Deep Convolutional Neural Network Algorithm for Prediction of the Mechanical Properties of Friction Stir Welded Copper Joints from its Microstructures

Akshansh Mishra[a]*, Asmita Suman[b]

[a]Centre for Artificial Intelligent Manufacturing Systems, Stir Research Technologies, India

[b]Department of Finance, IIM Amritsar, India

[a]Email id: akshansh@stirresearchtech.in

[a]Orcid id: https://orcid.org/0000-0003-4939-359X

[b]Orcid id: https://orcid.org/0000-0002-8464-8108

Abstract: Convolutional Neural Network (CNN) is a special type of Artificial Neural Network which takes input in the form of an image. Like Artificial Neural Network they consist of weights that are estimated during training, neurons (activation functions), and an objective (loss function). CNN is finding various applications in image recognition, semantic segmentation, object detection, and localization. The present work deals with the prediction of the welding efficiency of the Friction Stir Welded joints on the basis of microstructure images by carrying out training on 3000 microstructure images and further testing on 300 microstructure images. The obtained results showed an accuracy of 80 % on the validation dataset.

Keywords: Machine Learning; Welding Efficiency; Friction Stir Welding; Convolutional Neural Network

1. Introduction

Images have become ubiquitous in all the fields which basically means that vast amount of information can be extracted from imagery. While image classification has now become prevalent in fields like computer vision, self−driving cars, robotics, etc., it is fairly new in the field of microstructures [1−3].

Although the above−mentioned applications differ by various factors, yet they share the common process of correctly annotating an image with one or a probability of labels that correlates to a series of classes or categories. This is known as image classification. The process of identifying the type of microstructure in diverse tasks linked to image−based scene perspectives has taken advantage of the combination of machine learning techniques applied to the development of neural networks.

Deep learning is basically a class of machine learning algorithms that uses multiple layers to progressively extract higher−level features from the raw input. It allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [4−5]. It discovers intricate structure in large data sets by using the back propagation algorithm to indicate how machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. These methods are extracted by neural networks. Among the deep learning models, CNNs (Convolutional Neural Networks) and their modification (FCNNs) have been shown particularly successful used for image classification and segmentation [6−7].

There is a limited number of studies on the application of Convolutional Neural Networks in the Friction Stir Welding Process. Hartl et al. [8] used Convolutional Neural Network modeling for the process monitoring in the Friction Stir Welding Process. Convolutional Neural Network DenseNet−121 was used for automated visual inspection in Friction Stir Welding process [9].

The present study focuses on the application of the Deep Convolutional Neural Network algorithm for predicting the welding efficiency of Friction Stir Welded Copper joints on the basis of microstructure images.

2. Experimental Procedure

CNN models is one of the oldest deep neural networks hierarchy that have hidden layers among the general layers to help the weights of the system learn more about the features found in the input image. The basic architecture of a CNN consists of a convolutional layer as shown in Figure 1 that separates and identifies the various features of the image for analysis in a process called as feature extraction and a fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

