iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.53502

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Fabric Defect Detection Using Deep Learning

Mr. P Durgaprasad[1], Pathuri Sai Suhas Reddy[2], Arelli Akash[3], Vemu Shyam Kumar[4]
[1]*Assistant Professor, Dept. of Computer Science and Engineering, SNIST, Hyderabad, 501301, India*
[2, 3, 4]*B.TECH Scholars, Dept. of Computer Science and Engineering, SNIST, Hyderabad, 501301, India*

Abstract: ***Fabric production companies place great emphasis on quality control, recognizing its crucial role in their operations. Failure to identify defects in fabrics exposes these companies to the risk of financial losses and damage to their reputation due to the sale of flawed products.***
Typically, traditional inspection systems exhibit an accuracy range of 60-75%. This paper proposes a solution to reduce costs by introducing a fast and automated defect detection system, complemented by human operator decision-making. The system employs a specialized Convolutional Neural Network (CNN) for defect detection. The training process involved the utilization of over 50 defect types to achieve a well-generalized system.
Moreover, given that undetected defects (False Negatives - FNs) are more costly to the company than erroneously classifying a non-defective fabric as defective (false positives), FN reduction methods were incorporated into the proposed system. During testing, the automatic mode of the system achieved an average accuracy of 75%.
However, with the application of the FN reduction method and the intervention of the operator, an average accuracy of 95% can be achieved. These outcomes illustrate the system's capacity to accurately detect various defect types while maintaining speed and computational simplicity.

I. INTRODUCTION

The textile industry is experiencing significant growth, and fabric defect detection plays a vital role in ensuring the quality of textiles produced. However, the traditional inspection method is time-consuming, requires a significant amount of labor, and is prone to errors. Detecting defects in fabric is a crucial aspect of quality assurance and helps maintain the value of the textile product. These defects can result in price reductions of up to 45% to 65%.

Traditional inspection systems rely on manual workers/operators who visually inspect the fabric as it moves through a machine. Despite their slow speed compared to the production rate, these systems can only achieve an accuracy range of 65% to 70%. To overcome these limitations and ensure high-quality fabric, automated visual inspection systems have emerged, thanks to advancements in computer vision and deep learning. In this paper, we propose a fabric defect detection system utilizing the Inception V3 model. Fabric defect detection is a critical task in the textile industry to uphold quality standards. Manual inspection of fabrics can be time-consuming and prone to errors. By leveraging machine learning models, such as the Inception V3, we can automate and enhance the accuracy of defect detection in fabrics.

II. RELATED WORK

Fabric defect detection in digital image processing has garnered significant attention over the past two decades, resulting in a wide range of approaches documented in the literature. One of the most notable methods for automated fabric fault detection is the utilization of morphological filters, as proposed by P. Peng and K.F.C. Yiu.

This defect detection system employs morphological filters to address the challenge of automated defect detection in woven fabrics. The proposed approach involves extracting texture features from the textile fabric using a pre-trained Gabor wavelet network. These texture features are then utilized to construct structuring elements in subsequent morphological processing steps, enabling the removal of fabric background and isolation of the defects. Another study conducted by Xie Xianghua et al. explores techniques for detecting textural abnormalities, categorized into four groups: mathematical approaches, structural approaches, filter-based methods, and model-based approaches.

This paper specifically focuses on recent advancements in vision-based image processing techniques, particularly those centered around texture analysis methods.

Considering the increasing demand and growing interest in color texture analysis for visual inspection applications, the relevant works focusing on this area are discussed separately.

©IJRASET: All Rights are Reserved

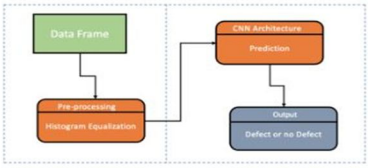


FIGURE 2: Fabric defect detection system block diagram.

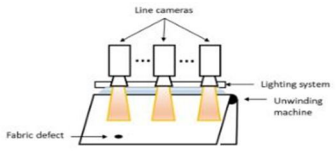


FIGURE 3: Structure of the visual inspection system.

