Starting with an image processing technique that accurately detects defects, a two-stage target detection algorithm widely used in face detection and defect detection[26]. Segmented steel surface defects using thresholds[27]. Using fuzzy theory to discover defects by changing the image's grey level[28], with an accuracy of 97% Surface defect feature extraction has typically been reliant on experience, which is subjective and inefficient. Deep learning makes it feasible to overcome these challenges by automatically detecting visual features[29]. Several studies have been undertaken on the identification of steel surfaces using CNN and deep learning approaches. Target identification of minor imperfections on the steel surface with an average accuracy of 75% utilising the

enhanced Faster RCNN to rebuild the network structure[30]. Increasing productivity and cutting production costs by using deep learning algorithms to recognise and categorise steel sheets in smart factories (with 96% accuracy, 95% recall, and 97% precision[31]).

Classification for strip steel flaws using Mask-GAN and upgraded EfficientNet may overcome the problem of data scarcity in deep learning and correctly and effectively classify faults on the steel surface[32]. The TLU-NET framework, which was employed for steel surface identification, was able to enhance convergence and performance by 26% as training data dropped[33]. With a classification accuracy score of 100% and a segmentation F1-Score of 92%, the UNet and ResUNet architectures are employed to identify and discriminate the non-uniform product defect textures in the cartridge case[34]. Surface defects on military cartridges may be detected with 97% accuracy using DenseNet169 with data transmission and segmentation[35].

Wang et al. proposed a steel surface defect detection algorithm based on the improved YOLOv5 model, which enhances the model's feature extraction capability through one-shot aggregation[36]. A previous study proposed using a transfer learning-based U-Net framework for steel surface defect detection, comparing the performance of ResNet and DenseNet encoders with random initialization and pre-trained networks[37]. The research that has been carried out proposes a real-time steel surface defect detection technology based on the YOLO-v5 detection network, not UNET-based[38]. A previous study proposed a multi-task model for steel defect segmentation and severity estimation, but it did not mention the use of pretrained models[39].

3.   Methods

The initial part of this work is binary classification, developing a classification model to predict the presence or absence of defects on steel surfaces[40]. The results of the initial stage for data that has defects are used as data for the second stage. The second stage is multi-label classification, developing a classification model for four types of steel surface defect classes[37] (classes 1, 2, 3, and 4). The next stage is to carry out a training process to estimate the location of defects (segmentation)[41] that are within the threshold value range.

For the classification training process (binary and multi-label) using EfficientNet[42], and to obtain good results and speed up the training process, transfer learning (imagenet) is used. For the segmentation training process, the UNet architecture is used with EfficientNet as the backbone[43]. To compare the findings, five versions of the EfficientNet architecture[44] (versions B0 to B4) were used. Before being used as input data, raw data must first go through several pre-processing steps to ensure it is ready and conforms to the input data format required for subsequent model training[45]. Many parameters must be determined to obtain accurate results[46]. To evaluate the

success of this method, the accuracy and losses of the training process must be measured[47]. In general, the stages of this research are illustrated in Figure 1.

3.1. Dataset

The dataset includes one csv file and two folders, each having steel surface pictures and photos displaying the shade (mask) of the fault site. The defect data in the csv file consists of 7,095 rows with three columns (ImageId, ClassId, and EncodedPixels) (Figure 2). ImageId is to show the name of the image identity, and the annotation is EncodedPixels, which is a set of pixel masks of the steel surface, including the faults. Each row indicates a single form of flaw in a picture. Figure 3 depicts the number of photos for each type of defect, with type 3 having the most (5,150) and type 2 having the fewest (247). The image name appears many times in the ImageId column, indicating that the picture has more than one type of defect. Figure 4 indicates that there are 6,239 photos with a single kind of flaw, 425 images with two types of faults, and only two images with three types of defects.
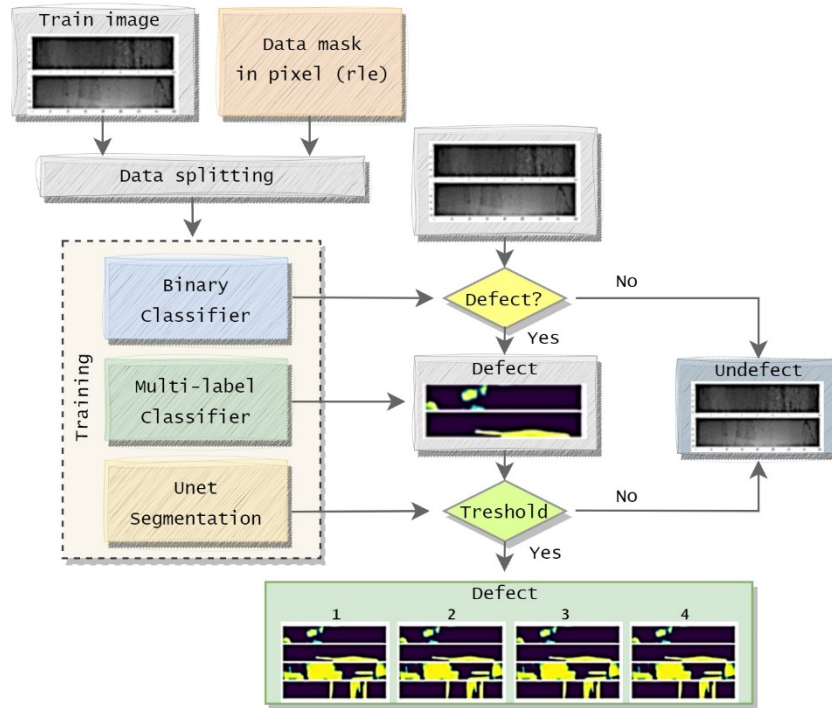

Fig. 1: Flowchart of the Research Model

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
...	...	...	...
7092	fffe98443.jpg	3	105929 5 106177 14 106424 24 106672 33 106923 ...
7093	ffff4eaa8.jpg	3	16899 7 17155 20 17411 34 17667 47 17923 60 18...
7094	ffffd67df.jpg	3	30931 43 31103 127 31275 211 31489 253 31745 2...

