Paper No: SC-21 **Smart Computing**

Application of AlexNet convolutional neural network architecture-based transfer learning for automated recognition of casting surface defects

Shiron Thalagala*
Dept. of Electromechanical Engineering
University of Macau, China
shironceylon@gmail.com

Chamila Walgampaya
Dept. of Engineering Mathematics
University of Peradeniya, Sri Lanka
ckw@pdn.ac.lk

***Abstract* - Automated inspection of surface defects is beneficial for casting product manufacturers in terms of inspection cost and time, which ultimately affect overall business performance. Intelligent systems that are capable of image classification are widely applied in visual inspection as a major component of modern smart manufacturing. Image classification tasks performed by Convolutional Neural Networks (CNNs) have recently shown significant performance over the conventional machine learning techniques. Particularly, AlexNet CNN architecture, which was proposed at the early stages of the development of CNN architectures, shows outstanding performance. In this paper, we investigate the application of AlexNet CNN architecture-based transfer learning for the classification of casting surface defects. We used a dataset containing casting surface defect images of a pump impeller for testing the performance. We examined four experimental schemes where the degree of the knowledge obtained from the pre-trained model is varied in each experiment. Furthermore, using a simple grid search method we explored the best overall setting for two crucial hyperparameters. Our results show that despite the simple architecture, AlexNet with transfer learning can be successfully applied for the recognition of casting surface defects of the pump impeller.**

Keywords - automated inspection, casting defect detection, convolutional neural networks, hyperparameters, transfer learning

I. INTRODUCTION

Cost and time effective quality management [1] in a manufacturing operation is a significant aspect regardless of the domain. Nevertheless, producing higher quality products that yield higher customer satisfaction with the least cost and time has been a challenging task for manufacturing firms. Product visual inspection for defects, being a crucial element in quality management, is increasingly automated in present manufacturing firms due to numerous benefits [2] which ultimately result in higher business performance.

Metal casting is a manufacturing process where molten metals are solidified in a mold to obtain the required shape [3]. Though metal casting processes span across a wide variety of metals and several specific techniques, the most common defect types can be categorized as blowholes, shrinkages, cracks, sand inclusions, defective surfaces, and mismatches [4]. Proper identification of casting defects effectively is vital as unnoticed defective finished products which go to the customers' hand can cause fatal mechanical failures [5]. Automating the process of visual inspection of metal castings with the aid of intelligent systems [6] is beneficial in terms of accuracy, inspection time, and cost. Especially, it prevents the facilitation of human labor in hazardous environments including costly concerns of the safety of such employees.

The visual identification process of defects in metal castings needs to entertain two main requirements during the process of inspection. One is the identification of surface defects on the casting, and two is the identification of defects located inside the cast product which are not visible to the naked eye. The latter is relatively complicated and expensive, commonly accomplished by non-destructive testing (NDT) methods such as ultrasonic testing, eddy-current testing, magnetic particle testing, and radiographic (X-ray) testing [7].

The main purpose of non-destructive testing is to identify defects located inside the test object by the naked eye without damaging the object. X-ray computer tomography (XCT) is a widely used non-destructive casting inspection method that generates two-dimensional/three-dimensional images of the object interior structure [8]. Inspecting such interior images along with the inspection of casting surfaces of every manufactured product is necessary to maintain lower defect levels. Not only the interior images generated by XCT but also the conventional photographs of the casting surfaces can be fed into intelligent systems that use image processing and machine learning techniques for recognition, categorization, and localization of casting defects [6].

Convolutional neural networks (CNNs), which lie in the domain of machine learning have been well studied for their appropriateness in computer vision applications [9]. The structure of CNNs is analogous to that of the connectivity pattern in the visual cortex of the human brain. CNNs are capable of extracting features by themselves and there is no need to perform manual feature extractions in the input images which, however, is essential in some primitive machine learning techniques. Fig. 1 illustrates the difference in image classification approach between primitive machine learning methods and CNNs. Hence, over the last decade, CNNs have successfully applied for automated inspection of casting defects with varying performances [10]–[12]. Since the onset of the CNNs, numerous architectures have been generated by carrying out structural reformulations, regularizations, parameter optimizations, etc. [13]. AlexNet [14] is a prominent CNN architecture that performs competently in the tasks of image recognition. While CNNs perform better in the realm of images over traditional machine learning techniques still some common hindrances for lack of generalization of models are not fully conquered by research. Specifically, models trained for the same feature space and the same distribution drastically reduce their performance when

tested on a different dataset with different feature distribution.

