Asian Review of Mechanical Engineering
ISSN: 2249-6289 (P) Vol. 10 No.1, 2021, pp.26-35

DOI: https://doi.org/10.51983/arme-2021.10.1.2937

Automatic Detection and Characterization of Weld Defects Using CNN Algorithm in Machine Learning

S. Ramesh Krishnan, T. V. Abhishek, Akhil Vinod, Allen George and C. Harikrishnan
Department of Mechanical Engineering, Government Rajiv Gandhi Institute of Technology, Kerala, India
E-mail: rameshkrishnans@rit.ac.in
(Received 9 March 2021; Revised 28 March 2021; Accepted 19 April 2021; Available online 23 April 2021)

***Abstract* - Conventional radio graphic technique uses visual inspection of scanned output for defect detection. This makes the inline testing of products time consuming and hectic. Convolutional Neural Network (CNN) algorithm in machine learning can be used for the automation of defect detection in radiography thereby reducing human intervention and associated delays. By the use of robotics the welding parameters can be adjusted and the issue of welding defects can be resolved. By combining the two, the defect detection process can be modified into a digital manufacturing process. A dataset created from radiography test data is used for training the algorithm and for writing a program to train this dataset which can be used for defect detection and its characterization.**
Keywords: **Radiography, CNN, Machine learning, Digital Manufacturing**

I. INTRODUCTION

The evaluation of engineering materials or structures without impairing their properties is very important in the quality control of products, failure analysis, and prevention of failure of systems in Service etc. This kind of evaluation can be carried out with Non Destructive Test (NDT) methods. However, these techniques generally require considerable operator skill. During mass Production in a manufacturing industry, the defect detection is done by taking radiographic image of the welded product which is analyzed by a radiographer. Also accurate interpretation of test results may be difficult because the results can be subjective. Radiographic Testing (RT), which is based on differential radiation absorption by variation of part thickness or differential material density, is a widely used NDT technique to evaluate the structural continuity of welded components.

The defects are detected by the brightness (darkness) of analog radiographic films. Conventional radiographic technique uses visual inspection of scanned output for defect detection. This makes inline testing of the products in a convey or belt time consuming and hectic. The time off light diffraction (TOFD) technique which is widely used for automatic weld inspection, particularly in the petrochemical industry, performs classification of defects using ultrasound signals. But this technique also is dependent heavily on the knowledge and experience of the operator as in a conventional radiographic method [1].

Several methods have been proposed for weld defect detection with reduced human intervention. Reza *et al.,* used image processing method for welding defect detection [2] which was applied for the detection of surface defects in welded joints. Nacereddine *et al.,* recommended digitized radiographic image processing with threshold methods [3] whereas Madani *et al.,* proposed image processing by using K-Means algorithm to put the similar colors of the image up to a certain distance [4]. The results included the general and basic concepts related to image processing.

Kasban et al., presented an approach using neutral network for defect detection from radiographic images [5]. The approach had two phases: a training phase and a testing phase. The result was capable of recognizing excessive penetration, lack of side-fusion, undercut, longitudinal crack inclusion, tungsten inclusion, localized porosity, burn-through and crater pipe at low degradation cases. Dimitrios and Ioannis proposed multiclass defect detection and classification in weld radiographic images using geometric and texture feature [6]. A set of descriptors corresponding to texture measurements and geometrical features was extracted for each segmented object and given as input to a classifier. The classifier was trained to categorize each of the objects into one of the defect classes or to characterize it as non-defective.

A methodology to detect defects in NDT of materials using Artificial Neural Network (ANN) and signal processing technique was proposed by Sambath *et al.,* This technique was proposed to improve the sensibility of flaw detection and classification of defects in ultrasonic testing [7]. Zapata *et al.,* put forward an ANN with a modified performance function which could be used in an automatic inspection system of welding defects in radiographic images. The ANN was analyzed by modifying the performance function using a γ parameter in its function for different neurons in the input and hidden layer in order to obtain a better performance on the classification stage. The correlation coefficients, confusion matrix and the accuracy of predictions were determined obtaining a value of 80% for the ANN using a modified performance function with a parameter γ=0.6 [8].

In general, the implementation of welding defect assessment by image processing methods is receiving wider interest. To improve the quality and productivity of weld assessment, a

tool that automatically evaluates the weld defect is highly demanded and needs to be developed. The process of defect detection can be automated by taking the radiographic image of the welded product and then by analyzing it using machine learning [9, 10]. The automation of defect detection in radiography using CNN (Convolutional Neural Network) algorithm in machine learning can be implemented to reduce human intervention and associated delays by a great extent. CNN can be used as the tool for image processing. This tool ensures quality assessment, reduces inspection cost and increases competitiveness.