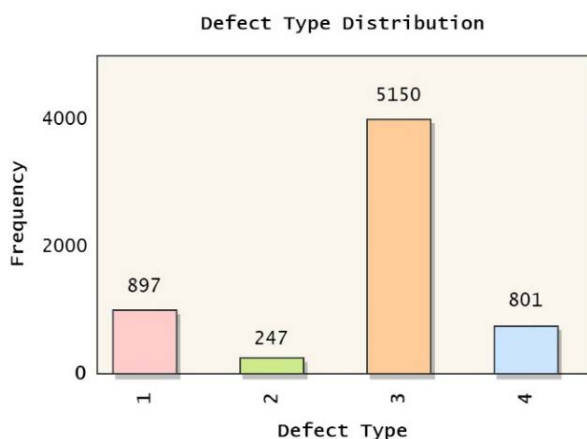7095 rows × 3 columns

Fig. 2: File datasets
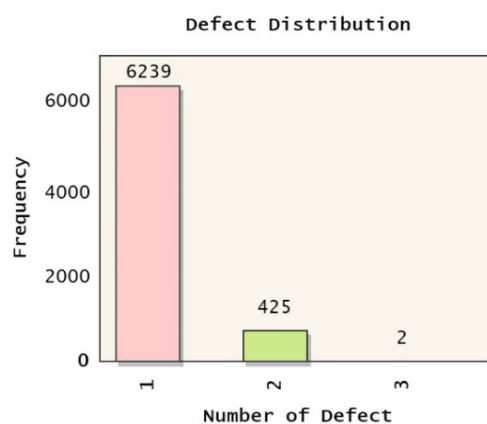
Fig. 3: Defect type distribution

Fig. 4: Defect distribution

3.2. Pre-Processing Data

After comprehending the raw dataset, the next step is to phase. A data rotation (pivoting) technique is required to turn the raw data into data that may be known for the number of flaws in each ImageId. The output of the data pivoting procedure is shown in Figure 5, where the ImageId column includes data with unique values.

prepare the data for use in the subsequent modelling Columns 'Defect_1', 'Defect_2', 'Defect_3', and 'Defect_e4' reflect annotation features for classes 1, 2, 3, and 4, respectively. After pivoting the data into 12,568 rows (Figure 5).

Fig. 5: Result of pivoting data

3.3. UNet

In general, the UNet architecture is divided into two parts: the encoder and the decoder in the left and right locations, respectively (Figure 6). The picture is initially broken into smaller pieces (pixels) with similar features. The encoder receives input data in the form of a picture, and in order to facilitate the training process and get input images of varying dimensions, the encoder reduces the dimensions of the image in phases. More comprehensive informational characteristics will be sought at each level of dimension reduction. While the decoder section is useful for gathering all of the information collected by the encoder in the form of segmentation output.

In this work, we utilise four convolutional blocks, each of which increases the number of features by twofold over the previous step. Convolution, relu, and max-pooling routines are used at the start of the encoder process to increase the number of features to 64. A skip connection is a connecting path to the decoder that exists at each stage of the encoder process. The encoder process follows the same procedures as the encoder stage, but without the max-pooling phase. In contrast to the encoder, which performs down sampling, each stage in the decoder will do up sampling such that the number of features equals the number of features in the input data picture. The activation function used in the decoder is sigmoid to obtain segmented image results.
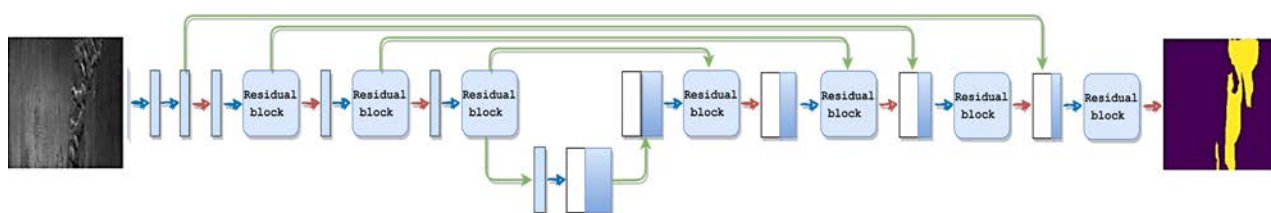

Fig. 6: UNet Architecture

3.4. EfficientNet architecture

EfficientNet is a neural network model in the deep learning category because of its thick layer. Many deep learning architectures are broadening[48] or multiplying the layers in an effort to increase accuracy, making the training process heavier and requiring more processing time. In contrast to other architectures, EfficientNet increases its capabilities effectively and efficiently [49] by using a balanced architecture basis and transfer learning dataset when adding layers. This gives EfficientNet a more dependable network resolution. Transfer learning can be used to deal with small amounts of input and speed up the model's convergence during training. EfficientNet versions b0 through b4 are used in this study as the foundation of the primary architecture, which is UNet. To determine which version has the highest accuracy value, these five versions are compared.

The EfficientNet design is a MobConv layer stack (mobile convolution blocks). Each convolution sub-block in the MobConv block has a convolution network, batch normalisation, and Swish activation function (Figure 7). The Swish activation function is used because it has several advantages that allow neural networks to learn more complex relationships between input and output data. This helps the neural network understand and extract more abstract features from the image. Swish has a better ability to overcome the vanishing gradient problem, which often arises in deep networks. Computational efficiency makes Swish very suitable for devices that have limited resources.

The picture input will be augmented using Conv (1x1) for the first time to obtain a large amount of feature information. Finally, to boost performance even more, choose Conv (3x3) or 5x5. A normalising batch is added to each subblock to compress the features and execute feature excitation procedures on each channel. The SWISH activation function is used to calculate the weight difference.

Figure 8 depicts the architecture version B0, with the image input performing a Conv (3x3) operation for the first time, followed by 16 MobConv modules. Conv (1x1), pooling, FC (completely connected), and softmax activation function make up the last layer. Table 1 lists architectures with versions B1, B2, B3, and B4. Table 1 lists the number of channels in Sandler's[50] EfficientNet architecture, which consists of many convolutional mobile subblocks (MobCon).
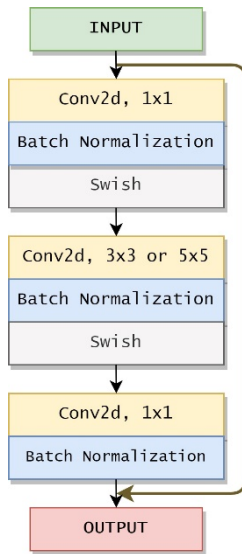
Fig. 7: Mobile Convolution Blocks (MobConv)
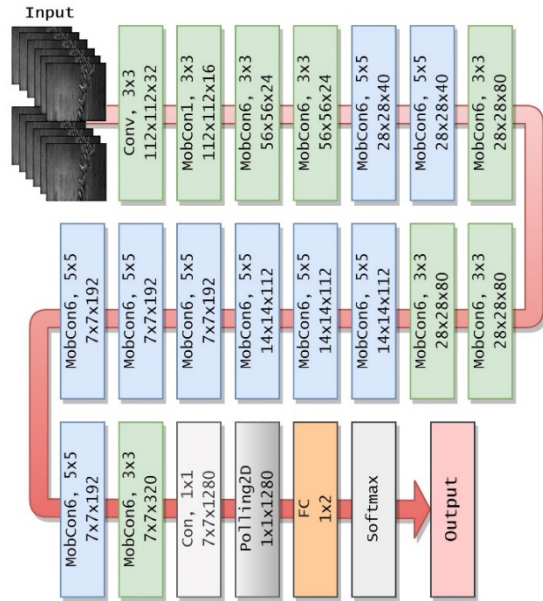
Fig. 8: EfficientNetB0 architecture