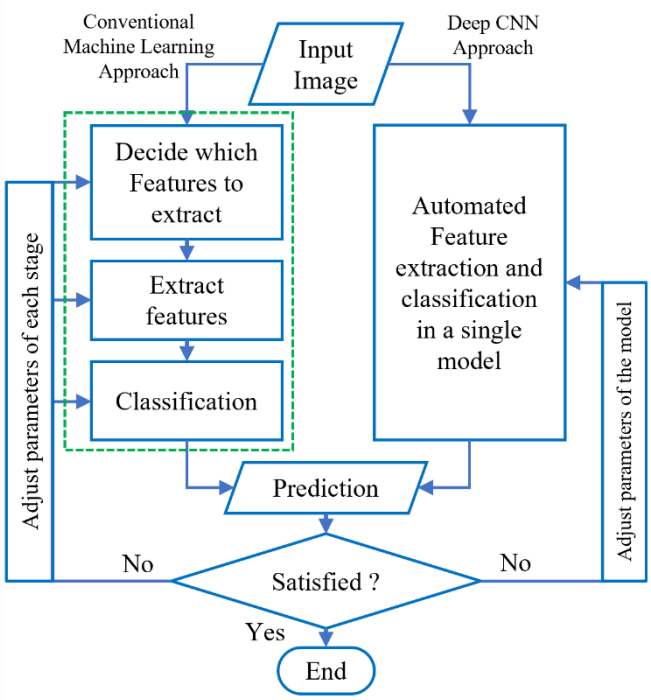

Fig. 1. Difference in image classification approach between conventional machine learning techniques and CNNs

Transfer learning has significantly addressed the issue of using a single CNN model for the recognition tasks in different image fields. Transfer learning in CNNs is the use of knowledge gained by training a model in one domain, on another in a dissimilar domain [15]. It helps not only to mitigate the computational cost in training but also to generalize the CNN models over different domains. Moreover, transfer learning is beneficial in situations when adequate data is lacking for learning from scratch. Despite the successful applications of transfer learning in automated recognition of casting defects, selection of the unique CNN model parameters (hyperparameters) [16] relevant to each casting image dataset is still necessary.

This paper focuses on: (1) investigating the application of an AlexNet CNN model which is pre-trained on an entirely different larger dataset to recognize images of casting surface defects, and (2) optimizing hyperparameters for best performance. The pivot of this study is a classification task to segregate faulty casting products in a manufactured batch through pattern recognition. Further classification of defect types or localization of defects, however, are out of the scope of this study. The dataset [17] used in the study comprised only two classes named 'defect' and 'defect-free' representing images with one or more defects, and images without any visible defect, respectively.

II. Related Work

Recognition and localization of manufacturing defects using machine learning techniques are explored in numerous studies over the recent years with the focus of achieving high-performing robust models. Several primitive computer vision techniques were used by several authors at the early stages of the pattern recognition field. A background subtraction method followed by a thresholding algorithm is proposed in [18]. The idea is to generate an image with the same pixel intensities as the original image except defective regions using low-pass filtering [19]. The newly constructed image is then subtracted from the original image resulting in a residual image containing only defective regions. In [20] the Modified Median filter, MODAN-Filter, is proposed to identify contours of the casting defects from non-defective areas with a function to calculate the pixel values of the background image. Furthermore, equations of the MODAN-Filter are generalized in [21] to achieve higher robustness. These filtering-based methods that depend on optimum filter parameters, however, can be unreliable when image noise is present substantially. In [22], the wavelet transform method is described as a potential technique to identify certain casting defect types.

Feature-based detection of casting defects is another trending approach that can be seen applied in [10], [23]. During this process, each pixel is classified as a defect or not based on the features calculated using sets of nearby pixels. Common features include statistical descriptors such as mean, standard deviation, skewness, kurtosis, energy, and entropy [24]. In [25], a hierarchical and a non-hierarchical linear classifier has been implemented based on six geometric and gray value features namely contrast, position, aspect ratio, width-area ratio, length-area ratio, and roundness. A Fuzzy logic-based method for the detection and classification of defects that appear in the radiographic images is proposed in [11].

Many modern studies have tested numerous CNN architectures in terms of the performance and accuracy of casting defect recognition tasks. Among those, Region-Based Convolutional Neural Networks (R-CNNs) are used for the automatic localization of casting defects significantly [12]. R-CNNs are capable of setting bounding boxes around categorical patches in the images where this can be implemented easily to mark the defects in the casting defect images. In [10], a new CNN architecture called Xnet-II is introduced which comprises five convolutional and fully connected layers. Moreover, they have used a dataset generated through simulation using Generative Adversarial Networks (GAN) [27] instead of real casting defect images.

