Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Casting Defect Detection and Classification using CNN Algorithm

[a], Sambhram Padmashali[b], Sanath Poojary[c]
a Bachelor of Engineering, CSE Department, NMAMIT, Nitte, Udupi and 574110, India
b, c Bachelor of Engineering, CSE Department, NMAMIT, Nitte, Udupi and 574110, India

Abstract

This study aims to develop a neural network to categorize the submersible pump impeller casts as defective or not. Nearly 7348 images of the casts provided by Pilot Technocast cast manufacturer are used. The model used here is Convolutional Neural Network (CNN). The activation function used is Relu. A two layered convolutional network is used with Conv2d and Maxpooling2d layers in it. This model is trained using the above dataset and is then used to classify new images with an accuracy of roughly 94%. This work also proposes a defect classification method which can identify the defect type.

Keywords: Convolutional Neural Network, Image Dataset, Training, Evaluating, Image, Libraries;

1. Introduction

In the manufacturing process known as casting, a mould containing hollow chamber is filled with a liquid substance which will then get solidified to a desired shape. Since thousands of finished items are manufactured using these moulds, any flaw in even one of them could result in inappropriate product manufacturing. The imperfections that can occasionally be present in these casts range from small blowholes to metallurgical flaws. Human supervision can be used to inspect these casts, but it takes time and is not always reliable. This model is used to examine the casts for physical abnormalities. To increase classification accuracy, the model will be trained on photos that are evenly lit and of a similar size. It can distinguish between casts that are defective and those that are not, as well as between other kinds of defects, such as heat cracks and fins (outward protrusions).

2. Related Work:

There has been research work done to identify the defects in casting materials by using various neural network models.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

This work proposes the use of You Only Look Once version 3(YOLOv3 to reduce the detection speed, increase the detection efficiency and to replace the manual inspection methods being used currently. This method increases the chance of finding defects especially the minute ones which cannot be recognized by the naked eye. In order to produce standard defect samples, the guided filtering approach is first utilised to increase the flaws in the industrial digital radiography images (DR images). A defect detection data set for network training is created after the defect samples have been annotated. In this paper, we use the improved YOLOv3 network model structure to find defects. [1]

Casting X-ray scans are frequently utilised in production to ensure product quality. This article examines the faulty detection using X-ray images. First, a novel full-image method to distinguish between defective and non-defective castings is suggested. Second, a novel spatial attention bilinear Convolutional Neural Networks (CNN) is suggested to enhance the representational capability of CNN by fusing two techniques, the spatial attention method and bilinear pooling used in deep convolutional neural networks. [2]

Large, annotated dataset acquisition for manufacturing applications continues to be a difficult task due to the time and expense associated with their gathering. In a novel system for data augmentation that solves this flaw, synthetic images are created using Generative Adversarial Networks (GANs) (GANs). The generator produces realistic fakes by gradually training random noise to produce new surface fault pictures. Using these synthetic images, classification systems can be further trained. [3]

Images are frequently analysed as part of Non-Destructive Testing (NDT) in order to find (rare) flaws. This paper suggests a technique that makes use of convolutional neural networks for finding and categorising anomalies (CNNs). A specific issue is that it is frequently difficult to obtain a lot of samples of photos of flaws, which makes training a classifier difficult. To solve this issue, numerous synthetic images are created by fusing genuine flaws with various backdrops A U-Net-style network is trained using these images to perform pixel-level fault identification. Finding fault zones at both pixel level and picture level can be done by these models [4]

Traditionally, conventional methods are ineffective when the detection targets are tiny, localized, and subtle in the complex scenario. But what inhibits deep neural networks like the Convolutional Neural Network (CNN) from surpassing their competition is a large volume of data with precise labels. To address these issues, it is first proposed to use a potent CNN model which can only be trained using image-level labels to find minute

casting errors in a challenging industrial setting. Furthermore, bilinear pooling is employed to improve the model's capability to identify local defects in contrast casting. Finally, a unique training technique is provided that, during the training phase, can help the model develop a new object-level attention mechanism. [5]

In order to automatically identify casting faults in radiography pictures, a novel method based on the Grayscale Arranging Pairs (GAP) characteristic is proposed in this work. First, a model is created utilising pixel pairs from previously obtained images with a steady intensity relationship. Second, by statistically comparing the intensity-difference signs between the input image and the model, flaws can be retrieved. On cast radioscopic pictures with flaws, the suggested method's resistance to noise and light changes has been confirmed. [6]

The enhanced metal surface flaw detection technique known as You Only Look Once (YOLO) is used here. To construct a new scale feature layer based on the YOLOv3 network structure, the shallow features of the Darknet-53's 11th layer are combined with the deep qualities of the neural network. It tries to infer larger traits from smaller defects. When analysing the anchor box's size data, K-Means++ is utilised to lessen sensitivity towards the initial cluster centre. [7]