Table 1. Series of channels from 5 variants of EfficientNet

Subblock	B0	B1	B2	B3	B4
Con3x3	32	32	32	40	48
MobCon1, k3x3	16	16	16	24	24
MobCon1, k3x3	24	24	24	32	32
MobCon1, k3x3	40	40	48	48	56
MobCon1, k3x3	80	80	88	96	112
MobCon1, k3x3	112	112	120	136	160
MobCon1, k3x3	192	192	208	232	272
MobCon1, k3x3	320	320	352	384	448
Conv1x1+Pool+FC	1280	1280	1408	1536	1792

3.5. Threshold

The regional profiles on the defect mask containing the steel surface appear to be indistinguishable between the different classes. Although the type 1 defect appears to have many small-sized regions, the image of the type 4 defect has many medium-sized regions. Type 3 defects appear to also contain several medium-sized regions.

Meanwhile, images of type 2 and type 3 defects appear to have some of the same regional characteristics (Figure 9). There is a lot of overlap in regional ranges. The minimum areas for each type of defect can be seen closer to each other. Meanwhile, the maximums are mostly different. We can use the minimum and maximum area values in the training images to test the image defect prediction threshold.
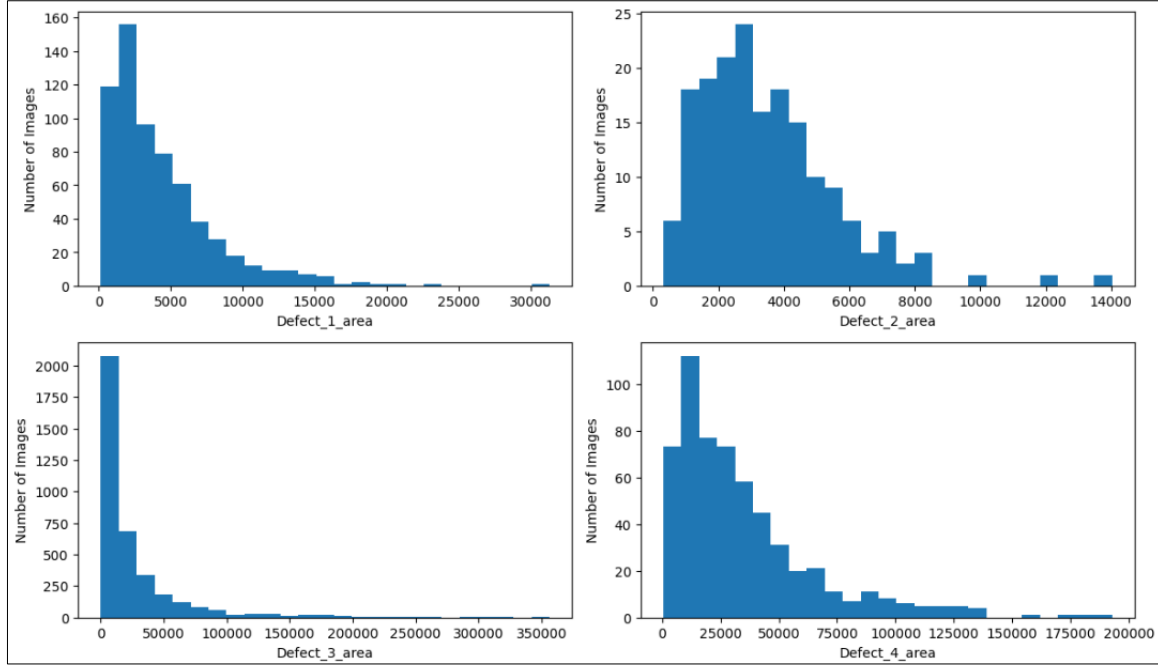
Fig. 9: Distribution of the number of images against the area of defects for types Defect_1, Defect_2, Defect_3, and Defect_4.

Based on the area range for each defect, we will determine a prediction threshold to filter outliers. For example, some of the predicted masks only have 4 pixels that have a value of 1. Such images will reduce the model's performance in the final metric. To determine the threshold value, we will use data between the 2nd percentile and the 98th percentile, namely: for defect_1, it is between 500 and 15500; for defect_2, it is between 700 and 10000; for defect_3, it is between 1100 and 160000; and for defect_4, it is between 2800 and 127000.

3.6. Evaluation Metrics

There are several measuring tools that we use to evaluate classification results (binary and multi-label classification):

- Binary Cross-Entropy (BCE): this metric measures the extent to which the probabilities given by the model correspond to the actual class. For each sample in the dataset, the model produces a probability expressed as p(y|x), where y is the actual class and x is the input feature. Binary cross-entropy measures how close the probability generated by the model is to the actual class (0 for the negative class and 1 for the positive class). The formula is:

$$BCE = -\sum(y * log(p) + (1 - y) * log(1 - p)) \qquad (1)$$

where y is the actual label (0 or 1), and p is the probability generated by the model. A lower binary cross-entropy value indicates that the model is better at modelling the true class probability distribution.

- Accuracy: the accuracy metric measures the proportion of correct predictions from the model in the total predictions. The way to calculate accuracy is as follows:

$$Accuracy = (Number\ of\ Correct\ Predictions) / (Total\ Predictions) \qquad (2)$$

In the context of classification, "Number of Correct Predictions" refers to the number of samples correctly predicted in terms of class labels, while "Total Predictions" is the total number of samples evaluated.

- Precision measures the proportion of correct positive predictions (true positives) from the total positive predictions that have been made. This metric provides information about the extent to which your model can avoid classifying negatives as positives. The precision formula is as follows:

$$Precision = (True\ Positives) / (True\ Positives + False\ Positives) \qquad (3)$$

Here, "true positives" is the number of positive cases correctly predicted by the model, and "false positives" is the number of negative cases incorrectly predicted as positive by the model.

- Recall measures the proportion of positive classes that are predicted correctly (true positives) from the total positive classes that actually exist. This metric provides an idea of the extent to which your model is able to identify all true positive cases. The formula for calculating recall is as follows:

$$Recall = (True\ Positives) / (True\ Positives + False\ Negatives) \qquad (4)$$

- F1_score is the harmonic mean of precision and recall and measures how well your model can achieve a balance between precision and recall. F1-Score is calculated using the following formula:

$$F1 - Score = 2 * (Presisi * Recall) / (Presisi + Recall) \qquad (5)$$

For segmentation measuring tools, use:

- Dice loss is the loss function used during segmentation model training. The goal is to minimise the difference between model predictions and masks (ground truth) that correspond to segmentation accuracy. Dice loss is calculated using the following formula:
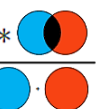
$$Dice\ Loss = 1 - (2 * Intersection) / (Union + Smoothing) \qquad (6)$$

where "intersection" is the area where the model predictions and the ground truth mask meet. "Union" is the total area of both model predictions and the ground truth mask. "Smoothing" is a small value added to the denominator to avoid dividing by zero.