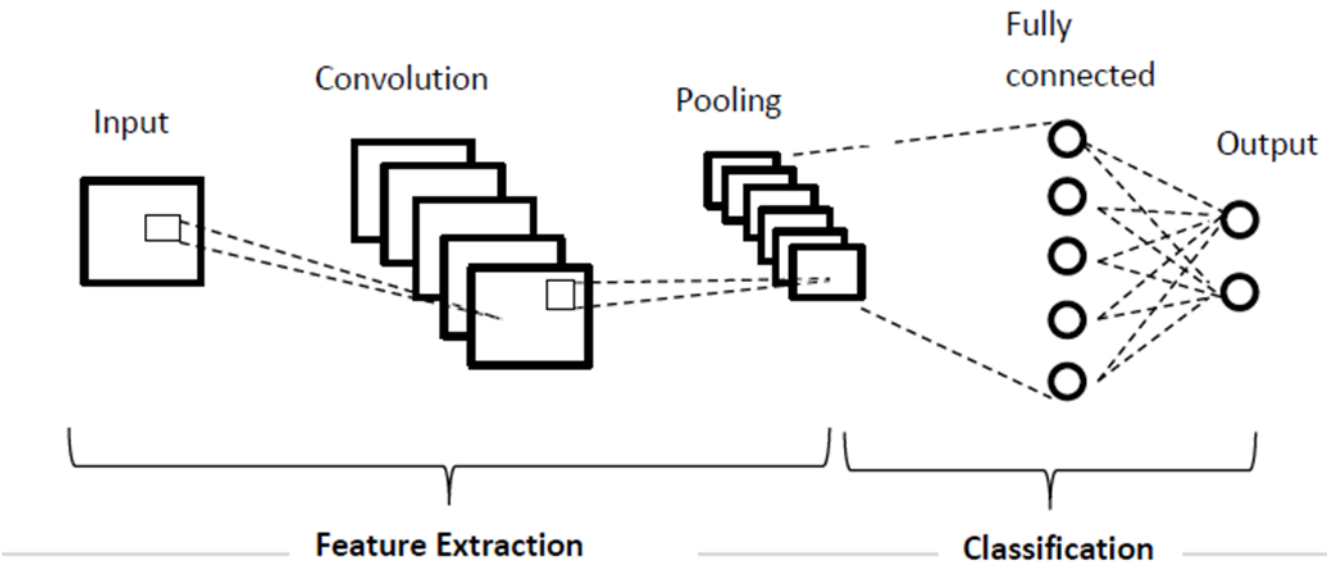

Figure 1: The Architecture of Convolutional Neural Network

The image is composed of various pixels which possess a different numerical value to yield the density within the RGB color space spectrum. The key idea is that there will be the existence of a relationship between which will further act as different features. It should be noted that the spatial arrangement of those obtained features will have no impact on the considered model. So, it can be concluded that there is not any relationship between these individual features. The Convolutional Neural Network model has variables in the form of pixels that have a natural topology. There should be a translation invariance too so that the orientation and size of the given object in an image do not affect the working of the Convolutional Neural Network architecture.

Kernels are used to capture the relationship between the different features i.e. different pixels composing the image. Kernels are considered as a grid of weights that are overlaid on the particular portion of the given image centered around a single pixel. Once the kernel is overlaid, each weight from the kernel is multiplied by the given pixel which is some number. As shown in Equation 1, the output of the central pixel is the sum of all those multiplications between the kernel and its respective pixel.

$$Outputoverthecenteredpixel = \sum_{p=1}^{p} W_p.pixel_p \quad (1)$$

There are three types of layers that make up CNN:

1. Convolutional Layer: This layer is the first layer which is used to extract various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM which is slided over the input image and the dot product is taken between the filter and the parts of the input image with respect to the size of the filter MxM. The output generated is known as the feature map which gives us the information about the image. This feature map is then fed to other layers.
2. Pooling Layer: A convolutional layer is followed by a pooling layer which is basically used to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operating on each feature map. There are several pooling techniques like max pooling (largest element is taken from the feature map), average pooling (average of elements is taken) etc.
3. Fully Connected Layer: It consists of weights and biases along with the neurons and is used to connect the neurons between two different layers.
4. Activation function: It usually follows a fully connected layer and introduces a non− linearity into the network. There are several commonly used activation function such as ReLu, Softmax, tanH and the Sigmoid function.
In Computer Vision, datasets are divided into two main categories: a training dataset used for training the algorithm learnt to perform the desired task and a testing dataset that algorithm is tested on. The dataset used in present study consists of two classes i.e. the microstructure images which have welding efficiency less than 80 percent and the microstructure images which have welding efficiency greater than and equal to 80 percent. The microstructure images for both classes are shown in Figure 2 a) and Figure 2 b).

Figure 2: a) Microstructure image of Friction Stir Welded Copper joints have welding efficiency less than 80 % b) Microstructure image of Friction Stir Welded Copper joints have welding efficiency equal to and above than 80%

3. Results and Discussion

The basic architecture of the model used in present study is shown in Figure 3. Firstly, it should be noted that the convolution operation is performed between two tensors in the Neural

Networks. In this operation, two tensors are taken into account as inputs which result in output as a tensor. The convolution operation is denoted by the "$*$" operator. The Equation 2 is used to carry out the convolution operation.

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix} * \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} = \sum_{i=1}^{9} x_i a_i \tag{2}$$

The input microstructure image can be considered as X and the filter can be denoted by f. So the convolution operation can be obtained by Equation 3.

$$Z = X * f \tag{3}$$

The convolution operation is shown in Figure 4.

