B. Tech. Project Report

Monitoring of Manufacturing Processes using Deep Learning

Submitted

By

Amlan Jyoti Pegu (200103010)
Karan Koch (200103057)

Under the supervision of

Dr. Sukhomay Pal

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

JANUARY-MAY 6th Semester (2023)

CERTIFICATE

It is certified that the work contained in the project report entitled "**Monitoring of manufacturing processes using deep learning**", by **Amlan Jyoti Pegu** (200103010) and **Karan Koch** (200103057) has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree or diploma.

Date:

Dr. Sukhomay Pal
Professor
Department Of Mechanical Engineering
Indian Institute Of Technology , Guwahati

DECLARATION

We certify that the writing we have submitted reflects our views in our own words, and when we've borrowed someone else's thoughts or words, we've properly acknowledged and referenced those sources. In addition, we affirm that we have followed all rules governing academic honesty and integrity and that we have not created or manipulated any idea, data, fact, or source in our work. We are aware that any violation of the aforementioned rules will result in Institute disciplinary action and may also result in penalties from the sources who were not properly cited or from whom proper permission was not obtained when required.

Amlan Jyoti Pegu
Karan Koch
Date :

ACKNOWLEDGEMENT

We would like to thank our supervisor, Dr Sukhomay Pal, for assigning us this project. His friendly nature and supportive attitude was great motivator for us. From him, We got a lot to learn and wowe uld specially thank him for constantly motivating us to work harder.

We dedicate the work to Dr. Sukhomay Pal and the Department of Mechanical Engineering at IIT Guwahati.

ABSTRACT

Deep learning is a rapidly growing field within machine learning, which is characterised by the use of neural networks with multiple layers. These networks can learn complex representations of data, which can then be used for a wide range of tasks, from image recognition to natural language processing. The present project aims to provide a comprehensive survey of the fundamentals of deep learning concepts, as well as the latest developments and applications in this exciting field. By examining the most recent research, we hope to gain a deeper understanding of the theoretical underpinnings of deep learning, as well as its practical applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Description
w	Weights
b	Bias
$\hat{y}$	Output
σ	Sigmoid function
J	Cost function
m	Total training examples
α	Learning rate
S	Softmax function
Ω	Euclidean or L2 norm
μ	Mean
σ^2	variance

CHAPTER 1

Introduction

1.1 Deep Learning:

Deep Learning is a subset of machine learning that powers numerous artificial intelligence (AI) applications and services that enhance automation by completing mental and physical tasks without the need for human intervention. Neural networks are used to implement deep learning.

1.2 CNN:

Convolutional Neural Network (CNN) is a popular deep learning neural network widely utilised in various image and video analysis applications. CNN is specifically designed to recognise and extract significant characteristics from visual data such as images, videos, and other multidimensional data. It uses convolutional layers to filter input data and reduce its dimensions while retaining essential features. As a result, CNNs are particularly effective in identifying patterns and objects within complex visual data, making them valuable tools for various computer vision tasks.

A Convolutional Neural Network (CNN) is composed of several layers that include convolutional, pooling, and fully connected layers. The convolutional layers use filters or kernels to scan the input image and identify significant features like edges and corners. The pooling layers then reduce the spatial dimensions of the output from the convolutional layers while retaining the essential features. This down-sampling helps to decrease the computational complexity of the network. Finally, the fully connected layers take the output from the previous layers and perform computations to produce the final classification or prediction. These layers work together to enable CNNs to identify patterns and objects within visual data, making them useful for image and video analysis tasks.

Convolutional Neural Networks (CNNs) are advantageous as they can automatically learn hierarchical features from images without manual extraction. This makes them useful for object detection, image segmentation, and facial recognition, among other tasks. Additionally, CNNs have been successful in audio and text analysis.

1.3 Data Augmentation:

Data augmentation is a technique frequently used in machine learning and deep learning to enhance the size and diversity of a dataset by applying various transformations to the original data. It is a widely adopted practice in tasks that require image, speech, or natural language processing.

Data augmentation is creating new samples that are similar to the original data but have variations in their features, such as rotation, scaling, flipping, colour adjustments, or adding noise. The goal of data augmentation is to enhance the model's generalisation and mitigate overfitting, where the model performs well on the training data but poorly on new, unseen data.

Data augmentation techniques can vary depending on the type of data and the desired transformations. For instance, in image processing, data augmentation can involve random cropping, resizing, or flipping of images. In speech recognition, it can include adding background noise, changing the pitch, or altering the speed of the audio. In natural language processing, it can involve replacing words with synonyms, adding or removing words, or changing the sentence order.

Overall, data augmentation is a simple yet effective technique that helps to increase the diversity of the dataset and improve the performance of the machine learning models. It has become a standard practice in many applications and is often used with other techniques, such as transfer learning and ensemble learning.