- Dice_coefficient is an evaluation metric used to measure the extent to which the segmentation model matches the ground truth mask. This is a measurement of segmentation accuracy. The dice coefficient is calculated using the following formula:

$$Dice\ Coefficient = (2 * Intersection) / (Union + Smoothing) \qquad (7)$$

where "intersection" and "union" are the same as described above. "Smoothing" is a small value added to avoid dividing by zero. The dice coefficient is used to calculate the degree of similarity[51] by comparing the pixels in the two pictures (predicted and ground truth) (Figure 10). The dice coefficient value ranges between 0 and 1. The closer the dice coefficient value is to 1 (one), the better it is for assessing loss (zero).

$$\frac{2*|X \cap Y|}{|X|+|Y|} =$$

Fig. 10: Illustration Dice Coefficient

4. Results and discussion

The resulting dataset after the merger and pivot processes is 12,568 rows. Before the training process is carried out, the dataset will be split into test_set 10% (1,257 rows) and train_valid 90% (11,311 rows), then the train_valid data will be split again into train set 80% (9,048 rows) and validation set 20% (2,263 rows). Due to the limited number of pages, the evaluation results in graphical form are only displayed for one model, namely EfficientNet-B0. This graph is grouped into 3 evaluation results, namely figure 11 for a collection of graphs of evaluation results from the binary classification process using 5 measuring tools (binary crossentropy, accuracy, f1_score, precision, and recall). Figure 12 is a collection of graphs of evaluation results from multi-label classification using the same five measuring tools as binary classification. Meanwhile, Figure 13 is a collection of graphs resulting from the evaluation of the segmentation process using two measuring tools (dice loss and dice coefficient).

The evaluation results in numerical form will be displayed in each table in full, starting from table 2 to table 6 for the respective training processes using EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, EfficientNet-B3, and EfficientNet-B4. Each table contains measurement values using five measuring tools for binary classification and multi-label classification and two measuring tools for segmentation evaluation. Measurements were carried out using three types of data: train data, validation data, and test data. Because the large number of observations made show many evaluation results, the best measurement value is the smallest evaluation value for BCE and dice loss and the largest measurement value for other measurements. The results of selecting the best value are collected in Table 7, which looks simpler and easier to conclude.