Lack of sufficient data is a common problem in the machine learning domain. Data augmentation where new images are generated by augmenting the existing images of casting defects efficiently and accurately with low background noise is proposed in [28]. This mechanism is based on a traditional image enlargement technique, precisely forcing the CNN to learn more in the regions of the image that need high attention in order to perform better in the classification task. On the other hand, transfer learning is effective not only in the lack of data scenarios but also in respective to the robustness of the model. In [5], the authors use ResNet CNN architecture for the recognition of casting defects. When compared to AlexNet, due to the architectural complexity, ResNet needs a significantly larger number of computations which ultimately consumes higher computational resources.

III. Methodology

In this section, we explain the approach used to recognize casting surface defects of an industrial product using AlexNet CNN architecture and transfer learning.

Improving the accuracy and the robustness of the AlexNet architecture using transfer learning in the context of casting defect detection is the major objective of this study.

A. Description of the dataset

The dataset, obtained from Kaggle datasets [17], consists of images of a submersible pump impeller which is manufactured as a casting product. All the images depict the top view of the impeller and belong to two classes. The images that exhibit at least one casting defect on the surface of the impeller are labeled as defect while all the other images, conversely, are labeled as defect-free. i.e., Any casting defect on the surface that cannot be identified by the naked eye from the images is labeled as defect-free.

This dataset is collected under stable lighting conditions with a Canon EOS 1300D DSLR camera. The dataset contains a total of 1300 gray-scaled images with the dimensions of each as (512×512) pixels. Among those, 781 images are labeled as defect, and the remaining 519 images are labeled as defect-free. Fig. 2 shows eight sample images (size and the resolution is altered in order to adhere to paper guidelines) and corresponding labels which are randomly picked from the two classes. All the images acquired for this study from the original dataset are only the raw images and the augmentation is done as a part of this study.

B. Image augmentation

In this section, we discuss the image data augmentation techniques applied for the dataset before the experimentation. As in [29], several classical techniques that belong to geometrical and color-based transformations were applied randomly to yield higher variability. As per geometric transformations, rotation, shearing, mirroring, scaling (zoom-in/out) and translation were applied. Nevertheless, color space transformations were limited only to change of apparent brightness as the dataset already contains grayscale images. Moreover, apparent brightness change (performed randomly) in each pixel intensity of an image was restricted to a maximum of 20% (either increase or decrease) of the current intensity. It prevents introducing new defect regions which were not in the original image or disappearing significant regions of the image with low intensities by further decreasing the intensity.

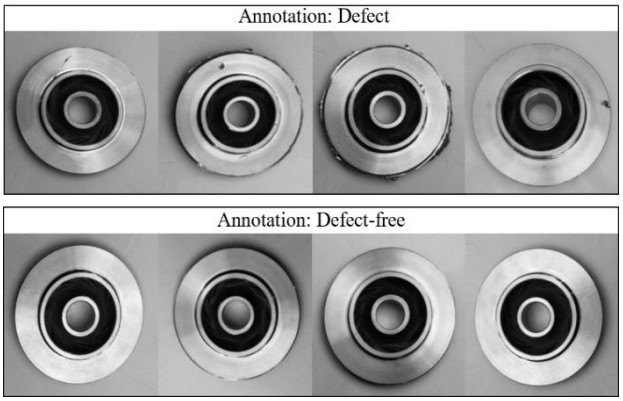

Fig. 2. Randomly picked eight number of sample images from the dataset annotated as defect and defect-free

Fig. 3 shows one sample image (annotated as defect-free) and corresponding images synthesized by augmenting that image using all the techniques used in this study. Among synthesized images, 5814 are annotated as defect-free and 7668 are annotated as defects. At last, all the images were resized to (224×224) pixels. Throughout all the experimentation, training and validation data split is diversified by changing the amount of training data to 20%, 40%, 50%, 60%, and 80% to understand the capacities of generalization of the used models [30]. Hereinafter, the ratio between the training image set and the validation image set will be referred as train-test split ratio.

C. Non-parametric classification using the k-nearest neighbor algorithm

K-Nearest Neighbor (KNN) algorithm, which is a basic supervised machine learning algorithm, is used to investigate the capability of performing the classification task using raw pixel intensities as the input and without any sophisticated feature extraction techniques.

In the context of computer vision, the KNN algorithm performs classification of the data points (pixel values) based on the distance between them and with the assumption that similar features exist nearby. Common methods of calculating the distance include the Euclidean distance:

$$d(p,q) = \sqrt{\sum_{i-1}^{N}(q_i - p_i)^2} \quad (1)$$

and the Manhattan/city block distance:

$$d(p,q) = \sum_{i-1}^{N}|q_i - p_i| \quad (2)$$

where $d(p,q)$ is the distance between two p and q points in the image spatial domain with N pixels.

In this study, the KNN algorithm is performed with the raw pixel intensities of casting images without any feature extraction with the Manhattan distance calculation metric and the k value equals to five. The variation of precision, recall, and f1-score is observed by varying the train-test split ration.