1.4: Metal Defects:

Metal surface defects are common in manufacturing and can significantly impact the quality of the final product. Some of the most common metal surface defects include rolled-in scale (RS), patches (Pa), crazing (Cr), pitted surface (PS), inclusion (In), and scratches (Sc).

1. Rolled-in scale (RS) refers to a type of defect caused by the presence of scale or oxide on the metal surface that has been rolled into the surface during the manufacturing process. This can result in a rough and uneven surface, affecting the overall appearance and quality of the product.

    i. Patches (Pa) are another type of metal surface defect that can occur due to variations in the composition or processing of the metal. Patches are typically characterised by areas of discolouration or uneven texture on the metal surface.

ii. Crazing (Cr) is a defect that appears as small, shallow cracks on the metal surface caused by thermal stress or strain during manufacturing or use. Crazing can weaken the metal and make it more susceptible to further damage.

iii. Pitted surface (PS) refers to forming small, deep pits or holes on the metal surface caused by corrosion, chemical reactions, or other environmental factors. Pitting can compromise the structural integrity of the metal and reduce its lifespan.

iv. Inclusions (In) are foreign particles or impurities that are present in the metal during manufacturing. These particles can cause defects on the metal surface and weaken the material's overall strength and durability.

v. Scratches (Sc) are common surface defects caused by contact with abrasive or sharp objects during manufacturing or handling. Scratches can reduce the aesthetic appeal of the product and create stress points that may lead to further damage.

In order to detect and classify metal surface defects such as RS, Pa, Cr, PS, In, and Sc, various image processing and machine learning techniques can be used. These techniques include feature extraction, segmentation, and classification algorithms, as well as data augmentation techniques to increase the size and diversity of the training dataset.

CHAPTER 2

Datasets

2.1 Datasets for the Architecture

The proposed architecture in this study aims to detect six common surface defects of hot-rolled steel strips, which include rolled-in scale (RS), patches (Pa), crazing (Cr), pitted surface (PS), inclusion (In), and scratches (Sc). The NEU Metal Surface Defects Database provided the dataset, obtained by scanning six hot-rolled steel strips, and is divided into the train, test, and validation classes, with six subclasses of defects in each class. The training dataset consists of

256 grayscale images for each defect subclass, while the test and validation classes have 12 grayscale images in each subgroup. The data is divided into six-folds, each containing unique defects to ensure reliable results during cross-validation. The same split was used as the previous state-of-the-art approach to compare results.

2.2 Use of data augmentation:

The purpose of data augmentation is to increase the diversity and variability of the training data to improve the model's ability to generalise to unseen data and enhance its robustness to different variations in the input data. Some common data augmentation techniques that we used in image classification tasks are:

i. Rescale: This parameter is used to normalise the pixel values of the images by scaling them. In this case, the pixel values are divided by 255 to bring them to the range of [0, 1], which is a common practice in image processing and deep learning.
ii. Shear range: This parameter specifies the range of shear angles (in radians) for randomly applying shearing transformations to the images. Shearing is a transformation that distorts the shape of an image by shifting the position of pixels along a certain direction.
iii. Zoom range: This parameter specifies the range of zoom levels for randomly applying zooming transformations to the images. Zooming is a transformation that changes the scale of an image by either enlarging or shrinking it.
iv. Horizontal flip: This parameter specifies whether to flip the images horizontally randomly. Horizontal flipping is a transformation that mirrors an image horizontally, which can be useful for training models to be invariant to left-right orientation.

In our work, we use the “ImageDataGenerator” Library in Keras to generate augmented images on-the-fly during training. The augmentation is applied to the original images to create variations, but it does not change the original image files or add more images to the dataset directory. The ImageDataGenerator will generate augmented images from these original images during training, effectively increasing the effective dataset size without actually adding more images to the original directory.Data augmentation can help prevent overfitting, which occurs when a model becomes too specialised to the training data and performs poorly on new, unseen data.

CHAPTER 3

Methodology for the CNN Model

3.1 Problem Statement:

The objective of this project is to create a deep learning model that can effectively identify and classify various types of metal surface defects using the Metal Surface Defects Dataset. Specifically, the model will be trained to distinguish between six defect categories: Crazing, Inclusion, Patches, Pitted Surface, Rolled-in Scale, and Scratches. The performance of the model will be evaluated using relevant metrics to measure accuracy and precision and compared to current state-of-the-art methods for metal surface defect classification.