Each measuring instrument has advantages and disadvantages; the use of more than one measuring instrument is highly recommended so that a comparative analysis can be carried out to obtain the best results. Table 7 is the best evaluation result of all the observations made in this research. It can be seen that for all measurements, the Eff-B0 architecture can provide the best results for all processes (binary classification, multi-label classification, and segmentation). There are only two measurements that show differences, namely Eff-B4 for measuring recall in the binary classification process and Eff-B2 for measuring dice loss in the defect-4 segmentation process.
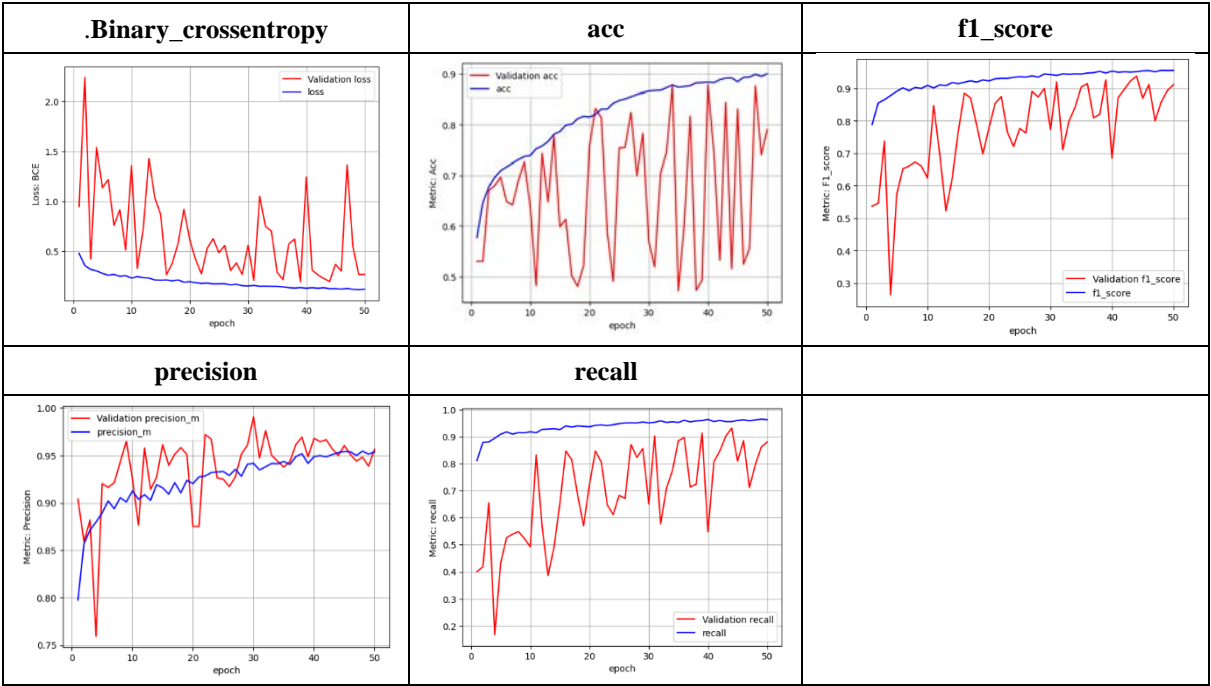
Fig. 11: Graph of binary classification evaluation results for the EfficientNet-B0 model
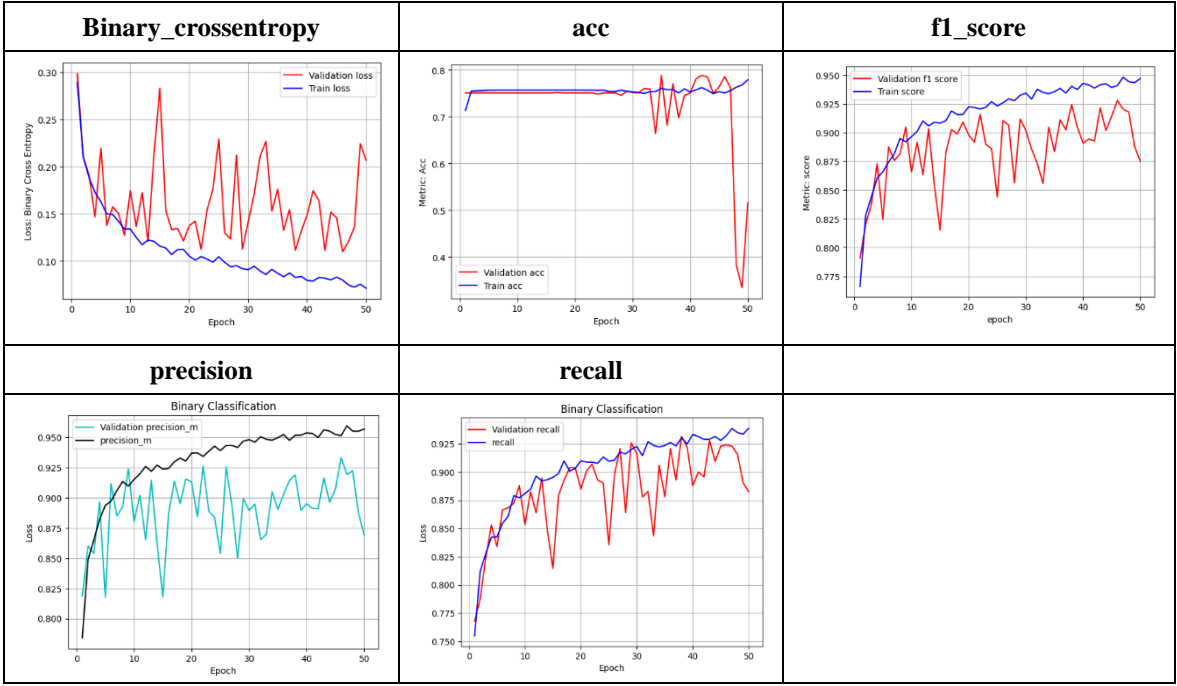
Fig. 12: Graph of multi-label classification evaluation results for the EfficientNet-B0 model
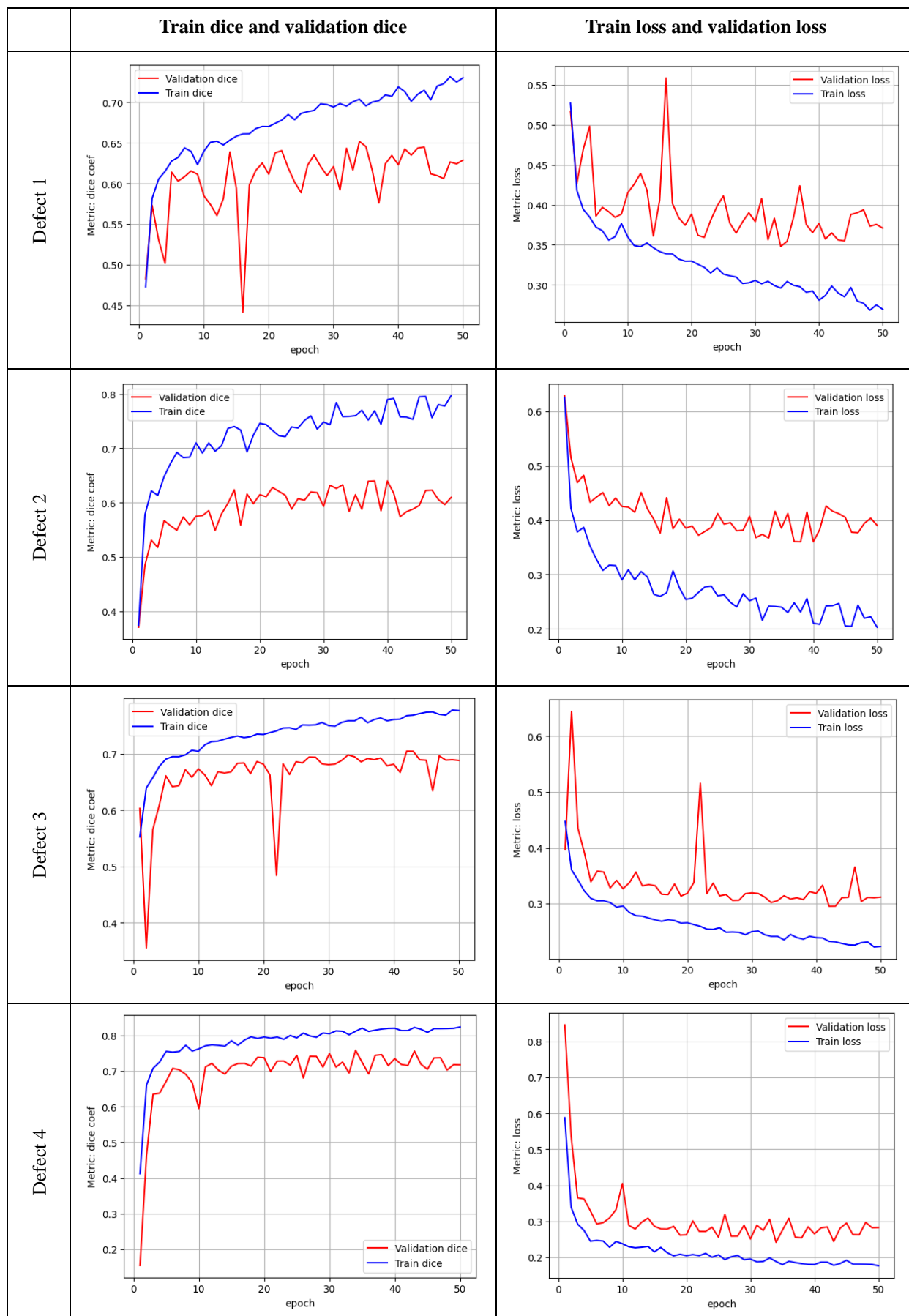
Train dice and validation dice	Train loss and validation loss
Defect 1	
Defect 2	
Defect 3	
Defect 4	

Fig. 13: Graph of image segmentation evaluation results for the EfficientNet-B0 model

