The Performance Analysis of Transfer Learning for Steel Defect Detection by Using Deep Learning

M Abu¹, A Amir¹, Y H Lean¹, N A H Zahri¹, S A Azemi¹
¹Advanced Communication Engineering Centre (ACE), School of Computer and Communication Engineering, Universiti Malaysia Perlis

amizaamir@unimap.edu.my

Abstract. Detecting steel surface defects is one of the challenging problems for industries worldwide that had been used in manufacturing quality management. Manual inspection of the steel surface detects is a time-consuming process. This work aims at developing deep learning models that can perform steel defect detection and evaluating the potential of transfer learning for this task. In this paper, four types of transfer learning methods: ResNet, VGG, MobileNet, and DenseNet are experimentally evaluated to develop models for steel surface defect detection. The models were developed for binary classification (defect and no-defect) using the SEVERSTAL dataset from that contains 12,568 images of the steel surface. Then, these models were also assessed for multiclass classification using NEU dataset with 1800 images. In this work, image pre-processing is included to improve the result of steel defects detection. The experimental results have shown that the model developed by using the MobileNet method have the highest detection rate with 80.41% for the SEVERSTAL dataset and 96.94% for the NEU dataset compare to ResNet, VGG, and DenseNet transfer learning.

1. Introduction
Steel manufacturing is essential in the modern world of manufacturing, especially the out coming industrial revolution 4.0 [1]. Many industry fields that produce steels for the usage of an automobile, electronics, furniture, infrastructure, weapons, household materials, machines, shipbuilding, and so on. The quality control of the products is vital as it will affect the end-product quality. Steel defects could reduce the features of the products, which are steel corrosion resistance, abrasion resistance, and endurance limit, which will cause significant economic losses.

Commonly, the inspection of the quality of the steel items is conducted manually. Manual inspection of steel defects could lead to delay in the manufacturing process, and it is not an efficient way to warranty defect-free for steel manufacturing. The accuracy of identifying the steel defect might depend on the experience of the inspectors.

Manual inspection of steel defect is a troublesome and time-consuming process where the inspectors have to inspect each of the steel products manually. The unit per hour (UPH) rate of steel defect detection by inspectors is low. The delay will cause the reduction of production rate, thus results in substantial economic losses for manufacturing companies. Furthermore, if the undetected defects of steels are released to the end-users or customers, this could potentially lead to a bad reputation for the company.
As part of Industry 4.0, automatic steel defect detection is the key to steel manufacturing companies. These machine vision-based methods are being invented to solve the inefficiency of manual inspection, such as poor accuracy, time-consuming, high labor demand, and waste of resources. In the past decade, many researchers have studied machine learning techniques in defect detection, which is not only for the defect detection of steel. Many methods have been proposed to train neural network models in machine vision applications, such as object tracking, facial recognition, quality evaluation of fruits and vegetables. Deep learning is currently one of the hot topics among industries and academics. This trend involves in the manufacturing area. It is the field of learning a deep structured and unstructured representation of data. Deep learning is used in extensive complex data to obtain better results.

Several works have been conducted for steel defect detection by using deep learning. These works used various deep learning models such as compact CNN (Convolutional Neural Network) [2]; DAN (Deep Auto-encoder Network) [3]; YOLO (You Only Look Once) [4]; and Faster RCNN (Regions with Convolutional Neural Networks) [5]. Based on these studies, deep learning shows promising results for steel defect detection. However, the computational resources needed to train deep learning are expensive to buy, where most of the current works performed training on a high-performance computer with GPU. This requirement impedes the study to use deep learning for steel defect detection in the absence of such a high-performance machine.

The use of transfer learning methods can reduce the training time of deep learning models significantly. Hence, this research studied four transfer learning methods (ResNet, VGG, MobileNet, and DenseNet) and applied these methods for the development of a deep learning model for steel defect detection. The outcome from this preliminary research can further improve in the future by incorporating the suitable transfer learning method in designing the new architecture of a deep learning model for steel defect detection.