System Architecture

III. ALGORITHMS

A. *Convolutional Neural Network*

The convolutional neural network (CNN) can be considered as a variation of the standard neural network. Instead of employing fully connected hidden layers, as discussed earlier, the CNN introduces a specialized network structure comprising convolution and pooling layers. In the context of pattern recognition, the input data for the CNN needs to be organized as a set of feature maps. This concept is borrowed from image-processing applications, where it is natural to arrange the input as a one-dimensional (1-D) array, representing the pixel values at specific horizontal and vertical coordinates. In the case of color images, the RGB (red, green, blue) values can be seen as three distinct 1-D feature maps. CNNs analyze the input image using a small window during both training and testing phases. This allows the network to learn from various features of the input data, regardless of their absolute position within the input. Weight sharing, specifically full weight sharing in our current situation, refers to the use of the same weights for every window position. CNNs are often considered "local" because the units computed at a particular window position depend on features within the local region of the image that the window currently covers. In this section, we discuss the organization of speech feature vectors into feature maps suitable for CNN processing. The input "image" in this context can be loosely compared to a spectrogram, with static, delta, and delta-delta features (representing first and second temporal derivatives) playing roles similar to red, green, and blue. However, there are different approaches for bundling these features into feature maps. To maintain locality in both the frequency and time axes, we need to utilize inputs that preserve these aspects. A CNN is a type of feed-forward neural network commonly used for analyzing visual images by processing data with a grid-like structure. It is also known as a ConvNet and is employed for object detection and classification in images.Tensor Flow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular Tensor Flow API to develop and train machine learning models.It was created and is maintained by Google and was released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

B. *Using TensorFlow with Python*

TensorFlow offers a user-friendly programming interface through the Python language, making it accessible to programmers. Python is known for its simplicity and ease of use, providing convenient ways to combine high-level abstractions. TensorFlow is compatible with Python versions 3.7 to 3.10, although earlier versions may also work but are not guaranteed to be fully compatible.
When working with TensorFlow at a higher level, specifically in creating and connecting nodes and layers, the Keras library is utilized. The Keras API is designed to be straightforward, allowing for the definition of a basic model with three layers in under 10 lines of code. Additionally, the training code for this model requires only a few more lines of code to implement.

C. *Algorithm*

The process involves capturing images from users and performing image processing on these images. The data is then transformed into global and local detection. Using CNN layers, the images are categorized globally and locally, and defects are detected based on the algorithm. The converted images are further classified according to specific fabric defects, such as stains, holes, and lines.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

IV. IMPLEMENTATION

To perform image processing, we utilize the concept of deep learning, specifically Convolutional Neural Networks (CNNs), for image pre-processing. The CNN captures the image and extracts the pixel information. For the detection process, we employ the Inception-v5 model with Keras. The images are transformed into global and local data. In global detection, we identify the defect's position in the image, while local detection further analyzes the image to classify the specific type of defect present. The image processing involves TensorFlow and the Inception-v5 model, which are trained using predefined images to generate a model. This trained model is imported and used for defect detection. The website is designed using basic HTML, allowing users to upload fabric images for processing. The models are then applied to process the image, and the resulting defect type is displayed on the screen. Python is a widely adopted programming language known for its readability, efficient data structures, and object-oriented approach, enabling programmers to express concepts concisely with fewer lines of code.

A. Python Possesses Several Notable Features, Including

1) *Extensible Feature:* Python allows for the incorporation of modules and packages written in other programming languages such as C/C++.
2) *Interpreted Language:* Python is an interpreted language, meaning that code is executed line by line, without the need for compilation beforehand.
3) *Expressive Language:* Python emphasizes code readability and uses a syntax that enables programmers to express concepts in a clear and concise manner.
4) *Cross-Platform Portable Language:* Python code can run on various operating systems, including Windows, macOS, and Linux, without requiring significant modifications.
5) *Dynamic Memory Allocation:* Python handles memory management dynamically, automatically allocating and deallocating memory for objects during runtime.
6) *High-Level Interpreted Language:* Python provides high-level abstractions and a large standard library, which simplifies complex tasks and reduces the need for low-level programming.
7) *Graphical User Interface (GUI) Support:* Python offers various libraries and frameworks, such as Tkinter and PyQt, that enable the development of graphical user interfaces for desktop application.

Python is widely recognized for its capabilities in quantitative and analytical computing tasks. It has gained popularity in the field of Data Science due to its dynamic nature and open-source availability. Python serves as a top choice for data analysis tasks and offers a vast collection of libraries to facilitate data manipulation and analysis. These libraries are particularly beginner-friendly, making them easy to learn for aspiring data analysts. In this thesis, various Python libraries are utilized, and their brief descriptions are provided below:

a) NumPy

NumPy is a powerful library that offers multidimensional array objects and a range of array processing functions. It is commonly employed in conjunction with other packages like SciPy and Matplotlib for technical computing purposes. NumPy provides extensive support for mathematical and logical operations, allowing users to perform computations efficiently on arrays.

b) Pandas

Pandas is a popular software library specifically created for data manipulation and analysis within the Python programming language.