D. CNN architecture

Despite the emerging CNN architectures, we base our model around AlexNet architecture due to three reasons. (1) To the best of our knowledge, application of AlexNet based transfer learning in recognition of casting defects is not addressed in past literature, (2) AlexNet is applied in a diverse set of deep learning problems witnessing promising results [30], [31], (3) AlexNet, which was proposed in 2012, is regarded as the first deep CNN architecture which showed pioneering results in image recognition and classification tasks [32]. We show that AlexNet is sufficiently deep and reliable for a modest classification of casting surface defects when compared to other deeper sophisticated architectures born after AlexNet, if hyperparameters are properly optimized.

AlexNet consists of five 2D convolutional layers (Conv2D) followed by three fully connected layers (FC). The build of the AlexNet architecture is illustrated in Table I and it is constructed with several common CNN components

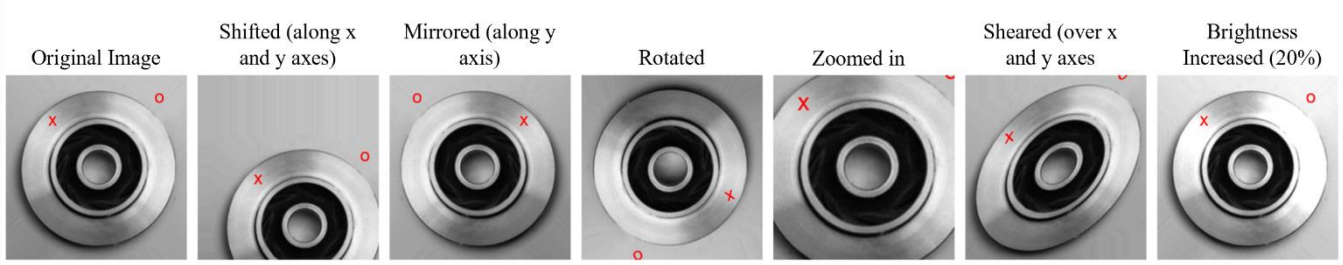

Fig. 3. Six transformations applied to a single original image (the symbols 'x' and 'o' in red color are used to understand the transformation in respect to the original image). Relevant transformation is labeled on top of the image.

a) Convolution layers

Each convolutional layer consists of a set of filters known as convolutional kernels where each neuron plays the role of a kernel. The kernel is a matrix of integers where it will multiply its weights with corresponding values of a subset of pixels of the input image. The selected subset of pixels of the input image has a similar dimension to the kernel. Then, the resulting values are summed up to generate one value that represents the value of a pixel in the output (feature map). The kernel strides across the input image producing the output (feature map of the entire image) of the convolution layer. In each layer, the kernel strides over a varying number of pixels at a time in both dimensions (height and width). The convolution process can be mathematically expressed as [33]:

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y).e_l^k(u,v) \qquad (3)$$

where, $i(x,y)$ is an element of the input image tensor with x and y coordinates, which is element-wise multiplied by $e(u,v)$ index of the k^{th} convolutional kernel of the l^{th} layer. u and v are the rows and columns of the kernel matrix. $f(p,q)$ is the corresponding output feature map with p columns and q rows while c is the image channel index.

b) Pooling layers

Pooling operation sums up identical information in the local region of the feature map generated by a convolutional layer and outputs a single value within that region [34]. AlexNet consists of three pooling layers followed by the first, second and last convolution layers.

c) Activation function

Use of Rectified Linear Unit (ReLU) as a non-linear activation function of each layer is a significant characteristic in AlexNet. ReLU activation function is:

$$R(z) = \max(0, z) \qquad (4)$$

where z is the function input and $R(z)$ is the function output which equal to the input when the input is positive and equal to zero otherwise.

d) Batch normalization

As a countermeasure for the overfitting, batch normalization is performed after several layers of the AlexNet.