2. Literature Review
The previous researchers have proposed several methods for automatic steel defect detection by using deep learning. In Tao et al. (2018) [2], they discussed the procedure of accurately localizing and classifying defects that appeared on the surface of metallic. There are two parts of the process of identifying the metallic defect. The first part is to design a novel cascaded autoencoder (CASAE) architecture for segmentation and localizing defects. The CASAE cascading network transforms the input image of the steel surface into a pixel-wise prediction mask based on semantic segmentation. The second part of the system used a compact convolutional neural network (CNN) that is responsible for classifying the defect regions of segmented results in their specific classes.

In the paper by Kholief et al. (2017) [3], they discussed the detection and classification of hot roll steel strip surface defects. The feed-forward artificial neural networks and autoencoder network were used as a classifier to train for detecting six categories of steel defects: crazing, patches, pitted surface, inclusion, rolled-in scale, and scratches. The DAN technique processes massive raw steel defects data and automatically provides accurate results without extracting features.

The You Only Look Once (YOLO) network was designed with all convolutional in Li et al., (2018) [4]. The proposed system can predict the location and size of the defect regions. YOLO is a convolutional detection network that automatically extracts multi-scale features of steel surface defects and recognizes the defect regions [4]. In this research, the pooling layers are replaced with convolutional layers and made it all convolutional. Therefore, the network can learn its spatial down-sampling [6]. The YOLO network can simultaneously predict the location, class, and size of defect regions to increase the steel quality in production lines.

Lin et al. (2019) [5] evaluated two deep learning models, which are Faster RCNN (Regions with Convolutional Neural Networks) and SSD (Single Shot Multibox Detector) for steel defect detection. Fast RCN has better speed compare to RCNN because Fast RCNN simplifies the feature extraction process. Furthermore, Fast RCNN proposed the ROI (Region of Interest) pooling layer for softmax operations. SSD uses VGG (Visual Geometry Group) encoder to extract feature maps, and it combines feature images with different scales to create detectors. Next, the SSD method uses non-maximum
suppression to merge any possible bounding boxes to get final results. The results showed that the SSD could achieve high performance for region-based object detection. Also, the precision and recall rate of SSD is higher than Faster RCNN.

Kim et al. (2019) [7] proposed a Siamese Neural Network by using CNN (Convolutional Neural Network) with the contrastive loss for few-shot learning to classify few steel defects images. To solve the limited sample images used to train the network with a few images with labels, the solution is using few-shot learning techniques [8][9]. In the proposed classification technique, a pair of classes is generated, then, it is trained with the few-shot network with some pairs. Features from each image are extracted from the generated pair. Next, the L1 distance is calculated for each feature. Afterward, images in a pair are determined whether those are from the same classes or not, based on the calculated L1 distance [10][11].

Our review of related works has shown that deep learning has been used widely for steel defect detection, where the models were trained from scratch. The main issue of deep learning training is its complexity, and it requires high processing capability to train, especially when it involves large data. In contrast to these methods, we investigate the potential of four transfer learning methods for the steel defect detection problem.

### 3. Methodology

In this work, the dataset used is the SEVERSTAL dataset and NEU dataset. Both datasets will be trained using four types of transfer learning models VGG16, MobileNet, DenseNet121, and ResNet101. The result of each transfer learning model will be compared. Figure 1 shows the flowchart of steel defect detection.

![Flowchart of steel defect detection](image)

#### 3.1. Data Collection

For data collection, there are two dataset used for transfer learning analysis. First, SEVERSTAL steel defect dataset with 12,568 grey scale images of steel surface with and without defects [12] and second is NEU dataset with 1800 images of steel surface with six classes of steel defect ‘rolled in scale’,
‘patches’, ‘crazing’, ‘pitted surface’, ‘inclusion’ and ‘scratches’ [13]. These two dataset are train and test separately for each transfer learning method. Figure 2 and Figure 3 show the example of steel defect images on both datasets.

Figure 2  Examples of the defect and non-defect images in the SEVERSTAL dataset [12].

Figure 3  Defect image of 6 class of defect image on the NEU dataset [13].

3.2. Pre-processing
The dataset will undergo pre-processing to remove noise in the data before it is sent to transfer learning models. The dataset that is collected from different sources might contain unwanted distortions, which are not feasible for the analysis. Next, data is consisting of attributes with different scales. Rescaling the attributes will benefit the deep learning algorithms and make the learning process runs faster. Therefore, there is a need to rescale the images into proper size in order to make sure all images have equal sizes and resolutions.