In this work, a method based on convolutional neural networks for automatic detection of workpiece surface faults is proposed. The foundation of this method is to extract features using symmetric modules in Convolutional Network classification model (SCN). Utilizing three convolution branches with FPN structure, the features are then detected. An enhanced IOU (XIoU), which is used to train detection models, defines the loss function. [8]

This work presents a strategy combining upgraded Faster Region Convolutional Neural Networks (Faster R-CNN) and improved ResNet50 in order to reduce average running time and increase accuracy. In order to classify the sample as having defects or not, the improved ResNet50 model first inputs the image before adding the Deformable Revolution Network (DCN) and better cut-off. When the probability of a flaw is below 0.3, the algorithm produces a sample without any errors. A modified faster R-CNN that incorporates matrix NMS, enhanced feature pyramid networks, and spatial pyramid pooling is used in the absence of this. The output's ultimate findings are the location and type of the sample defect, if one exists. By looking at the data set gathered in the actual production environment, it is possible to raise the accuracy of this procedure to 98.2%. However, the total computation time is faster than other models. [9]

The use of an automated method for assessing casting products that uses convolutional neural networks as support allows for the automatic detection of a variety of flaws, including blow holes, chipping, cracks, and washes In order to achieve high accuracy in the inspection system, it is necessary to apply multiple labelling in accordance with the sequence of the sub-pictures and the presence of flaws in addition to sub-partitioning the original images. Experimental testing of the suggested inspection algorithm's performance with 400 casting products shows that it performs with significantly high accuracy greater than 98%. [10]

The goal of this work is to create a CNN model that can accurately anticipate porosity issues in images taken with a light optical microscope. Images of refined samples of various aluminium alloys with numerous defects were used to train the model. Several different kinds of porosity flaws were present. In the test set, the suggested custom CNN structure did a fantastic job of classifying 3,990 photos correctly while only making 254 errors. Consequently, 94% classification accuracy was attained. [11]

In this work they have examined and contrasted different pre-trained and custom-built architectures for the detection of defective casting goods utilising model size, performance, and CPU latency. The findings demonstrate that compared to pretrained mobile architectures, bespoke architectures are more effective. Additionally, custom models outperform lightweight models like MobileNetV2 and Neural Architecture Search Network (NASNet) by a factor of 6 to 9. The bespoke architectures include substantially less training parameters and smaller models than the top-performing models, such as MobileNetV2 and NASNet (386 times and 119 times, respectively). [12]

This work proposes a deep learning-based approach to detect DR photo defects. A casting fault DR image dataset made up of 18 311 DR pictures labelled for inclusions and defects is created in order to train and test the deep learning model. The method is based on the YOLOv3 EfficientNet object recognition method baseline, which employs EfficientNet as the foundation instead of the YOLOv3 darknet53. [13]

The Intelligent Production learning centre is converted into an I4.0 production system using a revolutionary framework that is presented in this study. The Visual Geometry Group (VGG-16) with CNN model serves as its foundation. Here, the difficulty of detecting tiny faults during an inspection is covered. The aim is to locate the pixel value connected to a defect with the least amount of false-positive results. For the majority of post-production product quality assurance procedures, destructive vs. non-destructive testing and

classification approaches are employed. This uses the Convolutional Neural Networks (CNN) model. This work investigates sophisticated Transfer Learning (TL) algorithms that automatically identify and categorise manufacturing faults in industrial product samples. [14]

This research proposes a YOLOV3-based online technique for surface fault detection. First, to increase network performance, the lightweight network MobileNetV2 is used to extract features. In order to accomplish high-precision online inspection, this research proposes the use of Improved MobileNet-YOLOV3, an end-to-end detection approach based on YOLOV3 (IMN-YOLOV3). [15]

3. System Design:

Data-set can be defined as a group of facts that are typically in a columnar format. Each of these columns corresponds to a different factor. One of the members of the dataset is represented by each row. For each parameter, including an item's height and weight, values are provided. Any fact can be referred to as data. The dataset used here is an image dataset which consists of 7348 images. 6633 images of 300X300 pixel size are used for training. There are two categories:

1. Defective

2. Non-defective

Another dataset is manually created by using the images from ‘Defective’ category with two categories i.e. the defect types.

1.Fins: A fin is any extra or undesirable material that is affixed to a cast. At the parting faces, this thin layer of metal often develops.

2.Heat cracks: The development of shrinkage cracks during the solidification of weld metal are known as heat cracks.