TABLE I. LAYERS OF THE ALEXNET ARCHITECTURE

ID	Layer Type	Layer Parameters (f=no. of feature maps, k=kernel size, s=strides, act=activation function)	Size of Feature Map
0	Input layer	Input image size=(224x224) pixels, Channels=1	224×224×1
1	Conv2D	f=96, k=(11 x 11), s=4, act=ReLU	55×55×96
2	Max Pool	f=96, k=(3 x 3), s=2,	27×27×96
3	Batch normalization	N/A	27×27×96
4	Conv2D	f=256, k=(5 x 5), s=1, act=ReLU	27×27×96
5	Max Pool	f=256, k=(3 x 3), s=2,	13×13×256
6	Batch normalization	N/A	13×13×256
7	Conv2D	f=384, k=(3 x 3), s=1, act=ReLU	13×13×384
8	Batch normalization	N/A	13×13×384
9	Conv2D	f=384, k=(3 x 3), s=1, act=ReLU	13×13×384
7	Batch normalization	N/A	13×13×384
11	Conv2D	f=256, k=(3 x 3), s=1, act=ReLU	13×13×256
12	Max Pool	f=256, k=(3 x 3), s=2,	6×6×256
13	Batch normalization	N/A	6×6×256
14	Dropout	Rate=0.5	6×6×256
15	FC	f, k, s are N/A, act=ReLU	4096
16	Dropout	Rate=0.5	4096
17	FC	f, k, s are N/A, act=ReLU	1024
18	Dropout	Rate=0.5	1024
19	FC	f, k, s are N/A, act=softmax	2

e) Fully connected layer

At the end of the feature extraction stage (accomplished by convolutional layers), three fully connected layers are introduced which perform classification globally [35].

f) Dropout

To achieve generalization, some units or connections with a certain probability within the network are randomly

skipped (dropout) [36]. The AlexNet model executes dropout after several fully connected layers in it.

g) Output layer

The final layer of AlexNet architecture which acts as the output layer uses the softmax activation function [37]. The softmax function is given by:

$$S(y_i) = \frac{exp\,(y_i)}{\sum_{j=1}^{n} exp\,(y_j)} \quad (5)$$

where $y^i$is the i^{th} element of the input vector, n is the number of classes which, in our case is two–defect and defect-free.

In our study, four modifications were carried out on the original AlexNet model creating an AlexNet variant. The modifications are: (1) Number of channels in the input convolutional layer is changed from three to one as our dataset consists of only grayscale images, (2) Dropout is imposed after each fully connected layer, (3) Batch normalization is performed after third and fourth convolutional layers, and (4) Number of output features of the second fully connected layer changed from 4096 to 1024.

E. Application of transfer learning and optimizing model hyperparameters

ImageNet dataset [38] is used for pre-training of the AlexNet model and the influence of the transfer learning is tested using three experimental configurations (EC):

- EC1: AlexNet is trained with the casting surface defect dataset without any pre-training with weights initialized randomly (training from scratch).
- EC2: the same process in the previous configuration repeated, but the weights initialized with the ones found from the pre-trained model instead of random weights.
- EC3: the exact weights of all the feature extraction layers pre-trained on the ImageNet dataset were used.
- EC4: the entire model parameters (including both parameters of convolutional and fully connected layers) of the pre-trained model on the ImageNet dataset is used on the casting surface defect dataset.

In each configuration, two types of hyperparameters including optimizer [39] and learning rate are optimized using the grid search method to achieve higher accuracy with modest robustness. In the grid search method, all the possible combinations of the selected hyperparameters are tested in multiple trials. The grid search methods suffers from the curse of dimensionality [40] where the number of trials grows exponentially with the increase of the number of hyperparameters. Nevertheless, the other sophisticated optimizations are not used as we obtained sufficient accuracies by varying only the two aforementioned hyperparameters.

F. Implementation

Training of the AlexNet model is accomplished using the Google Collaboratory tool–a free online python programming environment specially designed for machine learning tasks. CPU is composed of a single core hyper threaded Intel Xeon Processors at 2.3Ghz speed and 13GB RAM while GPU is a Tesla K80 GPU with a 12 GB GDDR5 VRAM.

For the implementation of the AlexNet model and KNN, TensorFlow [41] and Scikit-learn [42] open-source tools are used. TensorFlow is an open-source framework designed for the implementation and experimentation of machine learning-related tasks while Scikit-learn is a high-level machine learning library for python programming language. Furthermore, pre-trained models including the weights are acquired using PyTorch–an open-source deep learning framework [43].

All the experiments ran for ten epochs, where epochs are the number of training iterations where each neural network accomplishes one learning instance over the dataset. The selection of ten epochs is based on the empirical observation that conveys all the training in each experiment is always converged with ten epochs with optimal hyperparameters.

TABLE II. PRECISION, RECALL AND F1-SCORE OF THE TWO CLASSES OBTAINED AFTER CLASSIFICATION USING KNN ALGORITHM

	Defect			**Defect-free**		
Test: Train	***Precision***	***Recall***	***F1-Score***	***Precision***	***Recall***	***F1-Score***
0.2:0.8	0.86	0.87	0.88	0.86	0.81	0.83
0.4:0.6	0.85	0.88	0.86	0.84	0.79	0.81
0.6:0.4	0.85	0.88	0.86	0.84	0.79	0.81
0.8:0.2	0.84	0.82	0.83	0.77	0.79	0.78

IV. RESULTS AND DISCUSSIONS

This section presents the results obtained by following the methods discussed in the previous section and related interpretations.