Rescaling [7] the input images to a suitable size is essential for image classification because if the image are too small, image overlapping can occur. Still, if the image size is too big, it is time consuming for training dataset. The pre-processing stage can be done by using OpenCV (Open Source Computer Vision). OpenCV is a Python’s library; it is developed to solve computer vision problems. By using OpenCV, the steel defect images will be rescaled to 256x480 pixels, which is a recommended size for the SEVERSTAL dataset and 32x32 pixels for the NEU dataset.

3.3. Transfer Learning
Transfer learning is a machine learning technique to solve the fundamental problem of insufficient training data [14]. A model is trained and developed for one task and is then reused on a second related task. It refers to the situation whereby what has been learned in one setting is exploited to improve optimization in another environment [15]. It is usually applied when there is a new dataset smaller than the original dataset used to train the pre-trained model [16].

This work proposes a system that uses the transfer learning model. The pre-trained model of transfer learning already trained on ImageNet database (The ILSVRC dataset that already trained for transfer learning pre-trained) [17]. With regards to the initial training, transfer learning allows the model to start with the learned features on the ImageNet database and adjust these features and the
structure of the model to suit the new dataset instead of beginning the learning process on the data from scratch with random weight initialization [16].

The transfer learning proposed by this work are explained below:

i) VGG-Net: VGG-Net are built with 3x3 kernel, stride=1, and padding=same. It contained eight layers; the first five layers were convolutional layers followed with max-pooling layers, and the last three are fully connected layers. ReLU activation function is used in this model [17]. There are two versions of VGGNet; VGG16 and VGG19.

ii) ResNet: There are four version of ResNet; ResNet34, ResNet50, ResNet101 and ResNet152. This model was designed to skip the dead connection and decrease the effect of layers on the performances [17]. The number of layers for ResNet is defined based on its version; for example, ResNet34 has 34 layers deep. In this work, Resnet101 is proposed because Akshay & Harisson (2019) had done experiment on different layer of ResNet and obtained the best accuracy at 90.6% with the SEVERSTAL dataset by unfreeze 98 last layers of ResNet101 [12]. So, this method will be used by unfreeze all layer off ResNet to see if the performance increase if the layer increase and compare it performance with NEU dataset that have small dataset.

iii) DenseNet: There are three version of DenseNet; DenseNet121, DenseNet169 and DenseNet201. DenseNet121 is chosen for this work to save time during the training process. In DenseNet, each layer obtains collective knowledge from all preceding layers and passes on its own feature maps to all subsequent layers so the network can be thinner and compact. For example, the number of channels can be fewer. [18].

iv) MobileNet is lightweight in its architecture. It uses depth-wise separable convolutions, which basically means it performs a single convolution on each color channel rather than combining all three and flattening it. This has the effect of filtering. This model has 30 layers deep, with normal, depth-wise and point-wise convolutions layer (apply a filter to an input to create a feature map that summarizes the presence of detect feature). It is very low maintenance thus performs well with high speed [19].

Figure 4 shows an example of transfer learning architecture.

Figure 4  Example of Transfer Learning Architecture

4. Result and Discussion
The detection of steel defect will be written in python interphase because python is suitable for a robust programming language that has its main focus on rapid application development. The package used to develop this system is KERAS (Tensorflow backend). Therefore, a high-performance GPU and CPU in Google Colab is used to run the algorithm.
4.1. Validation

For validation, the dataset will be split into 80:20, which is 80% of the dataset will be used as the sample images to train the model. After the training process, the deep learning model for the classification of the steel defects is developed. Next, the deep learning model can be used to classify the test data, which is 20% of the total dataset. The parameter used for transfer learning is 50 epochs and 32 batch-sizes is chose because the data used is small. After all, this method is frequently used by the previous techniques and show better result. The datasets will be trained in each of the transfer learning models to obtain the highest performance of steel defect detection. Moreover, the transfer learning model is trained with Google Colab on Tesla K80 GPU and 64GB RAM, the training time is significantly reduced due to its high performance GPU. There are several processes of techniques to trained steel defect dataset with transfer learning:

i) The dataset is minimal but similar to the ImageNet database: all the layers need to be freeze, and some dense layers are added to train the model [17].

ii) The dataset is large and similar to the ImageNet database: the complete network is fine-tuned with a small learning rate [17].

iii) The dataset is medium size and similar to the ImageNet database: only fine-tune the last layers and freezes the top layers [17].

iv) The dataset is significant but not similar to the ImageNet database: initialize the weights of network with the pre-trained model and retrain the whole model [17].