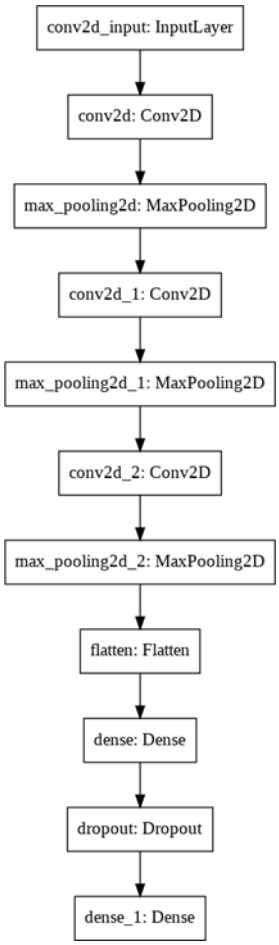

Figure 3: Architecture of CNN model in present study

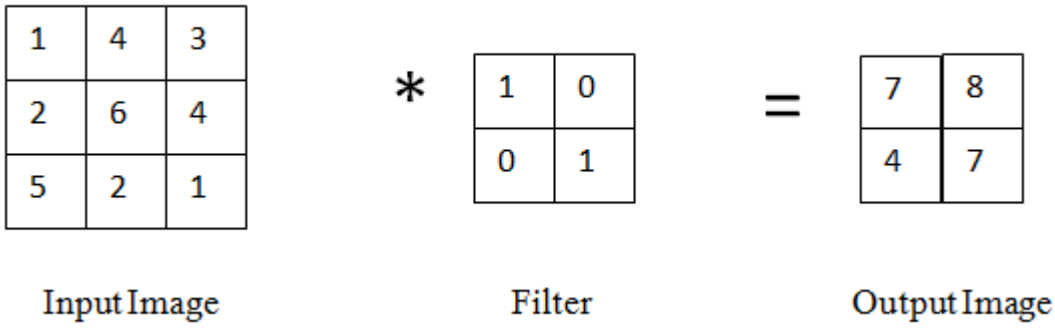

Figure 4: Representation of the basic convolution operation

The mathematical operations are as follows:
$1 \times 1 + 4 \times 0 + 2 \times 0 + 6 \times 1 = 1 + 0 + 0 + 6 = 7$
$4 \times 1 + 3 \times 0 + 6 \times 0 + 4 \times 1 = 4 + 0 + 0 + 4 = 8$
$2 \times 1 + 6 \times 0 + 5 \times 0 + 2 \times 1 = 2 + 0 + 0 + 2 = 4$
$6 \times 1 + 4 \times 0 + 2 \times 0 + 1 \times 1 = 6 + 0 + 0 + 1 = 7$
If the dimension of an input image is $n \times n$ and the dimension of applied filter is $f \times f$ then the dimension of an output image is given by $(n - f + 1) \times (n - f + 1)$.
The features which are extracted from the data by a given convolution layer are sent to the fully connected layer which generates the final output. Generally, the fully connected layer in a Convolutional Neural Network represents a traditional Neural Network.
The convolution layer results an output in the form of two−dimensional matrix but it should be noted that the fully connected layer can only operate with one−dimensional data. So, the value generated by applying Equation 3 is firstly converted into one−dimensional format as shown in Figure 5.

Figure 5: Conversion of two−dimensional matrix to one−dimensional matrix

After the conversion into one−dimensional array, the values are forwarded to the fully connected layer. The individual values obtained are treated as a separate feature representing an image. The incoming data is further subjected to two operations i.e. a non−linear transformation and linear transformation by a fully connected layers. Firstly, the data is subjected to linear transformation as shown in Equation 4.

$$Z_1 = W^T.X + b \tag{4}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} \tag{5}$$

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} \tag{6}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{7}$$

Where, b is the bias, W is the weight and an input image is represented by X. It should be noted that weight is the matrix of randomly initialized numbers. Putting the values of Equation 5, 6 and 7 in Equation 4, the following Equation is obtained:

$$Z_1 = \begin{bmatrix} W_{11} & W_{21} & W_{31} \\ W_{12} & W_{22} & W_{32} \end{bmatrix} \begin{matrix} W_{41} \\ W_{42} \end{matrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{8}$$

Now, in order to capture the complex relationship, non−linear transformation architecture in the form of an activation function is incorporated. In the present work, the sigmoid activation function has been used which is represented by Equation 9.

$$f(x) = \frac{1}{1+e^{-x}} \tag{9}$$

Deep networks need large amount of training data to achieve good performance. In order to achieve a large dataset, image augmentation was required. Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc. This can be easily done with the help of ImageDataGenerator API in Keras.