A. Classification without learning

The results of the KNN classification of the casting surface defect dataset are presented in this section. Table II shows precision, recall, and the f1-score corresponding to each class (defect and defect-free) obtained after performing the KNN algorithm with varying the train-test split ratio. With the reduction of the training set percentage, there is no significant gradual change in the accuracy as there is no learning that occurred during the training process by the KNN algorithm unlike the learning models discussed in this paper.

The overall average accuracy of the classification of casting surface image data using the KNN algorithm is relatively lower when compared to the results of CNN models discussed in the future sections. This lower accuracy reveals that the classification using raw pixel intensities and their proximities to neighbor values in the casting surface defect images are not significant. This phenomenon discloses that all the images in each class are unique up to a certain extent in respect of pixel intensities which in return, induces the importance of the feature extraction. On the

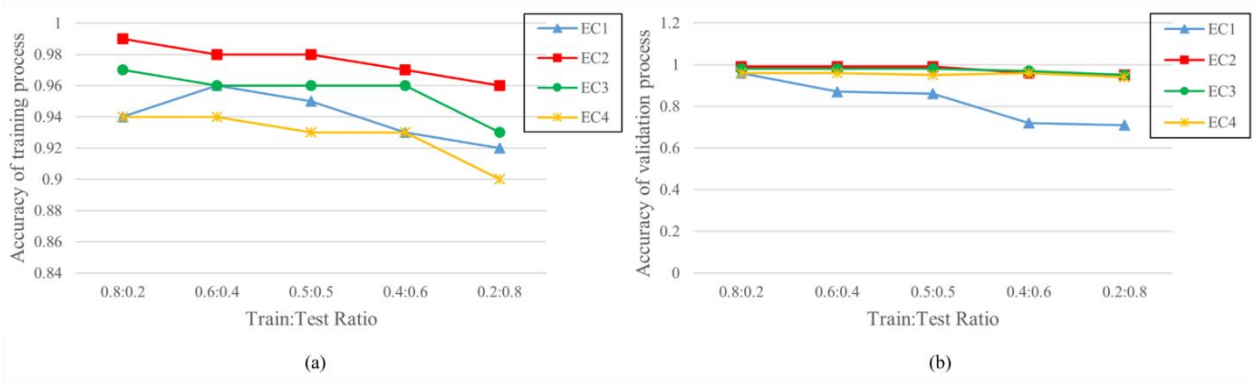

Fig. 4. (a) and (b) are the accuracies of training process and the validation process, respectively over different experimental configurations (ECs) (which are mentioned under methodology of this paper) and varying train-test split ratios.

other hand, when observed with a perspective of the accuracies (i.e., All the accuracies are around 0.8 which is regarded as a significant performance in image classification tasks) it reveals that the image dataset has lower levels of noise.

B. Classification with learning

Classification endorsed by the application of CNNs manifested higher accuracies when compared to the classification performed by the KNN algorithm. Fig. 4 illustrates the variation of accuracy with different train-test split ratios and different experimental configurations.

For each experimental configuration, training accuracy (as shown in Fig. 4-a) is dropped when the training image portion decreases while increasing the number of validation images. In fact, demonstrating the common idea that lesser training in deep learning models causes lesser accuracies. Nevertheless, the size of the drop is negligible as all the accuracies are above 0.9 (or equal to 0.9) in each scenario. The highest overall accuracy is achieved when the training weights are initialized from the pre-trained model (EC2) instead of random initialization (EC1).

TABLE III. RESULTS OF THE GRID SEARCH METHOD PERFORMED TO FIND BEST OPTIMIZER AND LEARNING RATE

Search 1: Learning Rate is Randomly Selected (=0.0001) and Fixed to Test Several Optimizer Types			
Learning Rate	***Optimizer***	***Training accuracy***	***Training time (seconds)***
0.0001	Adam	0.94	742
	Adadelta	0.57	757
	AdamW	0.90	484
	Adamax	0.89	518
	ASGD	0.57	505
	RMSprop	0.93	635
	SGD	0.58	744
Search 2: Best Optimizer (RMSprop) from Search 1 is Fixed and Tested Several Learning Rates			
Optimizer	***Learning rate***	***Training accuracy***	***Training time (seconds)***
RMSprop	0.1	0.55	630
	0.01	0.57	634
	0.001	0.94	641
	0.0001	0.93	637
	0.00001	0.93	642

Specifically, even with 20% training images, the use of the exact feature extractor of the pre-trained model for training (EC3) induced higher accuracy than training from scratch. In the instance where both feature extractor weights and classifier weights (weights of the fully connected layers) of the pre-trained model are used on training, an overall accuracy of 0.9 is achieved.