In this work, the latter method will be used by retrained the whole model for SEVERSTAL dataset and NEU dataset because this dataset is not similar in ImageNet. SEVERSTAL dataset is taken from SEVERSTAL Company [20], and the NEU dataset is taken from Northeastern University [13].

4.2. Result

After that, the recorded accuracy for each transfer learning for SEVERSTAL and NEU dataset will be compared. From Table 1, the accuracy of SEVERSTAL and NEU dataset is shown. MobileNet model shows the highest accuracy when retrained the whole model for both datasets, and the VGG16 model shows the lowest accuracy for both datasets. It is because VGG16 has the smallest layers compare to other models. Besides that, the bigger the layer's size of deep learning does not ensure to obtain higher accuracy because the accuracy will be looping when it achieves high efficiency until the iteration stops. Accuracy between CPU and GPU in Google Colab give same result with differences ~5 %.

| Dataset   | Model   | Time Taken (%) | CPU | GPU |
|-----------|---------|----------------|-----|-----|
| SEVERSTAL | VGG16   | 50.00%         | 50.00% |       |
|           | MobileNet | 79.91%         | 80.41% |       |
|           | DenseNet121 | 70.34%         | 70.27% |       |
|           | Resnet101 | 70.50%         | 72.35% |       |
| NEU       | VGG16   | 78.08%         | 78.33% |       |
|           | MobileNet | 96.86%         | 96.94% |       |
|           | DenseNet121 | 88.23%         | 93.72% |       |
|           | Resnet101 | 81.11%         | 80.27% |       |

Figure 5 and Figure 6 show the training and validation accuracy of the MobileNet model for SEVERSTAL and NEU dataset.
The time taken for each transfer learning is tested in GPU and CPU. The time taken for GPU is faster compared to CPU as the layers of transfer learning increase; more time is needed to training the dataset. The time taken for each model is compared in Table 3.

### Table 2  Time Taken for Each Transfer Learning model

| Dataset   | Model       | Time Taken (s) |  
|-----------|-------------|----------------|
|           | CPU         | GPU            |
| SEVERSTAL | VGG16       | 54009.08       | 1439.28        |
|           | MobileNet   | 19401.55       | 572.32         |
|           | DenseNet121 | 86700.00       | 2139.76        |
|           | Resnet101   | 140883.09      | 5348.24        |
| NEU       | VGG16       | 7463.01        | 149.09         |
|           | MobileNet   | 960.99         | 62.50          |
|           | DenseNet121 | 1526.08        | 229.60         |
|           | Resnet101   | 9980.35        | 589.82         |

Time taken for deep learning not only based on the layers of deep learning parameter of deep learning also may affect the coefficient of deep learning. [21] Beside that, because Google Colab runs in cloud, internet lost time delay will occur.

### 5. Conclusion

The outcome of this project is to evaluate the performance of the transfer learning models used in this project for classifying the steel defect on the SEVERSTAL dataset and NEU dataset. Our results show that MobileNet performed the best compared to ResNet, VGG, and DenseNet. The model developed by using MobileNet enables the classification of defects in steel with identifying rate up to for 80.41%
SEVERSTAL dataset and 96.94% for NEU dataset by using GPU. It shown GPU is better than CPU in recognition and time coefficient.

At the image pre-processing stage, different parameters will affect the prediction outcome. Image size may affect model training capability. In this work, SEVERSTAL dataset are resized into 256x480 because small size of image gives accuracy less than 50% for all transfer learning model. For NEU datasets, the images are resized into 32x32. The dataset also had been resized same size as SEVERSTAL dataset and give same accuracy but it is time consuming. Others, problem occurs in this work is chose batch size and epoch for each transfer learning. If larger size of batch size and epoch used the model will be overfitting. If larger size of batch size and epoch the model will be underfitting.