Figure 1 below illustrates the Algorithm's Block Diagram.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

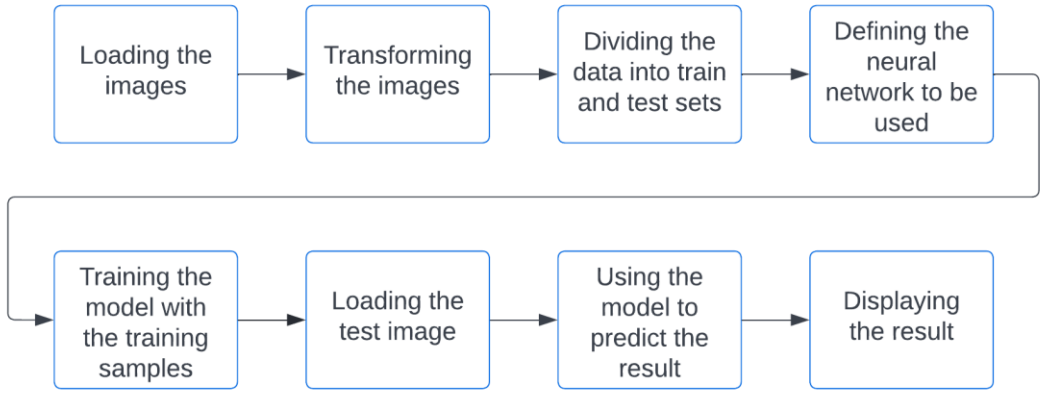

Figure 1: Block Diagram of the Algorithm

3.1 System Design Algorithm:

Step 1: Loading the images.

Step 2: Transforming the images.

Step 3: Dividing the images into train and test data sets.

Step 4: Defining the neural network to be used.

Step 5: Training the model with the training samples.

Step 6: Loading the test image.

Step 7: Using the model to predict the result.

Step 8: Displaying the result.

4. System Implementation:

4.1 Software Requirements:

The following packages should be present in the local machine to implement this project.

1. Keras: It is an open source Library for python for building and evaluating deep learning models that is effective as well as easy to use. It is possible to construct and train neural network models using only a few lines of code thanks to the coverage of Theano and

TensorFlow, two effective frameworks for numerical computing.

2. Scikit-learn: It is a Python toolkit that is open-source that implements several machine learning, pre-processing, cross-validation, and visualisation algorithms using a standardised user interface.

3. Matplotlib: Matplotlib is a Python library which is used to visualize data and plot graphs. It can run on any platform.

5. OpenCV (computer vision): It is a useful platform for computer vision and image processing operations. It is an open-source library that may be used for a variety of tasks, including facial recognition software, detection and tracking, landmark detection, and many others.

4.2 Convolutional Neural Networks algorithm:

Artificial Neural Networks (ANN) are modelled after the biological neural networks seen in animal brains. The mainstay of ANN's strategy is the use of weights and an activation function. The most accurate approach to explain how ANN functions is to say that it simulates the neural network of the brain. In the same way that a human would, it returns and "changes" its thinking after making a mistake.

An ANN's "layers" are rows of data points held by neurons that are all connected by the same neural network. Weights are an essential factor in an ANN. They are modified after each iteration of the neuron in an ANN. A "cost function" used by ANN to measure accuracy determines how the weights from the past are adjusted. An activation function and weights are used for the method.

In contrast, CNN lacks weights and neurons. Instead, CNN applies numerous filters and casting layers on images to analyse visual details. The mathematical layer, the rectified linear unit layer, and the fully linked layer are the three layers present in CNN model. These layers' functions include processing data output and producing an n-dimensional vector to comprehend patterns that the network can "see."

This n-dimensional output allows to identify distinctive features and link them to the supplied image input. The final result of the classification can be obtained later. Even though there are variations, both approaches produce epochs to evaluate the efficacy of the models created and use metrics of the error to improve learning.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Figure 2 below illustrates the convolutional network's structure in usage.