II. CONVOLUTIONAL NEURAL NETWORK

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a ReLU (Rectified Linear Unit) layer and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

A. Convolution Layer

Convolutional layers are the major building blocks used in convolutional neural networks. The input of model is the pixel values of whole image [11]. If we put such a large feature box into the model, there will be a large number of parameters to be trained, so that the quantity of samples is highly demanded. Compared to basic deep learning method, CNN gives a model where weight sharing exists. The weights and bias of the same layer are identical so that a large quantity of parameters can be reduced [12]. The features of a complete image depend on the position and pixel values. CNN keeps the position feature so that the features of adjacent pixels are kept. The deeper the model, the better will be the expressive of the model. Since CNN is sparse on parameter, we can use limited sample to train a deeper model and to get a higher recognition rate.

B. Max Pooling Layer

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. The purpose of this layer is to reduce the parameter of the next layer simultaneously leaving out the useless parameters and to find out the valuable output tothe nextlayer [13].

C. Fully Connected Layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. After the convolution part and pooling part have collected enough features of a picture, fully connect layer is used in order not to lose any details of the parameters [9].

D. Normalization

A batch normalization layer normalizes each input channel across a mini-batch. To speedup training of convolutional neural networks and reduces the sensitivity to network initialization, batch normalization layers are used between convolutional layers and non linearities such as ReLU layers. An approach to the detection of the main types of defects of welded joints using a combination of convolutional neural networks and support vector machine methods was made by Sizyakin *et al.,* They used CNN for primary classification and the support vector machine (SVM) for accurately defining the defect boundaries. A high efficiency of defect detection and its classification in comparison with a pure CNN method was reported by using the proposed method [14].

In CNN, the main concern is detection of defects regardless of their position in the given images. Another important aspect of CNN is to obtain abstract features when input propagates toward the deeper layers. For example in image classification, the edge might be detected in the first layers, the simpler shapes in the second layers and then the higher level features [15].

Some examples of the data image are shown in Figure 1.

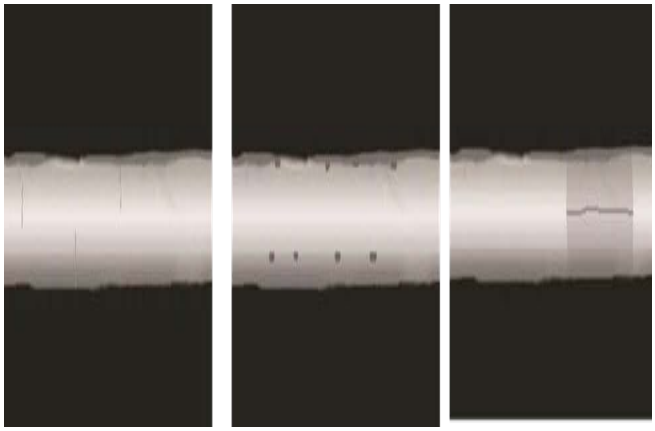

Fig. 1 Original weld images of crack (left), lack of fusion (middle) and lack of penetration (right)

III. SOFTWARE REQUIREMENTS

A. Open CV

Open CV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is a library used for image processing. It is mainly used to do all the operation related to images. Open CV [16] is used for pre-processing and feature extraction fromthe raw image. Image processing is used to make the acquired

image suitable for the model and to standardize all input images. Feature extraction is performed using Open CV and another Python image processing library called Mahotas.

B. Tensor Flow Lite

Tensor flow [17, 18] is an open source library for numerical computation and large-scale machine learning and is used to train the deep learning model in Google Colab cloud platform. The model generated is converted into Tensor flow Lite (tflite) file and downloaded. Tensor flow Lite makes it possible to run Tensor flow model in Android phones and embedded systems with limited resources. Tensor Flow bundles together a slew of machine learning and deep learning models and algorithms and makes them useful by way of a common metaphor.

C. Scikit Learn

The Scikit Learn [19, 20] module is used to train models such as SVM classifiers [21], random forest classifiers, K-Nearest Neighbor (KNN) classifiers etc. Ensemble learning techniques and transfer learning are used to combine all the trained models to improve accuracy of classification.

D. Keras

Keras [22] is a high-level neural network Application Programming Interface (API) written in Python and capable of running on top of Tensor Flow, CNTK [23, 24] or Theano [25]. It is developed with a focus on enabling fast experimentation. It contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

IV. METHODOLOGY

There are basically three steps in creating the required program in machine learning. They are

1. Collecting a large amount of data to create a dataset to train the algorithm. More training data produces a more accurate result. Data here refers to the radiographic image of the welded product.
2. Writing a program to train the dataset which is obtained in the first step, detect defects and characterizing them based on the defect. Various clusters of data are labeled depending on their defects.
3. Testing and troubleshooting of the program. This is done to check whether the program is giving the correct output for a known output. If not, the problem is rectified and tested again. The program is trouble shooted to solve problems which occur during testing.

The data set for training the algorithm is made from picture editing tools which contain five types of weld defects namely crack, lack of fusion, lack of penetration, pores and slag inclusion. A total of 6001 data images are obtained including 501 non-defect data images. Out of them 25% (1501 data images) are used as test dataset.