Table 2. Evaluation results for the EfficientNet-B0 model

	Metric		Evaluation score		
			Train	Validation	Test
Binary classification	binary_crossentropy		**0.178447**	**0.265499**	**0.250437**
	acc		**0.934240**	**0.912506**	**0.917263**
	f1_score		**0.933116**	**0.909538**	**0.912092**
	precision		**0.970758**	**0.953391**	**0.965039**
	recall		0.905879	0.880826	0.875233
Multi-label classification	binary_crossentropy		**0.155322**	**0.206799**	**0.192802**
	acc		**0.878725**	**0.860000**	**0.868066**
	f1_score		**0.903832**	**0.874959**	**0.895421**
	precision		**0.906790**	**0.868914**	**0.906721**
	recall		**0.902535**	**0.883114**	**0.885870**
Image segmentation	Defect-1	dice_lost	**0.241853**	**0.370989**	**0.397452**
		dice_coef	**0.758147**	**0.629011**	**0.602548**
	Defect-2	dice_lost	**0.178259**	**0.390348**	**0.419159**
		dice_coef	**0.821741**	**0.609652**	**0.580841**
	Defect-3	dice_lost	**0.263068**	**0.311629**	**0.328843**
		dice_coef	**0.736931**	**0.688371**	**0.671156**
	Defect-4	dice_lost	0.214004	0.282338	0.254890
		dice_coef	**0.807605**	**0.717663**	**0.745110**

Table 3. Evaluation results for the EfficientNet-B1 model

	Metric		Evaluation score		
			Train	Validation	Test
Binary classification	binary_crossentropy		0.788510	0.907279	0.801463
	acc		0.785477	0.763146	0.785203
	f1_score		0.756128	0.735602	0.742382
	precision		0.887375	0.876266	0.894966
	recall		0.679889	0.651853	0.653129
Multi-label classification	binary_crossentropy		0.240653	0.264186	0.252189
	acc		0.834341	0.820833	0.826087
	f1_score		0.840779	0.824454	0.837668
	precision		0.837435	0.817425	0.843749
	recall		0.846818	0.833748	0.834382
Image segmentation	Defect-1	dice_lost	0.373972	0.396631	0.402671
		dice_coef	0.626028	0.603369	0.597329
	Defect-2	dice_lost	0.360009	0.539270	0.453084
		dice_coef	0.639991	0.460730	0.546916
	Defect-3	dice_lost	0.308332	0.331337	0.338015
		dice_coef	0.691668	0.668663	0.661985
	Defect-4	dice_lost	0.273183	0.295402	0.317423
		dice_coef	0.726816	0.704598	0.682578

Table 4. Evaluation results for the EfficientNet-B2 model

	Metric	Evaluation score		
		Train	Validation	Test
Binary classification	binary_crossentropy	0.541636	0.565283	0.539540
	acc	0.806919	0.790102	0.796340
	f1_score	0.831168	0.815828	0.821966
	precision	0.755907	0.738654	0.403072
	recall	0.943803	0.932571	0.936905
Multi-label classific	binary_crossentropy	0.305657	0.304111	0.295493
	acc	0.764951	0.761667	0.755622
	f1_score	0.768916	0.768373	0.769678

			0.790763	0.786341	0.797085
	precision		0.790763	0.786341	0.797085
	recall		0.750545	0.753532	0.746478
Image segmentation	Defect-1	dice_lost	0.373497	0.4147	0.419832
		dice_coef	0.626502	0.5853	0.580168
	Defect-2	dice_lost	0.285572	0.370037	0.401043
		dice_coef	0.714428	0.629963	0.598957
	Defect-3	dice_lost	0.335688	0.352584	0.369652
		dice_coef	0.664313	0.647416	0.630348
	Defect-4	dice_lost	**0.192395**	**0.26306**	**0.237202**
		dice_coef	0.785996	0.73694	0.762798

Table 5. Evaluation results for the EfficientNet-B3 model

	Metric		Evaluation score		
			Train	Validation	Test
Binary classification	binary_crossentropy		0.332500	0.348610	0.351250
	acc		0.840628	0.836058	0.824185
	f1_score		0.820040	0.818294	0.789258
	precision		0.947515	0.935388	0.932504
	recall		0.736186	0.742428	0.698171
Multi-label classification	binary_crossentropy		0.251697	0.274614	0.251900
	acc		0.813294	0.810000	0.802099
	f1_score		0.845255	0.832750	0.840432
	precision		0.845516	0.825251	0.847375
	recall		0.847249	0.841944	0.835377
Image segmentation	Defect-1	dice_lost	0.368514	0.415835	0.428514
		dice_coef	0.631486	0.584166	0.571486
	Defect-2	dice_lost	0.269074	0.404454	0.745908
		dice_coef	0.730926	0.595546	0.596928
	Defect-3	dice_lost	0.340644	0.357973	0.380009
		dice_coef	0.659355	0.642027	0.619991
	Defect-4	dice_lost	0.283830	0.283830	0.246677
		dice_coef	0.716170	0.716170	0.753323

Table 6. Evaluation results for the EfficientNet-B4 model

	Metric		Evaluation score		
			Train	Validation	Test
Binary classification	binary_crossentropy		0.346099	0.363001	0.332109
	acc		0.847922	0.839152	0.841687
	f1_score		0.863654	0.856155	0.854614
	precision		0.796479	0.784471	0.799043
	recall		**0.959294**	**0.958249**	**0.936440**
72Multi-label classification	binary_crossentropy		0.275177	0.274807	0.256699
	acc		0.714107	0.710833	0.743628
	f1_score		0.730746	0.725928	0.753428
	precision		0.754317	0.746032	0.773512
	recall		0.710991	0.709270	0.735849
Image segmentation	Defect-1	dice_lost	0.378674	0.408794	0.448929
		dice_coef	0.621326	0.591206	0.551071
	Defect-2	dice_lost	0.347912	0.413195	0.391237
		dice_coef	0.652088	0.586805	0.608763
	Defect-3	dice_lost	0.360699	0.380269	0.394802
		dice_coef	0.639301	0.619731	0.605198
	Defect-4	dice_lost	0.235969	0.272217	0.240393
		dice_coef	0.764031	0.727783	0.759607

Table 7. Evaluation results with the best measurement values

	Metric		Evaluation score			Eff
			Train	Validation	Test	
Binary classificati-on	binary_crossentropy		0.178447	0.265499	0.250437	B0
	acc		0.934240	0.912506	0.917263	B0
	f1_score		0.933116	0.909538	0.912092	B0
	precision		0.970758	0.953391	0.965039	B0
	recall		0.959294	0.958249	0.936440	**B4**
Multi-label classification	binary_crossentropy		0.155322	0.206799	0.192802	B0
	acc		0.878725	0.860000	0.868066	B0
	f1_score		0.903832	0.874959	0.895421	B0
	precision		0.906790	0.868914	0.906721	B0
	recall		0.902535	0.883114	0.885870	B0
Image segmentation	Defect-1	dice_lost	0.241853	0.370989	0.397452	B0
		dice_coef	0.758147	0.629011	0.602548	B0
	Defect-2	dice_lost	0.178259	0.390348	0.419159	B0
		dice_coef	0.821741	0.609652	0.580841	B0
	Defect-3	dice_lost	0.263068	0.311629	0.328843	B0
		dice_coef	0.736931	0.688371	0.671156	B0
	Defect-4	dice_lost	0.192395	0.26306	0.237202	**B2**
		dice_coef	0.807605	0.717663	0.745110	B0

5. Conclusion

In this study, we use 5 versions of the EfficientNet architecture (B0, B1, B2, B3, and B4) in the binary classification and multi-label classification processes. And for the segmentation process, using UNet as the main architecture, and for the backbone (encoder), using 5 versions of EfficientNet The results of this research show that the EfficientNet-B0 version can generally provide the best measurement results. Except for measuring recall in the binary classification process, EfficientNet-B4 is the best, and measuring dice loss in the segmentation process for defect-4 turns out that EfficientNet-B2 gives the best results. For future studies, we would like to test another form of encoder (other than EfficientNet) to obtain even better accuracy values in order to generalise uncommon flaws.