3.2 Overview:

The code is a Convolutional Neural Network (CNN) model designed for classifying images of metal surface defects using the NEU Metal Surface Defects Data dataset. The model architecture consists of a series of convolutional layers that utilise ReLU activation functions, followed by max-pooling layers. The output is then flattened and connected to a stack of fully connected layers that also use ReLU activation functions. The final output layer utilises a softmax activation function. To optimise the model, categorical cross-entropy loss and RMSprop optimiser are applied. ImageDataGenerator is used to implement data augmentation to the training dataset. The augmentation techniques include rescaling, shear range, zoom range, and horizontal flip. The training dataset is then fed into the model in batches of 10, while the validation dataset is generated using ImageDataGenerator without any data augmentation.

The custom_callback class is implemented to track the accuracy of the model during training and stop the training process once the accuracy reaches 97.8%. Finally, the model is trained for 15

epochs, and the training history is saved in the history variable.

3.3 Datasets:

The dataset used is the NEU Metal Surface Defects Database, which includes 256 grayscale images of each defect subclass in the training dataset and 12 grayscale images in each subclass of the test and validation datasets. The data is divided into six folds to ensure reliable cross-validation results, and the same split was used for comparison with the previous state-of-the-art approach.

3.4 Data augmentation techniques :

The code utilises various data augmentation techniques during the training process to improve the model's ability to generalise. The ImageDataGenerator class is used to apply these techniques, which include rescaling, shear range, zoom range, and horizontal flip.

Rescaling is used to normalise the pixel values of the input images to a range of [0, 1]. The shear range is applied to randomly shear the images along the x-axis by a factor within a specified range. The zoom range is used to randomly zoom the images by a factor within a specified range. Horizontal flip is applied to flip the images horizontally randomly.

These augmentation techniques create new images from the existing ones, increasing the size of the training dataset and improving the model's performance on unseen data. They make the model more robust to variations in the input data, such as changes in lighting, orientation, and scale.

3.5 Model Architecture:

The model architecture in this code consists of several layers. The input layer takes images with a shape of (200, 200, 3), where 3 refers to the number of colour channels (red, green, and blue). This is followed by a stack of three convolutional layers, each with 32 filters and a kernel size of (3, 3), which extract features from the input images. ReLU activation functions are applied after each convolutional layer to introduce non-linearity.

Max-pooling layers with a pool size of (2, 2) are then used to downsample the feature maps generated by the convolutional layers. This helps to reduce the dimensionality of the feature maps

and extract the most important features. The output of the last max-pooling layer is then flattened into a 1D vector, which is then fed into a stack of two fully connected layers with 64 units each.

ReLU activation functions are again applied after each fully connected layer to introduce non-linearity. The final output layer consists of 6 units, which corresponds to the number of classes in the NEU Metal Surface Defects Data dataset. A softmax activation function is applied to the output layer, which produces a probability distribution over the classes.

The model is trained using categorical cross-entropy loss, which is commonly used for multi-class classification problems, and RMSprop optimizer, which is a variant of stochastic gradient descent that adjusts the learning rate adaptively.

Regenerate response

3.6 Compilation of the model:

In this code, the model is compiled with categorical cross-entropy loss and RMSprop optimiser. The choice of categorical cross-entropy loss as the loss function is appropriate for multi-class classification problems. The RMSprop optimiser is responsible for updating the weights of the model during the training process. The learning rate of the optimiser is set to 0.001, which is a commonly used value in deep learning models. To evaluate the performance of the model, metrics such as accuracy, precision, recall, and F1-score are used. These settings are crucial in ensuring that the model is effectively trained and evaluated.

3.7 Training of the model:

In this code, the model is trained for 15 epochs using the fit generator method of the model object. During training, the training dataset is generated in batches of 10 using ImageDataGenerator with data augmentation techniques, and the validation dataset is generated using ImageDataGenerator without any data augmentation. The custom callback class is used to monitor the accuracy of the model during training and to stop the training process once the accuracy reaches 97.8%. The training history is stored in the history variable, which contains the loss and metrics values for each epoch. The model is trained using a GPU to speed up the training process. Overall, the training process is designed to optimise the model parameters for the given problem and to achieve high accuracy on the validation set.