The digitalized welds x-ray images [26] are 16 bit tiff pictures which have 65536 levels of pixel value which cannot be checked on the personal computer. So the images are transferred to 8 bit pictures which are easy to analyze. First, the maximum and minimum pixel values of a picture are found out and then the pixel value is stretched from 0 to 255. This process is executed on all images and all are transferred to png format. Open CV library is used for image pre-processing and feature extraction. Image pre-processing [27] is used to make the acquired image suitable for the model and to standardize all input images. Background removal is the first step in image pre-processing.

A Python [28, 29] 3 program is developed to remove the background noise from images captured. The raw image is first resized to a standard size of 1772 x 945 pixels which is then converted to a greyscale image. The image is then smoothed using a Gaussian Blur filter of size 244 x119 pixels. The greyscale image is converted to a monochromatic image using Otsu's method [30-32] and any holes in the image are closed using the morphological closing function provided by OpenCV2. The contours of the images are computed using an inbuilt function. Using the image contours, an image mask is created. The background is removed by performing a bitwise-AND between the mask and the monochromatic image and then turning all the other pixels white in the original image.

V. MODEL

INPUT->[[CONV*N->POOL]]*M->[FC]*K

As shown in Figure 2, INPUT is the sample image, CONV is convolution part and POOL is maxpooling. The (N, M, K) of this sample is (1, 2, 2) where N is the number of training images, M is the number of test images and K is the number of features obtained. In the convolution layer, an image indicates the input matrix when it comes to the first convolutional layer, z is a piece of pixels in the image, w is the weights of filter, b is the bias of the filter, a is the output of the first convolution layer [33].

Pooling layer pick out the maximum feature of n*n filter size. When the picture is uploaded, it extracts features from the image. From the collected features it predicts to which class the image to be assigned. Fully connected layer gets all features together and gives a logit score as to whether these features should belong to one class or not.

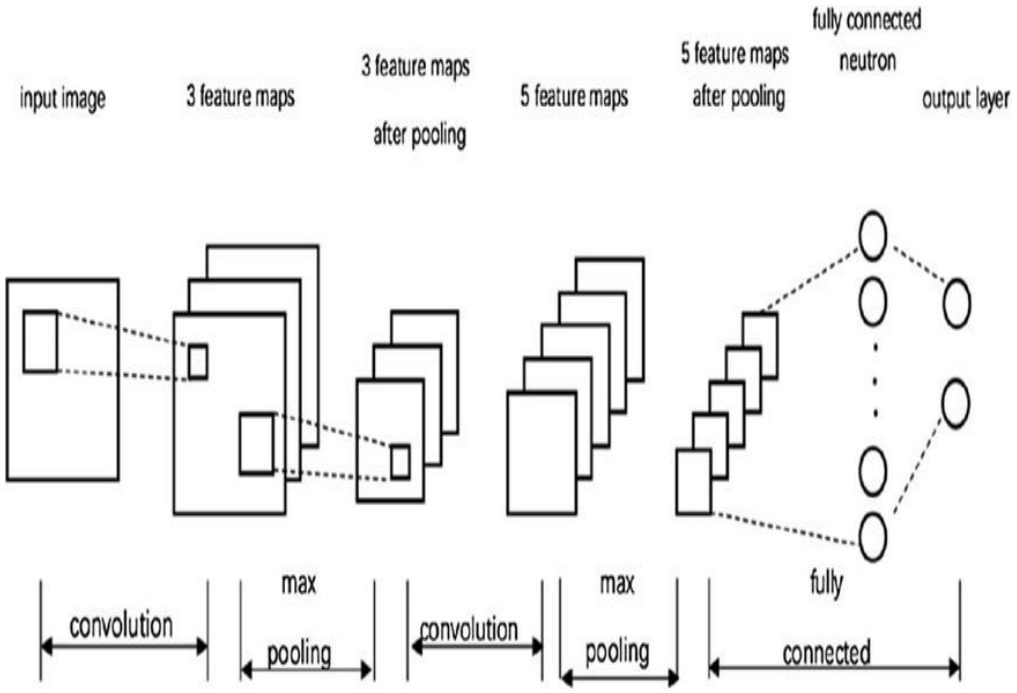

Fig. 2 Model

VI. ALGORITHM

Step 1: Import Necessary Modules

All necessary libraries along with their modules are imported. The program uses different libraries such as NumPy, matplotlib, scikit-learn, keras, tensorflow, random, openCV etc. Different modules from each of the above mentioned libraries are imported for effective working of the program. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays.

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations in Python. Scikit-learn is a machine learning library for Python. It features various algorithms like support vector machine, random forests and k-neighbours and it also supports Python numerical and scientific libraries like NumPy and SciPy. Here it is used for model selection, metrics and preprocessing functions. Many modules such as accuracy_score, precision_score, label encoder etc are imported for this program. Keras is used for manipulation of the model as well as the layers used for training the model.

Modules such as sequential, dense, adam, Conv2D, MaxPooling 2D etc. are implemented in the program. The OS module in Python provides away of using operating system dependent functionality. The functions provided by the OS module allow the interface with the underlying operating system that Python is running on. The random module implements pseudo-random number generators for various distributions including integer and float (real).