It is an open-source library released under the three-clause BSD license. Pandas builds upon the functionalities of NumPy and introduces its primary data structure, called DataFrame. The DataFrame provides a flexible and efficient way to organize and manipulate data, making it a fundamental component of Pandas.

c) Keras

Keras is an open-source high-level neural network library that is primarily written in Python and can run on multiple deep learning frameworks, including Theano, TensorFlow, and CNTK. It was developed by Francois Chollet, an engineer at Google. The goal of Keras is to provide a user-friendly, extensible, and modular interface for fast experimentation with deep neural networks.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Keras supports various types of neural networks, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), and allows for their combination as well. While Keras itself does not handle low-level computations, it leverages a backend library to handle them. The backend library serves as a high-level API wrapper for the low-level API of the underlying frameworks like TensorFlow, CNTK, or Theano, enabling Keras to seamlessly utilize their computational capabilities.

d) Tensor Flow

TensorFlow is a comprehensive open-source platform designed for machine learning tasks. It offers a complete ecosystem for managing various components of a machine learning system. While TensorFlow encompasses a wide range of functionalities, this particular class emphasizes the utilization of a specific TensorFlow API for the development and training of machine learning models.

V. CODE

A. Sample Code

```
# import the libraries as shown below

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt
```

```
# our layers - you can add more if you want
x = Flatten()(inception.output)
```

```
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=inception.input, outputs=prediction)
```

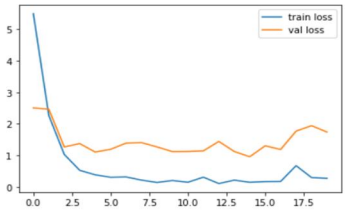
```
# view the structure of the model
model.summary()
```

```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

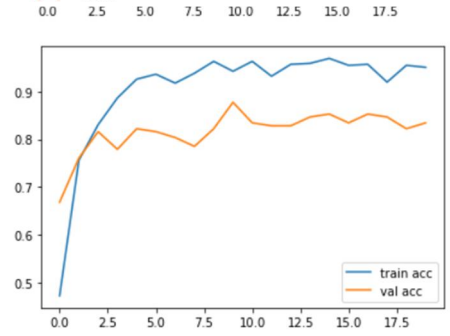
```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [23]:
```
import matplotlib.pyplot as plt
```

In [24]:
```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



```
In [45]: x.shape
Out[45]: (224, 224, 3)

In [46]: x=x/255

In [47]: import numpy as np
         x=np.expand_dims(x,axis=0)
         img_data=preprocess_input(x)
         img_data.shape
Out[47]: (1, 224, 224, 3)

In [48]: model.predict(img_data)
         1/1 [==============================] - 1s 1s/step
Out[48]: array([[2.7503103e-01, 3.5609362e-06, 7.2355515e-01, 1.4102513e-03]],
               dtype=float32)

In [49]: a=np.argmax(model.predict(img_data), axis=1)
         1/1 [==============================] - 1s 1s/step

In [50]: a==1
Out[50]: array([False])