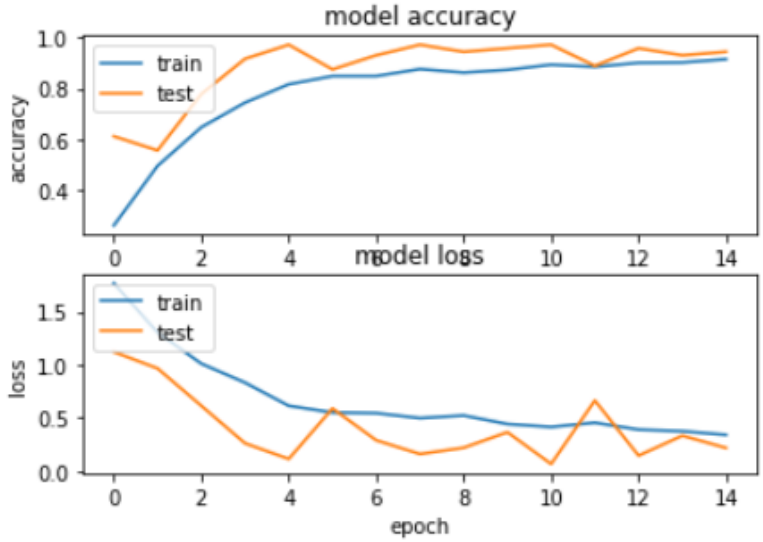
CHAPTER 4

Algorithm and Results

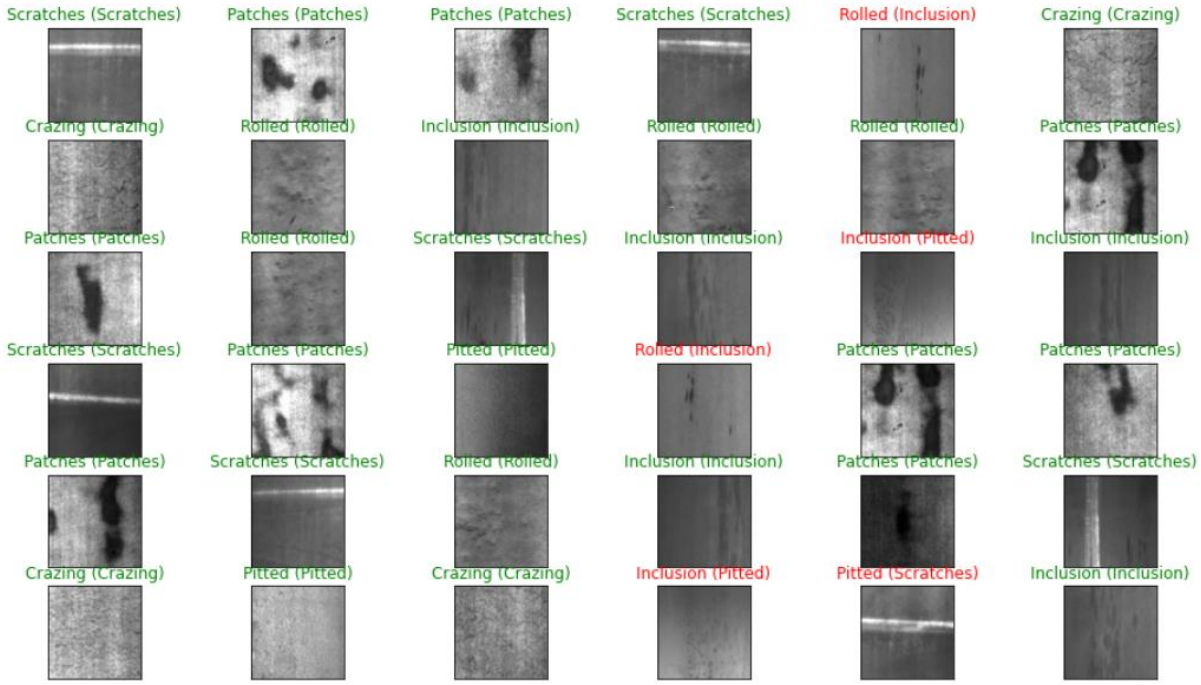
4.1 The Algorithm:

4.2Results and Observations:

1. We can see from the below graph that with the increase in the number of epochs of our model, the loss function decreases and the accuracy
    - Increases loss='categorical crossentropy'



2. Accuracy of our train model is more than 95 per cent, which is quite good enough to make a prediction of unknown datasets.
3. Finally, we feed the test data to our model and found the following observations:



The image above displays its predicted label and true label in a subplot with a title that is coloured green if the prediction is correct and red if it is incorrect

4. The Test accuracy is more than 90 per cent.

```
3/3 - 3s - loss: 0.1999 - accuracy: 0.9167 - 3s/epoch - 1s/step
Test accuracy: 0.9166666865348816
```

4.3 Summary of the Model Results:

The model demonstrated impressive performance by achieving an 85.7% accuracy on the validation dataset after 15 epochs of training. During the training process, the accuracy steadily increased from 18.4% in the first epoch to 93.5% in the 15th epoch. Similarly, the validation accuracy improved from 18.1% to 85.7% over the same period. Additionally, the model's training loss decreased from 2.2449 in the first epoch to 0.2018 in the 15th epoch, and the validation loss decreased from 1.6016 to 0.5062. It is noteworthy that the training accuracy curve spiked initially before levelling off, while the validation accuracy curve increased gradually before flattening. The training loss curve decreased sharply in the first few epochs and then plateaued, whereas the validation loss curve decreased gradually before plateauing. The model's performance showed signs of overfitting, with a higher training accuracy than the validation accuracy.

CHAPTER 5

Future Work