References

1) A. Kumar Sharma, R. Bhandari, A. Aherwar, R. Rimašauskienė, and C. Pinca-Bretotean, "A study of advancement in application opportunities of aluminum metal matrix composites," *Materials Today: Proceedings*, **26** 2419–2424 (2020). doi:10.1016/j.matpr.2020.02.516.

2) A.-C.S. Vogt, T. Arsiwala, M. Mohsen, M. Vogel, V. Manolova, and M.F. Bachmann, "On iron metabolism and its regulation," *IJMS*, **22** (*9*) 4591 (2021). doi:10.3390/ijms22094591.

3) A. Gramlich, C. Van Der Linde, M. Ackermann, and W. Bleck, "Effect of molybdenum, aluminium and boron on the phase transformation in 4 wt.–% manganese steels," *Results in Materials*, **8** 100147 (2020). doi:10.1016/j.rinma.2020.100147.

4) O. Gencel, O. Karadag, O.H. Oren, and T. Bilir, "Steel slag and its applications in cement and concrete technology: a review," *Construction and Building Materials*, **283** 122783 (2021). doi:10.1016/j.conbuildmat.2021.122783.

5) Z. Fan, and S.J. Friedmann, "Low-carbon production of iron and steel: technology options, economic assessment, and policy," *Joule*, **5** (*4*) 829–862 (2021). doi:10.1016/j.joule.2021.02.018.

6) L.C. Yule, V. Shkirskiy, J. Aarons, G. West, B.A. Shollock, C.L. Bentley, and P.R. Unwin, "Nanoscale electrochemical visualization of grain-dependent anodic iron dissolution from low carbon steel," *Electrochimica Acta*, **332** 135267 (2020). doi:10.1016/j.electacta.2019.135267.

7) M. Ghosh, A. Ghosh, and A. Roy, "Renewable and Sustainable Materials in Automotive Industry," in: Encyclopedia of Renewable and Sustainable Materials, Elsevier, 2020: pp. 162–179. doi:10.1016/B978-0-12-803581-8.11461-4.

8) X. Zhang, Q.J. Wang, K.L. Harrison, S.A. Roberts, and S.J. Harris, "Pressure-driven interface evolution in solid-state lithium metal batteries," *Cell Reports Physical Science*, **1** (*2*) 100012 (2020). doi:10.1016/j.xcrp.2019.100012.

9) D.B. Sitotaw, D. Ahrendt, Y. Kyosev, and A.K. Kabish, "Additive manufacturing and textiles—state-of-the-art," *Applied Sciences*, **10** (*15*) 5033 (2020). doi:10.3390/app10155033.

10) P. Aghdasi, and C.P. Ostertag, "Tensile fracture

characteristics of green ultra-high performance fiber-reinforced concrete (g-uhp-frc) with longitudinal steel reinforcement," *Cement and Concrete Composites*, **114** 103749 (2020). doi:10.1016/j.cemconcomp.2020.103749.

11) R. Hao, B. Lu, Y. Cheng, X. Li, and B. Huang, "A steel surface defect inspection approach towards smart industrial monitoring," *J Intell Manuf*, **32** (*7*) 1833–1843 (2021). doi:10.1007/s10845-020-01670-2.

12) J. Tomków, D. Fydrych, and G. Rogalski, "Dissimilar underwater wet welding of hsla steels," *Int J Adv Manuf Technol*, **109** (*3–4*) 717–725 (2020). doi:10.1007/s00170-020-05617-y.

13) Q. Luo, X. Fang, L. Liu, C. Yang, and Y. Sun, "Automated visual defect detection for flat steel surface: a survey," *IEEE Trans. Instrum. Meas.*, **69** (*3*) 626–644 (2020). doi:10.1109/TIM.2019.2963555.

14) B. Xue, and Z. Wu, "Key technologies of steel plate surface defect detection system based on artificial intelligence machine vision," *Wireless Communications and Mobile Computing*, **2021** 1–12 (2021). doi:10.1155/2021/5553470.

15) Y. Dong, D. Tao, X. Li, J. Ma, and J. Pu, "Texture classification and retrieval using shearlets and linear regression," *IEEE Transactions on Cybernetics*, **45** (*3*) 358–369 (2015). doi:10.1109/TCYB.2014.2326059.

16) Henry Pariaman, G M Luciana, M K Wisyaldin, and Muhammad Hisjam, "Anomaly detection using lstm-autoencoder to predict coal pulverizer condition on coal-fired power plant," *Evergreen*, **8**(*1*) 89–97 (2021). doi:10.5109/4372264.

17) Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, **521** (*7553*) 436–444 (2015). doi:10.1038/nature14539.

18) K. Islam, "Person search: new paradigm of person re-identification: a survey and outlook of recent works," *Image and Vision Computing*, **101** 103970 (2020). doi:10.1016/j.imavis.2020.103970.

19) Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Computer-Aided Civil and Infrastructure Engineering*, **33** (*9*) 731–747 (2018). doi:https://doi.org/10.1111/mice.12334.

20) L. Song, W. Lin, Y.-G. Yang, X. Zhu, Q. Guo, and J. Xi, "Weak micro-scratch detection based on deep convolutional neural network," *IEEE Access*, **7** 27547–27554 (2019). doi:10.1109/ACCESS.2019.2894863.

21) J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009: pp. 248–255. doi:10.1109/CVPR.2009.5206848.

22) R. Ren, T. Hung, and K.C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE Transactions on Cybernetics*, **48** (*3*) 929–940 (2018). doi:10.1109/TCYB.2017.2668395.

23) D. Amin, and S. Akhter, "Deep Learning-Based Defect Detection System in Steel Sheet Surfaces," in: 2020 IEEE Region 10 Symposium (TENSYMP), 2020: pp. 444–448. doi:10.1109/TENSYMP50017.2020.9230863.

24) O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in: N. Navab, J. Hornegger, W.M. Wells, A.F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015, Springer International Publishing, Cham, 2015: pp. 234–241.