Both the classes of the dataset were firstly divided into train and test. The datasets were then augmented to increase the dataset size. The following base code was used to generate images by varying the various parameters:

```
datagen = ImageDataGenerator(rotation_range = 10, width_shift_range=0.1,
                             height_shift_range=0.1,
                             shear_range=0.0,
                             zoom_range=0.1,
                              brightness_range= [0.1,0.1],
                             vertical_flip=True,
                              horizontal_flip=False,
                             fill_mode='constant',
                             cval=0)
```

Following the above procedure, around 3000 images for train and 300 images for test were generated. Further, for training the model, CNN algorithm was used and Keras library was used for writing the code for the same.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Dropout,Flatten
model = Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),input_shape= image_shape, activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape= image_shape, activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape= image_shape, activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer= 'adam',
              metrics=['accuracy'])
```

In python programming, the model type that is most commonly used is Sequential type. It is the easiest way to build a CNN model and permits us to build a model layer by layer. The add () function is used to add layers to the model. In the above model, there are three convolution and pooling pairs followed by a flatten layer which is usually used as a connection between the convolution and dense layer. Further, two dense layer was added with a dropout layer. Dense layer is the regular deeply connected neural network layer. A dropout layer is basically used for regularization and reduces over fitting. The optimizer used was adam.
The batch size was chosen to be 16 and the model was trained with class mode as binary as there are two classes to be classified. The number of epoch taken was 50 for training purpose. One epoch refers to one cycle through the full training dataset. The model was then evaluated using the test dataset. The classification report is used for determining the accuracy of the model. The model was further checked by taking an image from test dataset and predicting its class label with the help of following code:

```
pred = model.predict_generator(test_image_gen)
predictions = pred > 0.5
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(test_image_gen.classes,predictions))
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/train
  warnings.warn('`Model.predict_generator` is deprecated and '
              precision    recall  f1-score   support

           0       0.17      0.03      0.05        64
           1       0.82      0.97      0.89       288