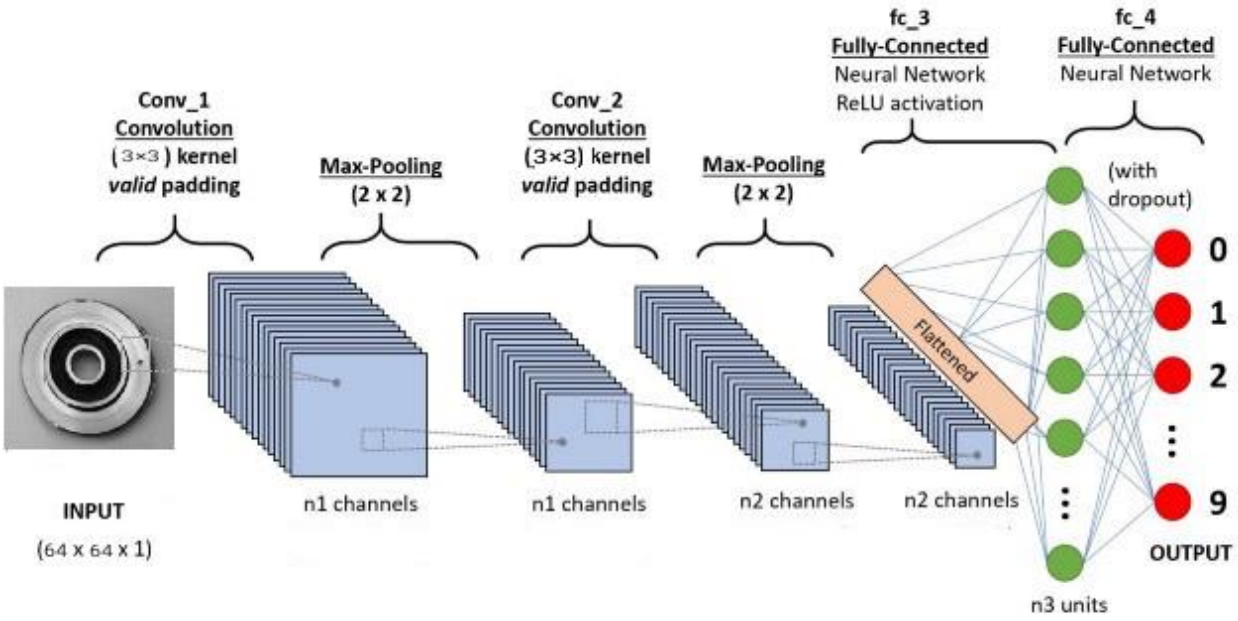

Figure 2: Convolutional Network Model Block Diagram

5. Results and Discussions:

The availability of the defect is identified with a 94 percent accuracy rate, and the type of the defect is identified with a 68 percent accuracy rate. To test the prepared model, a distinct image set was employed that was never used elsewhere.

The first convolutional layer has eight filters with a three by three kernel size matrix. Reduce the image size by half by making 2-pixel steps at a time.

First pooling layer: By using the maximum pooling matrix (2 by 2) and 2-pixel strides sequentially, the image size is further reduced by half.

The first convolutional layer and the second layer are identical. The first pooling layer is also same as the second layer. In order to be fed into the fully-connected layer, two-dimensional pixel data must be flattened into one dimension. In order to translate the scores into a chance that an image is defective, the output layer, which only has one unit, uses a sigmoid function for activation. With exception of the output layer, all layers use the Rectified Linear Unit (ReLU) activation function.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Detection Model:

Loading images for defect detection:

The below figure 3 shows the code for loading the images for training the detection model. There are two classes of images. There are 6633 images for training the model 715 images for testing it.

```
train_gen = ImageDataGenerator(rescale=1/255,
                horizontal_flip=True,
                zoom_range=0.2,
                shear_range=0.2   )
train_data = train_gen.flow_from_directory('/content/drive/MyDrive/casting_data1/casting_data/casting_data/train',
                                           class_mode='binary',
                                           batch_size=8,
                                           target_size=(64,64),
                                           color_mode='grayscale')
test_gen = ImageDataGenerator(rescale=1/255)
test_data = test_gen.flow_from_directory('/content/drive/MyDrive/casting_data1/casting_data/casting_data/test',
                                         class_mode='binary',
                                         batch_size=8,
                                         target_size=(64,64),
                                         color_mode='grayscale')
```

```
Found 6633 images belonging to 2 classes.
Found 715 images belonging to 2 classes.
```

Figure 3: Loading the images for defect detection

Training the model with the images:

The below figure 4 shows a Conv2D model being trained with the images for classification. It contains two convolutional layers along with two Maxpooling layers with an additional Relu layers

```
model.fit_generator(train_data,validation_data=test_data,epochs=10)
```

```
Epoch 1/10
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
830/830 [==============================] - 49s 59ms/step - loss: 0.0834 - accuracy: 0.9726 - val_loss: 0.0972 - val_accuracy: 0.9580
Epoch 2/10
830/830 [==============================] - 47s 57ms/step - loss: 0.0653 - accuracy: 0.9795 - val_loss: 0.1356 - val_accuracy: 0.9552
Epoch 3/10
830/830 [==============================] - 47s 57ms/step - loss: 0.0723 - accuracy: 0.9753 - val_loss: 0.3592 - val_accuracy: 0.9063
Epoch 4/10
830/830 [==============================] - 48s 58ms/step - loss: 0.0623 - accuracy: 0.9821 - val_loss: 0.2328 - val_accuracy: 0.9357
Epoch 5/10
830/830 [==============================] - 48s 57ms/step - loss: 0.0542 - accuracy: 0.9812 - val_loss: 0.1254 - val_accuracy: 0.9594
Epoch 6/10
830/830 [==============================] - 50s 60ms/step - loss: 0.0633 - accuracy: 0.9801 - val_loss: 0.1944 - val_accuracy: 0.9441
Epoch 7/10
830/830 [==============================] - 52s 62ms/step - loss: 0.0525 - accuracy: 0.9836 - val_loss: 0.1669 - val_accuracy: 0.9483
Epoch 8/10
830/830 [==============================] - 51s 61ms/step - loss: 0.0351 - accuracy: 0.9890 - val_loss: 0.2138 - val_accuracy: 0.9441
Epoch 9/10
830/830 [==============================] - 48s 58ms/step - loss: 0.0512 - accuracy: 0.9830 - val_loss: 0.1773 - val_accuracy: 0.9552
Epoch 10/10
830/830 [==============================] - 49s 60ms/step - loss: 0.0510 - accuracy: 0.9821 - val_loss: 0.0693 - val_accuracy: 0.9720
<keras.callbacks.History at 0x7fba8c816c90>
```

Figure 4: Training the model with the images

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Loading the test image:

Defective:

The below figure 5 shows a defective image being given as input to the model.

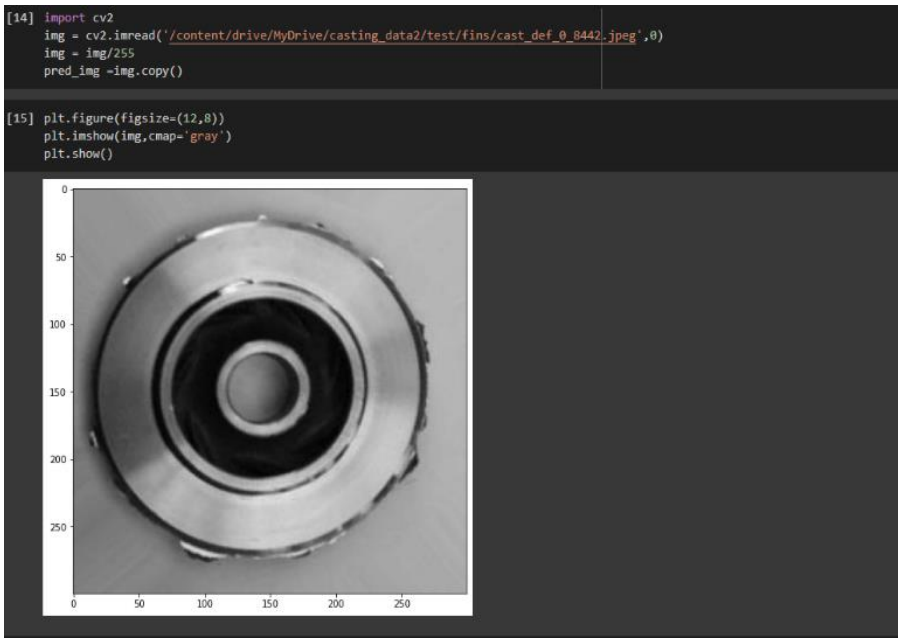

Figure 5: Loading the defective test image

Non-Defective:

The below figure 6 shows a non-defective image being given as input to the model.

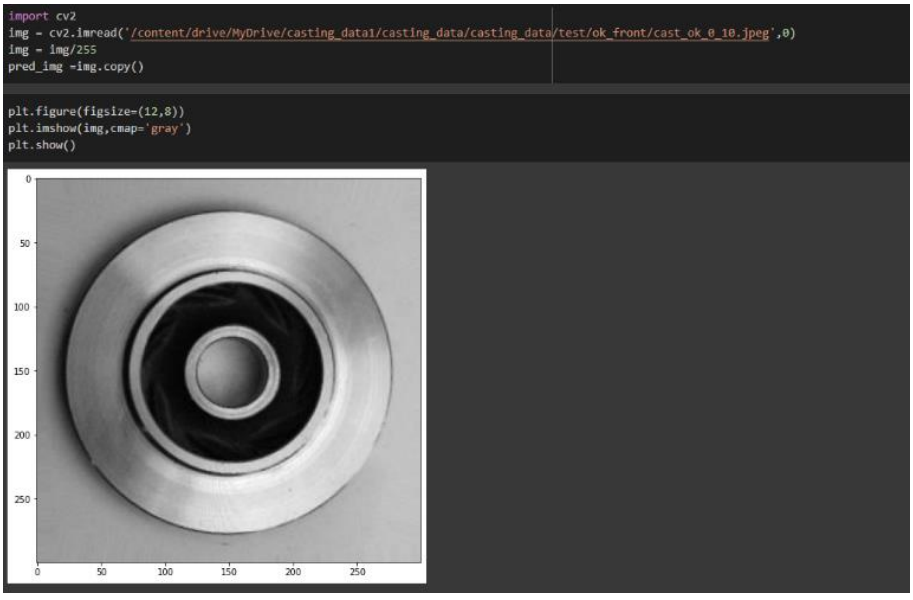

Figure 6: Loading the non-defective test image

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Result:

For Defective:

In the below Figure 7 a defective image is taken as input to the model which correctly classifies the image as defective.

```
test_image = image.load_img('/content/drive/MyDrive/casting_data2/test/heat crack/cast_def_0_9880.jpeg',target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result = model.predict(test_image)
```

```
result
```

```
array([[0.04392567]], dtype=float32)
```

```
if result[0]<=0.5:
    print('Defective')
else :
    print('Non Defective')
```

```
Defective
```

Figure 7: The model identifying the image to be defective

For Non-Defective:

In the below Figure 8 a Non-Defective image is taken as input to the model which correctly classifies the image as non-defective.

```
test_image = image.load_img('/content/drive/MyDrive/casting_data1/casting_data/casting_data/test/ok_front/cast_ok_0_10.jpeg',target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result = model.predict(test_image)
```

```
result
```

```
array([[0.70758575]], dtype=float32)
```

```
if result[0]<=0.5:
    print('Defective')
else :
    print('Non Defective')
```

```
Non Defective
```

Figure 8: The model identifying the image to be non-defective

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Classification Model:

Loading the Images:

The below figure 9 shows the code for loading the images for training the Classification model. There are two classes of images. There are 180 images for training and 16 images for testing the model.

```
[2]  train_gen = ImageDataGenerator(rescale=1/255,
                                    horizontal_flip=True,
                                    zoom_range=0.2,
                                    shear_range=0.2    )
     train_data = train_gen.flow_from_directory('/content/drive/MyDrive/casting_data2/train',
                                                class_mode='binary',
                                                batch_size=8,
                                                target_size=(64,64),
                                                color_mode='grayscale')
     test_gen = ImageDataGenerator(rescale=1/255)
     test_data = test_gen.flow_from_directory('/content/drive/MyDrive/casting_data2/test',
                                              class_mode='binary',
                                              batch_size=8,
                                              target_size=(64,64),
                                              color_mode='grayscale')

     Found 180 images belonging to 2 classes.
     Found 16 images belonging to 2 classes.
```

Figure 9: Loading the images for training the model

Training the Model:

The below figure 10 shows the model being trained. Here a Conv2d model with 2 convolutional layers, 2 Maxpooling layers and a Relu activation Function is used here and the above images are taken for training.

```
model.fit_generator(train_data,validation_data=test_data,epochs=10)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
Epoch 1/10
23/23 [==============================] - 60s 3s/step - loss: 0.6968 - accuracy: 0.5333 - val_loss: 0.6837 - val_accuracy: 0.6250
Epoch 2/10
23/23 [==============================] - 1s 58ms/step - loss: 0.6747 - accuracy: 0.6222 - val_loss: 0.6720 - val_accuracy: 0.5000
Epoch 3/10
23/23 [==============================] - 1s 59ms/step - loss: 0.6439 - accuracy: 0.6556 - val_loss: 0.6934 - val_accuracy: 0.5000
Epoch 4/10
23/23 [==============================] - 1s 55ms/step - loss: 0.5503 - accuracy: 0.7389 - val_loss: 0.6112 - val_accuracy: 0.5000
Epoch 5/10
23/23 [==============================] - 1s 58ms/step - loss: 0.4585 - accuracy: 0.7556 - val_loss: 0.4397 - val_accuracy: 0.9375
Epoch 6/10
23/23 [==============================] - 1s 58ms/step - loss: 0.3567 - accuracy: 0.8500 - val_loss: 0.5269 - val_accuracy: 0.6250
Epoch 7/10
23/23 [==============================] - 1s 57ms/step - loss: 0.2401 - accuracy: 0.9167 - val_loss: 0.9126 - val_accuracy: 0.5000
Epoch 8/10
23/23 [==============================] - 1s 59ms/step - loss: 0.1663 - accuracy: 0.9444 - val_loss: 0.3679 - val_accuracy: 0.7500
Epoch 9/10
23/23 [==============================] - 1s 58ms/step - loss: 0.1614 - accuracy: 0.9389 - val_loss: 0.4626 - val_accuracy: 0.6875
Epoch 10/10
23/23 [==============================] - 1s 60ms/step - loss: 0.0975 - accuracy: 0.9722 - val_loss: 0.0212 - val_accuracy: 1.0000
<keras.callbacks.History at 0x7f869f4396d0>
```

Figure 10: Training the model with the images for classification

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Loading the test image:

Heat-Cracks:

The below figure 11 shows an image of the Heat Crack defect being used as input to the model.