On the contrary, validation process accuracy (as shown in Fig. 4-b) does not fluctuate considerably over the variation of train-test split ratio regardless of the experimental configurations except where training is done from scratch. All the transfer learning schemes (EC2, EC3, and EC4) show improved validation accuracies when compared to training from scratch (EC1) on the casting surface image dataset.

Table III indicates the possible combinations of the hyperparameters used for the grid search method and related accuracies for EC3 with 20% of training images. During optimization of hyperparameters, first, we picked a random learning rate (0.0001) and performed a grid search with seven optimizer types. The best performance is gained by setting the optimizer to the RMSprop algorithm [39]. Fixing the optimizer as RMSprop algorithm, then we tested several learning rates which resulted in 0.0001 as the optimum value. Overall best hyperparameters (i.e., optimizer type and learning rate) found by the grid search method with the other hyperparameters found from the literature were standardized as shown in Table IV over the final run of each experiment.

TABLE IV. OPTMIZED HYPERPARAMETER SETTINGS/VALUES STANDARIZED THROUGHT ALL EXPERIMENTS

Hyperparameter	**Setting/Value**	**Obtained with Grid Search (GS) Method/Using Literature (LT)**
Optimizer	RMSprop	GS
Learning Rate	0.0001	GS
Learning rate policy	Step (decay over epoch)	LT
Momentum	0.9	LT
Batch Size	16	LT

V. CONCLUSIONS AND FUTURE WORK

Maintaining quality standards is vital in the casting product manufacturing industry for better business

performance and for the safety of the end-users who consume products with critical mechanical components fabricated by casting. Automated inspection of casting defects leads to lesser inspection times and circumvents safety problems of employees working in hazardous environments.

In this paper, we discussed the application of AlexNet CNN architecture-based transfer learning for automated inspection of surface defects of a submersible pump impeller manufactured by casting. Over the last decade, for the task of casting defect recognition, numerous sophisticated architectures were proposed with higher architectural complexity and better performance compared to the AlexNet architecture. Using the results of our study, we show (limited to the dataset used) that a simpler architecture like AlexNet can perform better when it is implemented with transfer learning and optimized model parameters. As future work, methods discussed in this study can be tested over other datasets containing images of casting surface defects of different products.

Over the several experimental configurations tested, the use of the exact feature extractor of the pre-trained model for training demonstrated the best performance in terms of training accuracy and the training time (Although training with weights initialized from the pre-trained model resulted in the overall highest accuracy the training time is higher in contrast to using the entire feature extractor.

Several recommendations for a casting surface defect detection system can be made based on the results of this study. Nevertheless, as future work, the practical usability of such a system needs to be tested prior to implementation as several dataset-specific parameters still need to be adjusted depending on the circumstance. The process of capturing the surface images of the casting products is vital including, but not limited to: (1) adhering to proper lighting conditions, and (2) maintaining unique and plain background when capturing. As shown in the results, transfer learning can be implemented to reduce the training time and enhance the robustness of the model. Moreover, transfer learning is beneficial when the number of training images is lower. Specifically, the use of a feature extractor from the pre-trained model and limiting the training only with the classification layers (fully connected layers) with casting defect data is advantageous instead of using all the parameters of the pre-trained model. Furthermore, fine-tuning the model hyperparameters is crucial for obtaining better results.

REFERENCES

[1] W. Barkman, In-process quality control for manufacturing. CRC Press, 1989.

[2] R. T. Chin and C. A. Harlow, "Automated visual inspection: A survey," IEEE transactions on pattern analysis and machine intelligence, no. 6, pp. 557–573, 1982.

[3] M. Sahoo, Principles of metal casting. McGraw-Hill Education, 2014.

[4] T. V. Sai, T. Vinod, and G. Sowmya, "A critical review on casting types and defects," Engineering and Technology, vol. 3, no. 2, pp. 463–468, 2017.

[5] M. K. Ferguson, A. Ronay, Y.-T. T. Lee, and K. H. Law, "Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning," Smart and sustainable manufacturing systems, vol. 2, 2018.

[6] D. Mery, T. Jaeger, and D. Filbert, "A review of methods for automated recognition of casting defects," INSIGHT-WIGSTON THEN NORTHAMPTON-, vol. 44, no. 7, pp. 428–436, 2002.

[7] S. Gholizadeh, "A review of non-destructive testing methods of composite materials," Procedia Structural Integrity, vol. 1, pp. 50–57, 2016.

[8] Q. Wan, H. Zhao, and C. Zou, "Effect of micro-porosities on fatigue behavior in aluminum die castings by 3D X-ray tomography inspection," ISIJ international, vol. 54, no. 3, pp. 511–515, 2014.