Step 2: Preparing the Data

Weld radiography images from the dataset are imported which are converted to binary format with a high threshold range sufficient for better isolation of defects. This is followed by making the functions to get the training and validation set from the images, assigning labels to each defect class, label encoding Y array and the one hot encoding and splitting into training and validation sets.

Step 3: Modeling

ACNN model is created using Keras which has 5 Conv 2D layers, 5 MaxPooling 2D layers, one flatten layer and one dense layer. The conv block corresponds to a 2D convolution layer with batch normalization, maxpooling and dropout layers. The drop outlayer randomly turns off connections of the neural network during training, which significantly reduces over-fitting. The batch normalization layer is used to achieve higher learning rates. In each convblock, the max-pooling layer reduces the size of the input image by half in each dimension. Image data augmentation can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

Training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. From Keras module, a function Image Data Generator is used for this purpose. The values of horizontal and vertical flip attributes are set as ‘True’ so that images get randomly flipped in those directions. Also a

zoom range of 0.2 factor is set for random zooming of images. Since the dataset used have large number of uniform images with homogeneous defect pattern, all other attributes in Image Data Generator are set to default values. The model is compiled using a dam optimizer, with loss attribute set to categorical cross entropy and the metrics attribute set to accuracy. The model summary is printed out for reference using summary () function. For fitting the training set, fit_generator function is used. Batch size is set to 50 and number of epochs is set to 10. The testing set is assigned as the validation set. This step fits the training set onto the model and cross verifies the prediction with the validation set-testing set in this case. Loss, accuracy, validation loss and validation accuracy are calculated in each epoch and is printed out. A constant increase invalidation accuracy is the expected outcome, with the highest value obtained at the last epoch.

Step 4: Evaluating Model Performance

Model performance is evaluated using the output from the previous step. To view the output in a perceivable manner, matplotlib is used to plot graphs with parameters of the previous step. Two Graphs are generated, model accuracy plot and model loss plot. Model accuracy plot is a graph with accuracy on abscissa and epochs on the ordinate where as model loss plot is a graph with loss on abscissa and epochs on the ordinate. The model accuracy graph is expected to increase over the epochs whereas it is vice versa for model loss plot.

Step 5: Visualizing Prediction on Validation Set

In this step, the aim is to visualize some of the predictions carried out on the validation set. For this, two lists - properly_ classified and mis_classified are created. Random images from both these lists are plotted using random and matplotlib modules.

Step 6: Generating Classification Report and Confusion Matrix

Scikit-learn library is used to create classification report and confusion matrix of the prediction on validation set. It is helpful in evaluating the results.

Step 7: Prediction of User Input Image

Accept image from the user and predict its class using the trained model.

VII. PERFORMANCE EVALUATION

For the preparation of data, the dataset is imported and training data is prepared by assigning label to each class. This is done with the help of custom written functions as shown in Figure 3.

A progress bar is created using ‘tqdm’ library for making it easier to follow up the progress of train data generation.

Fig. 3 Data generation

After the data is generated, Label Encoding and One Hot Encoding takes place followed by splitting of data into training and test set. In the third step, creation of model takes place. As stated Conv Net model is used. After building the model, data augmentation is carried out to prevent over fitting. The model is compiled and the summary is printed out as shown in Figure 4.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 119, 224, 32)      2432
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 59, 112, 32)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 59, 112, 64)       18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 29, 56, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 29, 56, 96)        55392
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 28, 96)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 28, 96)        83040
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 14, 96)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 14, 128)        110720
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2688)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1376768
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 3078
=================================================================
Total params: 1,649,926
Trainable params: 1,649,926
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4 Model summary

The model is fitted on the training set and predictions on validation set are done. The loss, accuracy, validation loss and validation accuracy are printed down for each epoch with an included progress bar. The performance of the model can be judged by the final validation accuracy. For this model it shows a very high accuracy of 99.5%.

```
Epoch 1/10
82/82 [==============================] - 30s 365ms/step - loss: 1.1927 - accuracy: 0.4022 - val_loss: 1.0971 - val_accuracy: 0.4160
Epoch 2/10
82/82 [==============================] - 24s 299ms/step - loss: 1.0752 - accuracy: 0.4501 - val_loss: 1.0824 - val_accuracy: 0.4298
Epoch 3/10
82/82 [==============================] - 24s 296ms/step - loss: 0.9610 - accuracy: 0.5367 - val_loss: 0.4418 - val_accuracy: 0.8698
Epoch 4/10
82/82 [==============================] - 24s 294ms/step - loss: 0.2677 - accuracy: 0.9090 - val_loss: 0.1282 - val_accuracy: 0.9629
Epoch 5/10
82/82 [==============================] - 24s 294ms/step - loss: 0.0926 - accuracy: 0.9693 - val_loss: 0.0636 - val_accuracy: 0.9825
Epoch 6/10
82/82 [==============================] - 24s 295ms/step - loss: 0.0645 - accuracy: 0.9791 - val_loss: 0.0707 - val_accuracy: 0.9782
Epoch 7/10
82/82 [==============================] - 24s 291ms/step - loss: 0.0626 - accuracy: 0.9818 - val_loss: 0.0378 - val_accuracy: 0.9927
Epoch 8/10
82/82 [==============================] - 24s 291ms/step - loss: 0.0456 - accuracy: 0.9855 - val_loss: 0.0388 - val_accuracy: 0.9935
Epoch 9/10
82/82 [==============================] - 24s 292ms/step - loss: 0.0342 - accuracy: 0.9907 - val_loss: 0.0246 - val_accuracy: 0.9942
Epoch 10/10
82/82 [==============================] - 24s 294ms/step - loss: 0.0292 - accuracy: 0.9917 - val_loss: 0.0114 - val_accuracy: 0.9956
```