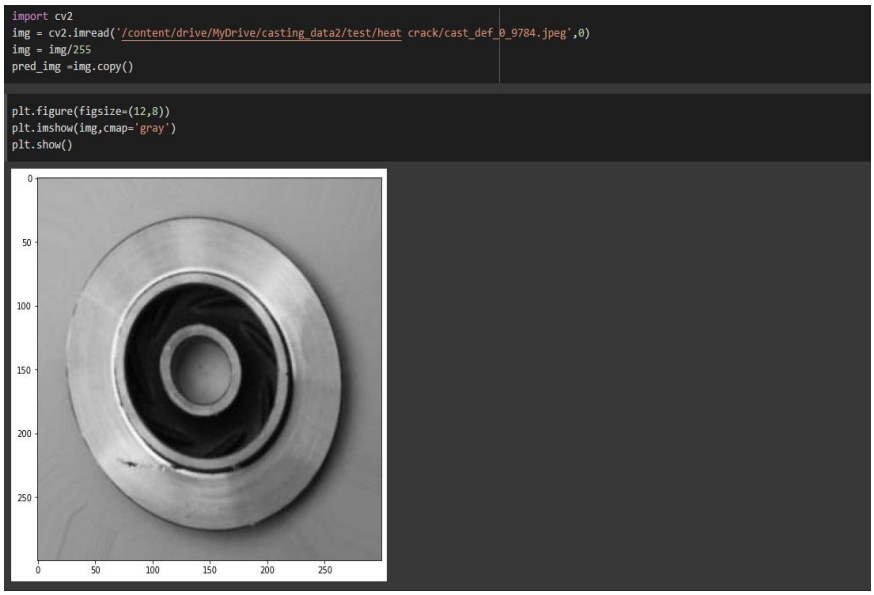

Figure 11: Loading the test image containing the heat crack defect

Fins:

The below figure 12 shows an image of the Fin defect which is used as input for the model.

Figure 12: Loading the test image containing the fin defect

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Results:

Heat-Cracks:

In the below Figure 13 a Defective image is given as input to the model which correctly classifies the image as defective and also predicts the type of defect as Heat Crack.

```
test_image = image.load_img('/content/drive/MyDrive/casting_data2/test/heat crack/cast_def_0_9784.jpeg',target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result = model.predict(test_image)
```

```
result
```

```
array([[0.9859657]], dtype=float32)
```

```
if result[0]<=0.5:
    print('Fins')
else :
    print('Heat Cracks')
```

```
Heat Cracks
```

Figure 13: The model predicting the image to have a Heat-Crack defect

Fins:

In the below Figure 14 a Defective image is given as input to the model which correctly classifies the image as defective and also predicts the type of defect as Fin.

```
test_image = image.load_img('/content/drive/MyDrive/casting_data2/test/fins/cast_def_0_8409.jpeg',target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result = model.predict(test_image)
```

```
result
```

```
array([[0.00539187]], dtype=float32)
```

```
if result[0]<=0.5:
    print('Fins')
else :
    print('Heat Cracks')
```

```
Fins
```

Figure 14: The model predicting the image to have a Fin defect

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Main model:

Both the defect detection and classification models are integrated in this model

Non-defective:

In the below figure 15 a Non-Defective image is given as input and it identifies it correctly.

```
test_image = image.load_img(testimg,target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result1 = detection_model.predict(test_image)

if result1[0]>=0.5:
    print('Not Defective')
else :
  result2 = classification_model.predict(test_image)
  print('The cast is found to be defective\nThe type of defect is:')
  if result2[0]<=0.5:
    print('Fins')
  else :
    print('Heat cracks')

Not Defective
```

Figure 15: The model classifying the image as non-defective.

Heat-Cracks:

The below figure 16 shows the image being classified as defective and the type of defect is identified as Heat Cracks.

```
test_image = image.load_img(testimg,target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result1 = detection_model.predict(test_image)

if result1[0]>=0.5:
    print('Not Defective')
else :
  result2 = classification_model.predict(test_image)
  print('The cast is found to be defective\nThe type of defect is:')
  if result2[0]<=0.5:
    print('Fins')
  else :
    print('Heat cracks')

The cast is found to be defective
The type of defect is:
Heat cracks
```

Figure 16: The model classifying the image as defective and the type of defect as Heat crack.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Fins:

The below figure 17 shows the image being classified as defective and the type of defect is identified as Fin.

```
test_image = image.load_img(testimg,target_size=(64,64),color_mode='grayscale')
test_image = image.img_to_array(test_image)
test_image = test_image/255
test_image = np.expand_dims(test_image,axis=0)
result1 = detection_model.predict(test_image)

if result1[0]>=0.5:
    print('Not Defective')
else :
  result2 = classification_model.predict(test_image)
  print('The cast is found to be defective\nThe type of defect is:')
  if result2[0]<=0.5:
    print('Fins')
  else :
    print('Heat cracks')
```

```
The cast is found to be defective
The type of defect is:
Fins
```

Figure 17: The model classifying the image as defective and the type of defect as Fins.