[9] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," arXiv preprint arXiv:1511.08458, 2015.

[10] H. Strecker, "A local feature method for the detection of flaws in automated X-ray inspection of castings," Signal Processing, vol. 5, no. 5, pp. 423–431, 1983, doi: https://doi.org/10.1016/0165-1684(83)90005-1.

[11] Z. Górny, S. Kluska-Nawarecka, D. Wilk-Ko\lodziejczyk, and K. Regulski, "Diagnosis of casting defects using uncertain and incomplete knowledge," Archives of Metallurgy and Materials, vol. 55, no. 3, pp. 827–836, 2010.

[12] M. Ferguson, R. Ak, Y.-T. T. Lee, and K. H. Law, "Automatic localization of casting defects with convolutional neural networks," in 2017 IEEE international conference on big data (big data), 2017, pp. 1726–1735.

[13] L. Alzubaidi et al., "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," Journal of big Data, vol. 8, no. 1, pp. 1–74, 2021.

[14] Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, pp. 1097–1105, 2012.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345–1359, 2009.

[16] Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," Journal of cheminformatics, vol. 9, no. 1, pp. 1–13, 2017.

[17] R. Dabhi, "Casting product image data for quality inspection," Kaggle.com. https://kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product (accessed Jun. 14, 2021).

[18] Gayer, A. Saya, and A. Shiloh, "Automatic recognition of welding defects in real-time radiography," Ndt International, vol. 23, no. 3, pp. 131–136, 1990.

[19] Eckelt, N. Meyendorf, W. Morgner, and U. Richter, "Use of automatic image processing for monitoring of welding processes and weld inspection," in Non-destructive testing, Elsevier, 1989, pp. 37–41.

[20] Filbert, R. Klatte, W. Heinrich, and M. Purschke, "Computer aided inspection of castings," in IEEE-IAS Annual Meeting, 1987, pp. 1087–1095.

[21] Mery, "New approaches for defect recognition with X-ray testing," Insight, vol. 44, no. 10, pp. 614–15, 2002.

[22] X. Li, S. K. Tso, X.-P. Guan, and Q. Huang, "Improving automatic detection of defects in castings by applying wavelet technique," IEEE Transactions on Industrial Electronics, vol. 53, no. 6, pp. 1927–1934, 2006.

[23] Kehoe and G. A. Parker, "An intelligent knowledge based approach for the automated radiographic inspection of castings," NDT & E International, vol. 25, no. 1, pp. 23–36, 1992.

[24] D. Wang, B. Wang, H. Yao, H. Liu, and F. Tombari, "Local image descriptors with statistical losses," in 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 1208–1212. doi: 10.1109/ICIP.2018.8451855.

[25] R. R. Da Silva, M. H. S. Siqueira, L. P. Calôba, and J. M. Rebello, "Radiographics pattern recognition of welding defects using linear classifiers," Insight, vol. 43, no. 10, pp. 669–74, 2001.

[26] Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53–65, 2018.

[27] L. Jiang, Y. Wang, Z. Tang, Y. Miao, and S. Chen, "Casting defect detection in X-ray images using convolutional neural networks and attention-guided data augmentation," Measurement, vol. 170, p. 108736, 2021.

[28] Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," Journal of Big Data, vol. 6, no. 1, pp. 1–48, 2019.

[29] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in plant science, vol. 7, p. 1419, 2016.

[30] Abd Almisreb, N. Jamil, and N. M. Din, "Utilizing AlexNet deep transfer learning for ear recognition," in 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP), 2018, pp. 1–5.
[31] M. Z. Alom et al., "The history began from alexnet: A comprehensive survey on deep learning approaches," arXiv preprint arXiv:1803.01164, 2018.
[32] Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artificial Intelligence Review, vol. 53, no. 8, pp. 5455–5516, 2020.
[33] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in Artificial intelligence and statistics, 2016, pp. 464–472.
[34] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," Neural computation, vol. 29, no. 9, pp. 2352–2449, 2017.
[35] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.
[36] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in ICML, 2016, vol. 2, no. 3, p. 7.
[37] 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.
[38] S. R. Labhsetwar, S. Haridas, R. Panmand, R. Deshpande, P. A. Kolte, and S. Pati, "Performance Analysis of Optimizers for Plant Disease Classification with Convolutional Neural Networks," arXiv preprint arXiv:2011.04056, 2020.
[39] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," Journal of machine learning research, vol. 13, no. 2, 2012.
[40] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in 12th symposium on operating systems design and implementation, 2016, pp. 265–283.
[41] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," the Journal of machine Learning research, vol. 12, pp. 2825–2830, 2011.
[42] Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in neural information processing systems, vol. 32, pp. 8026–8037, 2019.