    accuracy                           0.80       352
   macro avg       0.49      0.50      0.47       352
weighted avg       0.70      0.80      0.73       352
```

The basic architecture for the model is shown in Figure 3. In the model, there were three convolution layer followed by pooling layer.

Convolutional layers in a CNN systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input. A pooling layer is added after a convolutional layer to apply nonlinearity to the feature maps output by the convolutional layer. The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps. This involves selecting a pooling operation. The size of the pooling operation or filter is smaller than the size of feature map (we have used 2x2 pixels with a stride of 2 pixels). This means pooling layer will always reduce the size of each feature map. The pooling operation used was maximum pooling which basically takes the maximum value for each patch of feature map.

Further, a flatten layer was added which is usually used as a connection between the convolution and dense layer. Flattening is basically converting the data into 1−d array for inputting it to next layer. The output of convolutional layer is flattened to create a single long feature vector which is then connected to the final classification model, which is called a fully− connected layer. A dense layer was added followed by a dropout layer and another dense layer. A dense layer is connected deeply which means each neuron in the dense layer receives input from all neurons of its previous layer. It basically performs a matrix−vector multiplication and the values used in the matrix are actually parameters that can be trained and updated with the help of back propagation. The activation function used for this layer was 'relu' which is helpful in applying the element wise activation function. The dropout layer used was basically for reducing the over fitting.

Figure 6 shows the plot of loss function with respect to the number of epochs.

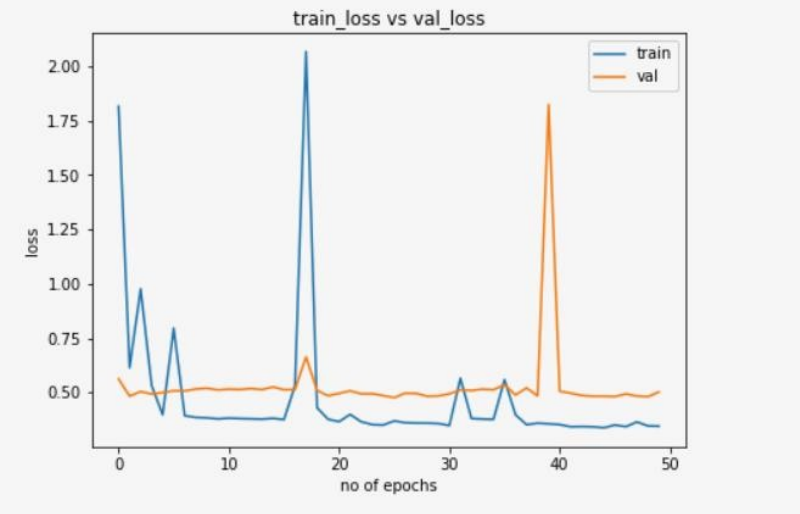

Figure 6: Plot of loss function with respect to the number of epochs

Figure 7 shows the plot of accuracy with the number of epochs.

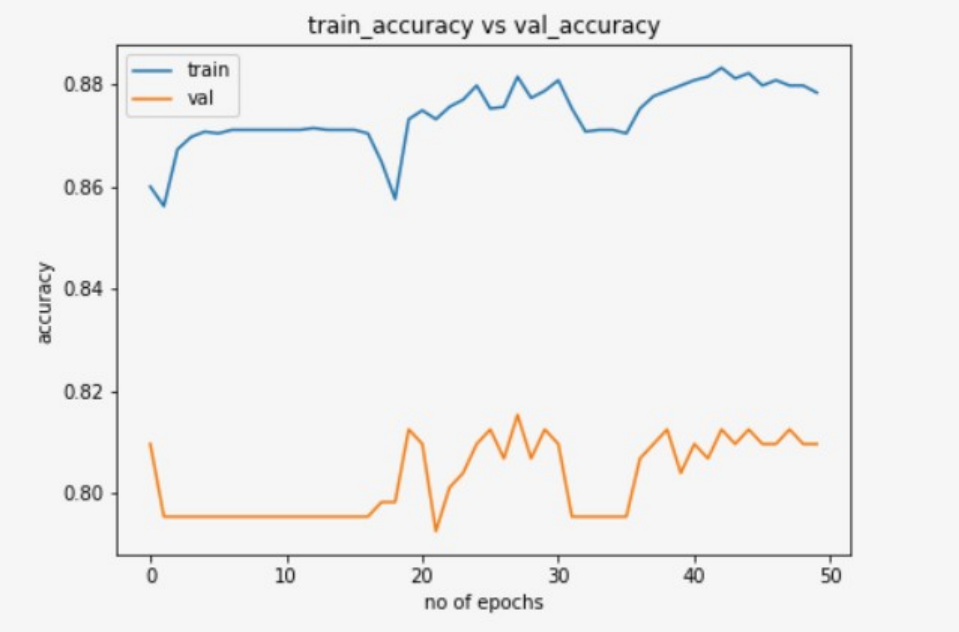

Figure 7: Plot of accuracy with respect to the number of epochs

It is observed from the Figure 7 that the accuracy increases for both training and testing (validation) dataset.

4. Conclusion

Convolution Neural Network is a powerful tool for image recognition and it has a great capability of complex problem−solving. While training the Artificial Neural Network models, few transformations should be kept in mind. There is a final layer with a sigmoid activation function and a single node for binary classification problems. The present research work basically focuses upon building a basic CNN model for the classification of the two classes of

microstructures. The model was made with an accuracy of 80% which can be improved. Further improvement in the model can be made by:
• Using more images for the dataset
• Adjusting the learning rate
• Adjusting the batch size which will allow the model to recognize the patterns better. If the batch size is low, the patterns will repeat less and hence convergence will be difficult whereas if the batch size is high, learning will be slow.
• Adjusting the number of epochs
The future scope of the work is to use the Convolutional Neural Network (CNN) algorithm for detection of defects present in Friction Stir Welded joints.

References

[1]− Petrou, M.M. and Kamata, S.I., 2021. Image processing: dealing with texture. John Wiley & Sons.
[2]− Aaron, J. and Chew, T.L., 2021. A guide to accurate reporting in digital image processing–can anyone reproduce your quantitative analysis?. Journal of Cell Science, 134(6), p.jcs254151.
[3]− Monga, V., Li, Y. and Eldar, Y.C., 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. IEEE Signal Processing Magazine, 38(2), pp.18−44.
[4]− Saxe, A., Nelli, S. and Summerfield, C., 2021. If deep learning is the answer, what is the question?. Nature Reviews Neuroscience, 22(1), pp.55−67.
[5]− Janiesch, C., Zschech, P. and Heinrich, K., 2021. Machine learning and deep learning. Electronic Markets, pp.1−11.
[6]− Wang, Z., Li, C. and Wang, X., 2021. Convolutional neural network pruning with structural redundancy reduction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14913−14922).
[7]− Chen, W., Jiang, M., Zhang, W.G. and Chen, Z., 2021. A novel graph convolutional feature based convolutional neural network for stock trend prediction. Information Sciences, 556, pp.67−94.
[8]− Hartl, R.; Bachmann, A.; Habedank, J.B.; Semm, T.; Zaeh, M.F. Process Monitoring in Friction Stir Welding Using Convolutional Neural Networks. Metals 2021, 11, 535. https://doi.org/10.3390/met11040535
[9]− R. Hartl, J. Landgraf, J. Spahl, A. Bachmann, and M. F. Zaeh "Automated visual inspection of friction stir welds: a deep learning approach", Proc. SPIE 11059, Multimodal Sensing: Technologies and Applications, 1105909 (21 June 2019); https://doi.org/10.1117/12.2525947