Fig. 5 Result

The data can be further evaluated by converting to graphical plots. Two such plots are generated–the model accuracy plot and the model loss plot as shown in Figure 6 and Figure 7. The model accuracy plot is a plot between accuracy on abscissa and epochs on the ordinate. The model loss plot is a plot between loss on abscissa and epochs on the ordinate. The model accuracy plot and the model loss plot for the given model give plots with expected pattern i.e., with accuracy increasing within creasing epochs and loss decreasing within creasing epochs.

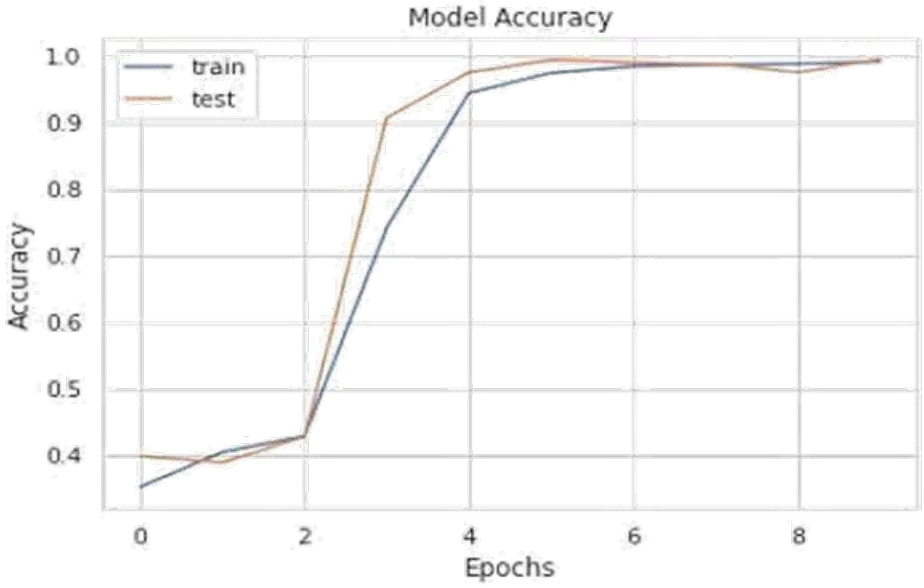

Fig. 6 Model Accuracy Plot

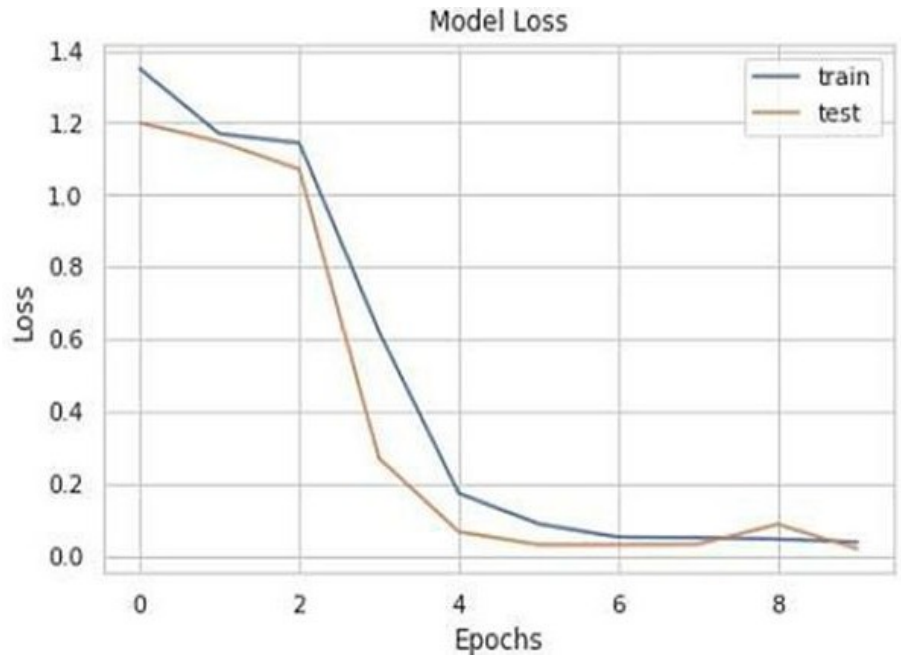

Fig. 7 Model Loss Plot

VIII. RESULTS

Random examples of correct prediction of validation data are shown in Figure 8. Index values used are: 0-cracks, 1-lack of fusion, 2-lack of penetration, 3-no defect, 4-pores, 5-slag inclusion.