Relu activation function:

An activation function is a straightforward function that converts its inputs into outputs with a specific range. One of the most used functions used in deep learning models is the Rectified Linear Unit. On discovering any negative input, it returns to 0, but any positive value x causes it to return that value. This can be expressed as f(x)=max (0, x).

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

Conclusion and Future Perspectives:

The model used here managed to classify the images has been sufficiently accurate in identifying the defects and also in increasing the type of defect found. This can be more efficient than the human supervision. But the accuracy can be further increased by using a better algorithm by using better computing resources. This can make the model usable in cast industrial level. The number of type of defects identified can also be increased and also use images of better quality to further increase the accuracy of the model.

Available online at www.sciencedirect.com

ScienceDirect

materialstoday: PROCEEDINGS

References

[1] Duan, Liming & Yang, Ke & Ruan, Lang. (2020). Research on Automatic Recognition of Casting Defects Based on Deep Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3048432.

[2] Tang, Z., Tian, E., Wang, Y., Wang, L., & Yang, T. (2020). Non-destructive defect detection in castings by using spatial attention bilinear convolutional neural network. *IEEE Transactions on Industrial Informatics*, *17*(1), 82-89.

[3] Jain, S., Seth, G., Paruthi, A., Soni, U., & Kumar, G. (2020). Synthetic data augmentation for surface defect detection and classification using deep learning. *Journal of Intelligent Manufacturing*, 1-14.

[4] Dong, X., Taylor, C. J., & Cootes, T. F. (2020). Defect detection and classification by training a generic convolutional neural network encoder. *IEEE Transactions on Signal Processing*, *68*, 6055-6069.

[5] Hu, C., & Wang, Y. (2020). An efficient convolutional neural network model based on object-level attention mechanism for casting defect detection on radiography images. *IEEE Transactions on Industrial Electronics*, *67*(12), 10922-10930.

[6] Balasubramani, S., Balaji, N., Ramakrishnan, T., Sureshkumar, R., & Chelladurai, S. J. S. (2020). Defect identification in casting surface using image processing techniques. In Green Materials and Advanced Manufacturing Technology (pp. 191-198). CRC Press.

[7], Y., Zhang, K., & Wang, L. (2021). Metal Surface Defect Detection Using Modified YOLO. Algorithms, 14(9), 257.

[8] Zhao, W., Chen, F., Huang, H., Li, D., & Cheng, W. (2021). A new steel defect detection algorithm based on deep learning. Computational Intelligence and Neuroscience, 2021.

[9] Wang, S., Xia, X., Ye, L., & Yang, B. (2021). Automatic detection and classification of steel surface defect using deep convolutional neural networks. Metals, 11(3), 388.

[10] Nguyen, T. P., Choi, S., Park, S. J., Park, S. H., & Yoon, J. (2021). Inspecting method for defective casting products with convolutional neural network (CNN). *International Journal of*

Precision Engineering and Manufacturing-Green Technology, *8*(2), 583-594.

[11] Nikolić, F., Štajduhar, I., g_a_Classifi& Čanađija, M. (2022). Casting Defects Detection in Aluminum Alloys Using Deep Learning: a Classification Approach. International Journal of Metalcasting, 1-13.

[12] Lal, R., Bolla, B. K., & Ethiraj, S. (2022). Efficient Neural Net Approaches in Metal Casting Defect Detection. arXiv preprint arXiv:2208.04150.

[13] Xue, L., Hei, J., Wang, Y., Li, Q., Lu, Y., & Liu, W. (2022). A high efficiency deep learning method for the x-ray image defect detection of casting parts. Measurement Science and Technology, 33(9), 095015

[14] Lilhore, U. K., Simaiya, S., Sandhu, J. K., Trivedi, N. K., Garg, A., & Moudgil, A. (2022, March). Deep Learning-Based Predictive Model for Defect Detection and Classification in Industry 4.0. In *2022 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 1-5). IEEE.

[15] Chen, X., Lv, J., Fang, Y., & Du, S. (2022). Online Detection of Surface Defects Based on Improved YOLOV3. Sensors, 22(3), 817.