25) B. Tang, L. Chen, W. Sun, and Z. Lin, "Review of surface defect detection of steel products based on machine vision," *IET Image Processing*, **17** (*2*) 303–322 (2023). doi:10.1049/ipr2.12647.

26) W. Zhao, F. Chen, H. Huang, D. Li, and W. Cheng, "A new steel defect detection algorithm based on deep learning," *Computational Intelligence and Neuroscience*, **2021** 1–13 (2021). doi:10.1155/2021/5592878.

27) N. Prappacher, M. Bullmann, G. Bohn, F. Deinzer, and A. Linke, "Defect detection on rolling element surface scans using neural image segmentation," *Applied Sciences*, **10** (*9*) 3290 (2020). doi:10.3390/app10093290.

28) J. Zhang, H. Wang, Y. Tian, and K. Liu, "An accurate fuzzy measure-based detection method for various types of defects on strip steel surfaces," *Computers in Industry*, **122** 103231 (2020). doi:10.1016/j.compind.2020.103231.

29) P. Panwar, P. Roshan, R. Singh, M. Rai, Asha Rani Mishra, and Sansar Singh Chauhan, "DDNet- a deep learning approach to detect driver distraction and drowsiness," *Evergreen*, **9**(*3*) 881–892 (2022). doi:10.5109/4843120.

30) W. Zhao, F. Chen, H. Huang, D. Li, and W. Cheng, "A new steel defect detection algorithm based on deep learning," *Computational Intelligence and Neuroscience*, **2021** 5592878 (2021). doi:10.1155/2021/5592878.

31) A.T. Prihatno, I.B.K.Y. Utama, J.Y. Kim, and Y.M. Jang, "Metal Defect Classification Using Deep Learning," in: 2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN), 2021: pp. 389–393. doi:10.1109/ICUFN49451.2021.9528702.

32) S. Guan, J. Chang, H. Shi, X. Xiao, Z. Li, X. Wang, and X. Wang, "Strip steel defect classification using the improved gan and efficientnet," *Applied Artificial Intelligence*, **35** (*15*) 1887–1904 (2021). doi:10.1080/08839514.2021.1995231.

33) P. Damacharla, A.R. M. V., J. Ringenberg, and A.Y.

Javaid, "TLU-Net: A Deep Learning Approach for Automatic Steel Surface Defect Detection," in: 2021 International Conference on Applied Artificial Intelligence (ICAPAI), 2021: pp. 1–6. doi:10.1109/ICAPAI49758.2021.9462060.

34) S. Tural, R. Samet, S. Aydin, and M. Traore, "Deep learning based classification of military cartridge cases and defect segmentation," *IEEE Access*, **10** 74961–74976 (2022). doi:10.1109/ACCESS.2022.3191328.

35) S. Tural, R. Samet, S. Aydin, and M. Traore, "Deep learning based classification of military cartridge cases and defect segmentation," *IEEE Access*, **10** 74961–74976 (2022). doi:10.1109/ACCESS.2022.3191328.

36) Y. Lu, and F. Qu, "Steel surface defect detection based on improved yolov5 algorithm," *J. Phys.: Conf. Ser.*, **2395** (*1*) 012063 (2022). doi:10.1088/1742-6596/2395/1/012063.

37) P. Damacharla, A.R.M. V., J. Ringenberg, and A.Y. Javaid, "TLU-net: a deep learning approach for automatic steel surface defect detection," (2021). http://arxiv.org/abs/2101.06915 (accessed October 29, 2023).

38) L. Wang, X. Liu, J. Ma, W. Su, and H. Li, "Real-time steel surface defect detection with improved multi-scale yolo-v5," *Processes*, **11** (*5*) 1357 (2023). doi:10.3390/pr11051357.

39) R. Neven, and T. Goedemé, "A multi-branch u-net for steel surface defect type and severity segmentation," *Metals*, **11** (*6*) 870 (2021). doi:10.3390/met11060870.

40) I. Konovalenko, P. Maruschak, J. Brezinová, J. Viňáš, and J. Brezina, "Steel surface defect classification using deep residual neural network," *Metals*, **10** (*6*) 846 (2020). doi:10.3390/met10060846.

41) D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," *J Intell Manuf*, **31** (*3*) 759–776 (2020). doi:10.1007/s10845-019-01476-x.

42) J. Wang, L. Yang, Z. Huo, W. He, and J. Luo, "Multi-label classification of fundus images with efficientnet," *IEEE Access*, **8** 212499–212508 (2020). doi:10.1109/ACCESS.2020.3040275.

43) N. Sharma, S. Gupta, D. Koundal, S. Alyami, H. Alshahrani, Y. Asiri, and A. Shaikh, "U-net model with transfer learning model as a backbone for segmentation of gastrointestinal tract," *Bioengineering*, **10** (*1*) 119 (2023). doi:10.3390/bioengineering10010119.

44) A.S. Bayangkari Karno, W. Hastomo, T. Surawan, S.R. Lamandasa, S. Usuli, H.R. Kapuy, and A. Digdoyo, "Classification of cervical spine fractures using 8 variants efficientnet with transfer learning," *IJECE*, **13** (*6*) 7065 (2023). doi:10.11591/ijece.v13i6.pp7065-7077.

45) Y.-S. Won, X. Hou, D. Jap, J. Breier, and S. Bhasin, "Back to the basics: seamless integration of side-channel pre-processing in deep neural networks," *IEEE Trans.Inform.Forensic Secur.*, **16** 3215–3227 (2021). doi:10.1109/TIFS.2021.3076928.

46) S.E.J. Bell, G. Charron, E. Cortés, J. Kneipp, M.L. De La Chapelle, J. Langer, M. Procházka, V. Tran, and S. Schlücker, "Towards reliable and quantitative surface-enhanced raman scattering (sers): from key parameters to good analytical practice," *Angew Chem Int Ed*, **59** (*14*) 5454–5462 (2020). doi:10.1002/anie.201908154.

47) O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, and D.J. Inman, "A review of vibration-based damage detection in civil structures: from traditional methods to machine learning and deep learning applications," *Mechanical Systems and Signal Processing*, **147** 107077 (2021). doi:10.1016/j.ymssp.2020.107077.

48) K. He, X. Zhang, & Ren, S., and J. Sun, "Deep residual learning for image recognition," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).

49) Q. Tan, M., & Le, "Efficientnet: rethinking model scaling for convolutional neural networks," *In International Conference on Machine Learning*, 6105–6114 (2019).

50) M. Sandler, A. Howard, M. Zhu, & Zhmoginov, A., and L.C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520 (2018).

51) A.D. Yao, D.L. Cheng, I. Pan, and F. Kitamura, "Deep learning in neuroradiology: a systematic review of current algorithms and approaches for the new wave of imaging technology," *Radiology: Artificial Intelligence*, **2** (*2*) e190026 (2020). doi:10.1148/ryai.2020190026.