In the future, we aim to improve the performance of the steel defect detection by devising a new architecture of deep learning network. By incorporating MobileNet model as feature map and combine it with custom layer of convolutional neural network, we may speed up the training time while ensuring the accuracy of the defect detection.

Acknowledgments
We are thankful to our colleagues who provided expertise that greatly assisted the research, although they may or may not agree with all of the interpretations provided in this paper and we also thankful for Fundamental Research Grant Scheme (FRGS900300646 1/2018) has support this work.

References
[1] S. Guan, M. Lei and H. Lu. (2020, 10 March) "A Steel Surface Defect Recognition Algorithm Based on Improved Deep Learning Network Model Using Feature Visualization and Quality Evaluation," in IEEE Access, vol. 8, pp. 49885-49895, doi: 10.1109/ACCESS.2020.2979755
[2] Xian Tao, D. Z. (2018, September 6). Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks. Retrieved November 15, 2019
[3] Kholief, E. (2017). Detection of Steel Surface Defect Based on Machine Learning Using Deep Auto-encoder Network. In Proceedings of the 7th International Conference on Industrial Engineering and Operations Management (IEOM), Rabat, Morocco.
[4] Li, J. Z. (2018). Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. IFAC-PapersOnLine, 51(21), 76-81.
[5] Springenberg, J.T., Dosovitskiy, A., Brox, T., & Riedmiller, M.A. (2015). Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806.
[6] Lin WY., Lin CY., Chen GS., Hsu CY. (2019) Steel Surface Defects Detection Based on Deep Learning. In: Goonetilleke R., Karwowski W. (eds) Advances in Physical Ergonomics & Human Factors. AHFE 2018. Advances in Intelligent Systems and Computing, vol 789. Springer, Cham
[7] M. S. Kim, T. P. (2019). Classification of Steel Surface Defect Using Convolutional Neural Network with Few Images. 12th Asian Control Conference (ASCC), 1398-1401.
[8] Fei-Fei, L., & Fergus, R. (2006). One-shot learning of object categories. IEEE transactions on pattern analysis and machine intelligence, 28:4: 594-611.
[9] Miller, M. a. (2000). Learning from One Example through Shared Densities on Transforms. Computer Vision and Pattern Recognition,
[10] Vinyals, O. (2016). Matching networks for one-shot learning. Advances in Neural Information Processing Systems, 3630-3683.
[11] Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. ICML Deep Learning Workshop.
[12] Akshay A. et al. (2019). CNNs for Bulk Material Defect Detection. Retrieved January 25, 2019
[13] K. Song, Y. Y. (2013). A robust noise method based on completed local binary patterns for hot-rolled steel strip surface defects. Applied Surface Science, 285(November 2013), 858-864.
[14] Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C. (2018) . A Survey on Deep Transfer Learning. In: Kůrková V., Manolopoulos Y., Hammer B., Iliadis L., Maglogiannis I. (eds) Artificial Neural Networks and Machine Learning – ICANN 2018. ICANN 2018. Lecture
Notes in Computer Science, vol 11141. Springer, Cham

[15] Gao, Y. and Mosalam, K. (2018). Deep Transfer Learning for Image-Based Structural Damage Recognition. Computer-Aided Civil and Infrastructure Engineering.

[16] Hussain M., Bird J.J., Faria D.R. (2019) A Study on CNN Transfer Learning for Image Classification. In: Lotfi A., Bouchachia H., Gegov A., Langensiepen C., McGinnity M. (eds) Advances in Computational Intelligence Systems. UKCI 2018. Advances in Intelligent Systems and Computing, vol 840. Springer, Cham

[17] Akshay, G. (2019). Transfer Learning with Keras. Retrieved November 15, 2019

[18] Sik H. T. (2018). DenseNet-Dense Convolutional Network (Image Classification). Retrieved November 25, 2018

[19] Ferhat, C. (2018). Transfer Learning using Mobilenet and Keras. Retrieved November 15, 2019

[20] Severstal: Steel Defect Detection Dataset http://www.kaggle.com/

[21] Keras Application. Retrieve from http://keras.io/api/applications/