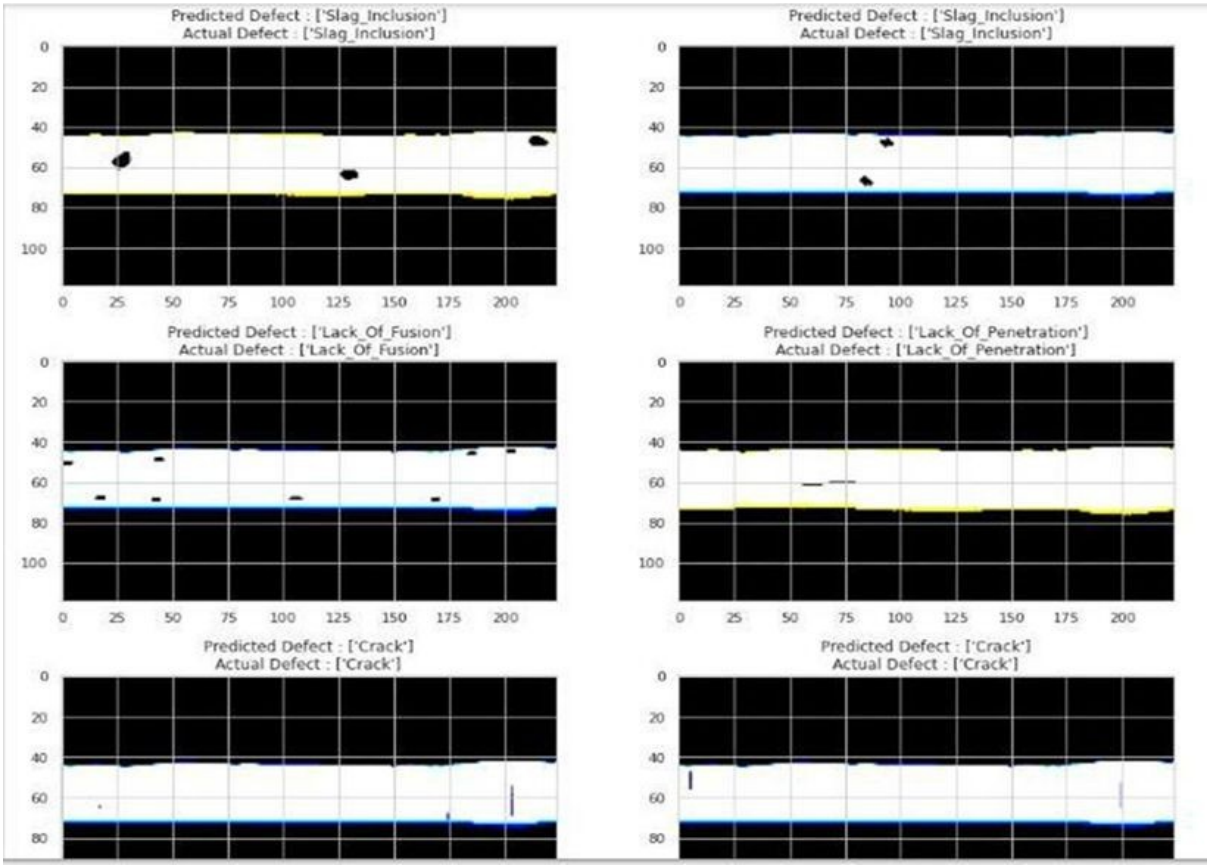

Fig. 8 Correct prediction of validation data

Using classification_ report attribute of scikit-learn library, classification report of the above prediction as in Figure 9 is generated. It depicts accuracy, precision, recall and f-measure for each label. As mentioned earlier, 25% of total data is used as validation data. The 'support' column shows the number of data belonging to the corresponding label in the validation set.

	precision	recall	f1-score	support
0	0.97	0.99	0.98	286
1	1.00	0.99	0.99	285
2	1.00	1.00	1.00	260
3	0.98	1.00	0.99	135
4	1.00	0.99	0.99	271
5	0.99	0.98	0.98	264
accuracy			0.99	1501
macro avg	0.99	0.99	0.99	1501
weighted avg	0.99	0.99	0.99	1501

Fig. 9 Classification report

The confusion matrix for the given prediction is also generate dusing confusion_matrix attribute of scikit-learn library which is shown in Figure 10.

```
[[283   0   0   3   0   0]
 [  0 282   0   0   0   3]
 [  0   0 260   0   0   0]
 [  0   0   0 135   0   0]
 [  4   0   0   0 267   0]
 [  6   0   0   0   0 258]]
```

Fig.10 Confusion matrix

It can be seen from confusion matrix that all 'lack of penetration' and 'no defect images' are correctly classified. Whereas a few images corresponding to 'crack defect' get predicted as 'no defect', a few images with the defect 'lack of fusion' get predicted as 'slag inclusion'. Also a few pores and slag inclusion images got predicted as crack images. When prediction is carried out using custom images not in the training or test dataset, the observed results are found to be mostly in accordance with the experiment results an instance of which is shown in Figure 11.

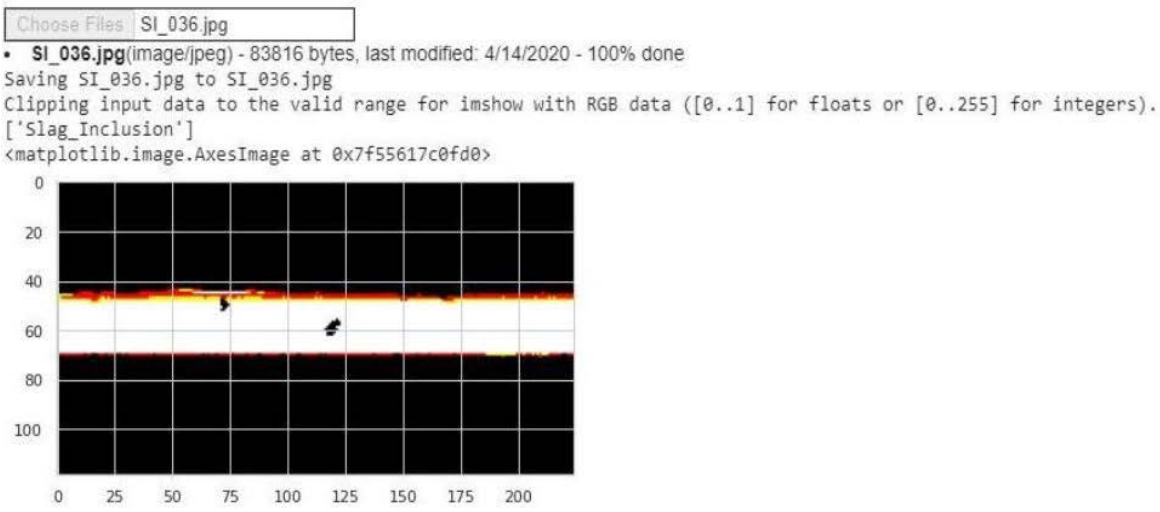

Fig. 11 Correct prediction of a custom image

IX. CONCLUSION

The result of the experiment shows validation accuracy of 99.5%. This is rather a high value for image classification using CNN. The reason for such a high value can be attributed to the uniform format of images in the dataset. Since the images in the dataset are created using picture editing tools, a high order of uniformity can be noted in all of the images. It is noted that even in such a perfect dataset a small error factor of 0.5% persists. If real radiographs were used in place of these images, the accuracy may go down due to their irregularity, noise and low resolution. From the confusion matrix, it can be seen that most are misclassified as 'crack'. This can be due to similarity in pattern of the classes. The classes 'Lack of Penetration' and 'No Defects' are the ones with zero misclassification in the validation set. This may be attributed to the fact that the validation set might have over fitted. Further addition of images to the validation set may provide an alternate result with different accuracy. Likewise, a different set of images invalidation set may show variance in the accuracy.

While prediction is done using new images, accuracy of prediction is found to be less than validation accuracy provided by the program. The increased number of misclassifications while prediction using custom images is accounted by the change in format of the new images or due to the difference in the method used to input the image. The accuracy metric of this experiment using custom images is less than validation accuracy of model due to the above mentioned reasons. The feedback system in a digital manufacturing workspace can be incorporated into this program to detect and characterize any defects present and utilize this information to run a secondary command that adjust parameters so as to rectify the cause of the defect. Robotics can also be implemented to control the welding parameters and thereby controlling defects which are produced by welding. This also makes it possible automatic Segregation of defective products in a conveyor system with no human interference.

REFERENCES

[1] P. Elineudo Moura, R. Romeu Silva, H. S. Marcio Siqueira and João Marcos Rebello, "Pattern Recognition of Weld Defects in Preprocessed TOFD Signals Using Linear Classifiers," *Journal of Nondestructive Evaluation*, Vol. 23, pp. 163-172, 2004.

[2] Reza Nejatpour, Ali Asmari Sadabad, and Ali Akbar Akbari, "Automated weld defects detection using image processing and CAD methods," *ASME International Mechanical Engineering Congress and Exposition.Mechanical Systems and Control*, Vol. 11, 2009.

[3] N. Nacereddine, M. Zelmat, S. S. BelaYfa, and M.Tridi, "Weld defect detection in industrial radiography based digital image processing," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 1, pp. 433-436, 2007.

[4] Saba Madani, and Mortaza Azizi, "Detection of Weld Defects in Radiography Films Using Image Processing," *Cumhuriyet Science Journal*, Vol. 36, pp. 2397-2404, 2015.

[5] H. Kasban, O. Zahran, H. Arafa, M. El-Kordy, S. M. S. Elaraby, and F. E. A. El-Samie, "Welding defect detection from radiographic image using cepstral approach," *Nondestructive Testing and Evaluation*, Vol. 44, No. 2, pp. 226-231, 2011.

[6] Ioannis Valavanis, and Dimitrios Kosmopoulos, "Defect detection and classification in weld radiographic images using geometric and texture features," *Expert Systems with Applications*, Vol. 37, No. 12, pp. 7606-7614, 2010.

[7] S. Sambath, P. Nagaraj, and N. Selvakumar, "Automatic Defect Classification in Ultrasonic NDT Using Artificial Intelligence," *Journal of Nondestructive Evaluation*, Vol. 30, pp. 20-28, 2011.

[8] Juan Zapata, Rafael Vilar, and Ramón Ruiz, "Automatic Inspection System of WeldingRadiographic Images Based on ANN Under a Regularisation Process," *Journal of Nondestructive Evaluation*, Vol. 31, pp. 34-45, 2012.

[9] M. Tom Mitchell, *Machine Learning*, McGraw-Hill Science/ Engineering/Math, 1997.

[10] [Online].Available: https://www.geeksforgeeks.org/machine-learning

[11] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, Vol. 177, pp. 232-243, 2020.

[12] H. Samuel Huang, Peng Liu, Abhiram Mokasdar, and Liang Hou, "Additive manufacturing and its societal impact: a literature review," *The International Journal of Advanced Manufacturing Technology*, Vol. 67, pp. 1191-1203, 2013.

[13] Ciresan Dan, Ueli Meier, Jonathan Masci, M. Luca Gambardella, and Jurgen Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification", *Artificial Intelligence*, Vol. 2, pp. 1237-1242, 2011.

[14] Roman Sizyakin, Viacheslav Voronin, Nikolay Gapon, Aleksandr Zelensky, and Aleksandra Pižurica, *Automatic detection of welding defects using the convolutional neural network*, Automated Visual Inspection and Machine Vision III, 11061, 2019.
[15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *International Conference on Engineering and Technology*, pp. 1-6, 2017.
[16] Adrian Kaehler, and Gary Bradski, "Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library," *Computer Programming/Robotics*, O' Reilly Media, 2016.
[17] Martin Abadi *et al.*, "Tensor Flow: Large-scale machine learning on heterogeneous distributed systems," [Online]. Available: https://www.tensorflow.org., 2015.
[18] [Online]. Available: https://www.tensorflow.org/guide/effective_tf2, 2019.
[19] [Online]. Available: https://scikitlearn.org/0.20/_downloads/scikit-learndocs.pdf.https://scikit-learn.org., 2019.
[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Perrot, and Édouard Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, pp. 2825-2830, 2011.
[21] Cortes Corinna, Vapnik, and N. Vladimir, "Support-vector networks," *Machine Learning*, Vol. 20, No. 3, pp. 273-297, 1995.
[22] [Online]. Available: https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html, 2016.
[23] Chris Basoglu, CNTK_2_7_Release_Notes-Cognitive Tool kit, 2020.
[24] Yuqing Tang, and Sayan Pathak, "Scalable Deep Learning with Microsoft Cognitive Toolkit," [Online]. Available: https://cntk.ai/Tutorials/AAAI18/CNTK%20Tutorial_AAAI_Feb_2018_final.pdf, 2018.
[25] [Online]. Available: https://www.geeksforgeeks.org/theano-in-python.
[26] Luis Lanca, and Augus to Silva, "Digital Radiography Detectors: A Technical Overview," *Digital imaging systems for plain radiography*, *Springer*, pp. 14-17, 2013.
[27] Statistical Parametric Mapping. [Online]. Available: https://www.fil.ion.ucl.ac.uk.
[28] V. John Guttag, *Introduction to Computation and Programming Using Python*, MIT Press, ISBN978-0-262-52962-4, 2016.
[29] M. Scott Shell, *An introduction to Python for scientific computing*, Free Tech Books, 2019.
[30] Jianzhuang Liu, Wenqing Li, and Yupeng Tian, "Automatic thresholding of gray-level pictures using two-dimension Otsu method", *International Conference on Circuits and Systems*, pp. 325-327, 1991.
[31] Zhang, Jun Hu, and Jinglu, "Image segmentation based on 2D Otsu method with histogram analysis," *International Conference on Computer Science and Software Engineering*, Vol. 6, pp. 105-108, 2008.
[32] Huang, and Deng-Yuan, "Optimal multi-level thresholding using a two-stage Otsu optimization approach," *Pattern Recognition Letters*, Vol. 30, No. 3, pp. 275-284, 2009.
[33] Geoffrey Hinton, "Some demonstrations of the effects of structural descriptions in mental imagery," *Cognitive Science*, Vol. 3, No. 3, pp. 231-250, 1979.