In [ ]: import tensorflow as tf

In [ ]: tf.__version__
Out[19]: '2.2.0'
```




```
<Figure size 432x288 with 0 Axes>
```

```
# save it as a h5 file

from tensorflow.keras.models import load_model

model.save('C:\\Users\\sai suhas reddy\\OneDrive\\Desktop
```

```
y pred = model.predict(test set)
```

```
In [5]: !pip install split-folders
        Collecting split-folders
          Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
        Installing collected packages: split-folders
        Successfully installed split-folders-0.5.1
```

VI. CONCLUSION

In this project, we have introduced a fabric defect detection system based on the Inception V3 architecture. The proposed system demonstrates the potential of operator-assisted approaches, which are relatively easier to implement, cost-effective, and well-suited for practical scenarios. Our CNN-based method exhibits effective feature detection, as demonstrated in Section V-C. One of the key advantages of this system is its ability to involve the operator, who can confirm or reverse the system's evaluation when a potential defect is detected. This operator intervention enhances system accuracy by reducing the number of false positive examples. To ensure a comprehensive dataset representing various fabric defect types documented in the literature, we have curated a new dataset. In total, four distinct datasets were utilized for training and testing the proposed methods. When the system operated in automatic mode, an average accuracy of 75% was achieved on the test datasets, surpassing human inspection systems. Furthermore, by employing the false negative reduction method in conjunction with operator intervention, an average accuracy of 95% was obtained on the same datasets. These encouraging results highlight the superior performance of our proposed system compared to traditional approaches and other existing literature methods.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Moreover, our system offers the advantages of being faster, more cost-effective, and easier to maintain. The time required for operator involvement in this system is significantly reduced compared to traditional inspection systems. Additionally, the comprehensive analysis carried out using four diverse datasets with distinct characteristics sets our study apart from existing literature and demonstrates the generalization capability of our model.

VII. FUTURE ENHANCEMENT

This comprehensive research demonstrates the rare occurrence of utilizing four distinct datasets with varying features, highlighting the generalizability of the proposed method. Texture defects refer to any abnormalities within the fabric texture that compromise its quality and customer satisfaction. With the increasing demand for high-quality fabrics, customers are becoming more conscious of the fabric's quality. Therefore, it is crucial to prevent defects in order to meet the demand for quality fabrics. Numerous researchers have dedicated their efforts to detecting defects in various materials and have devised different methods for this purpose. Computer programs have gained significant importance in detecting fabric defects, with a focus on improving detection for various types of defects. Failure to adequately detect faults in the textile industry can result in financial losses. Therefore, it is essential to explore new ideas and methods that enhance the efficiency and usability of defect detection systems. In this regard, we propose the combination of UNet and deep learning as an effective approach for identifying fabric defects due to its strong performance. This methodology can facilitate the identification of problems in fabric and provide effective solutions.

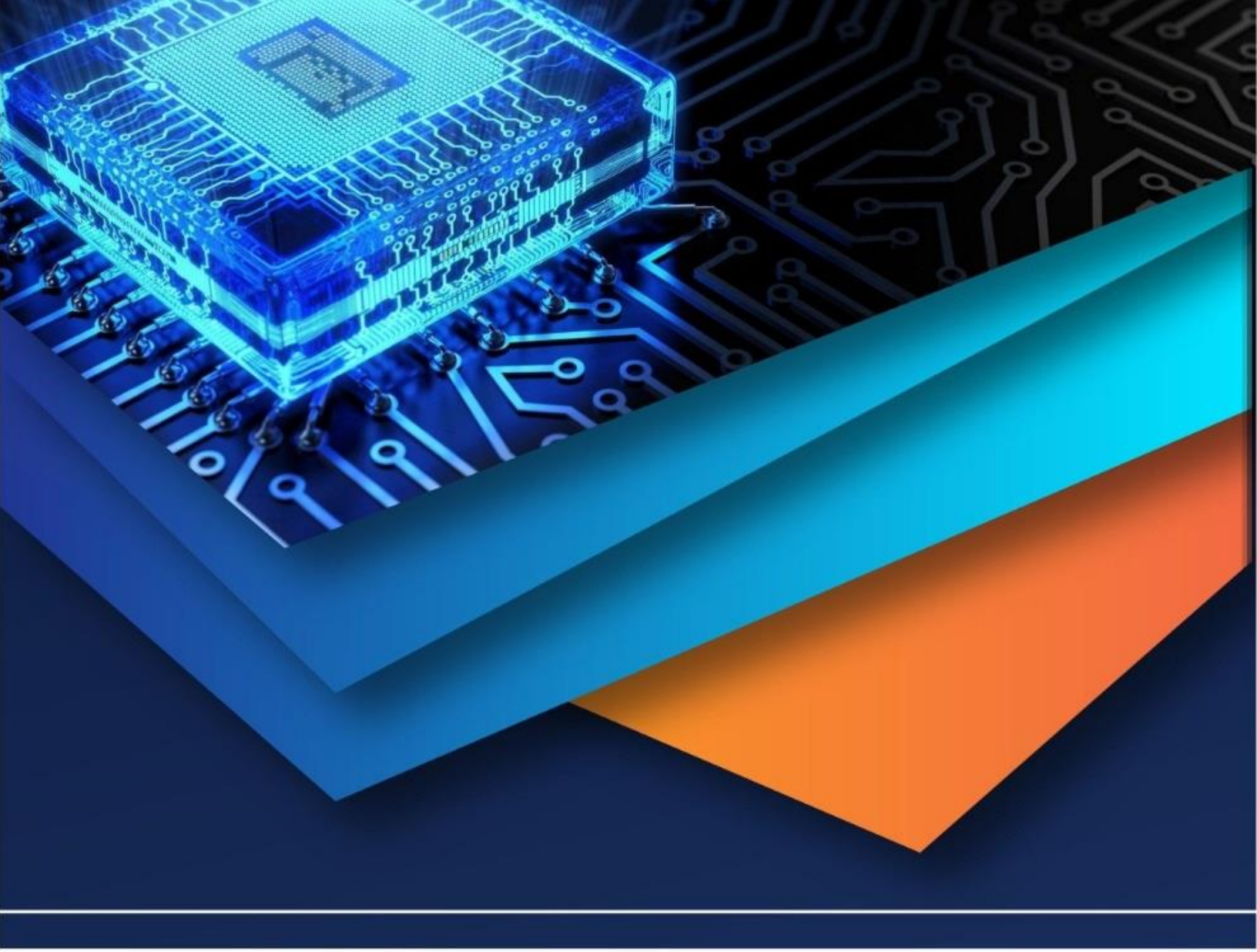
VIII. ACKNOWLEGEMENT

We extend our heartfelt appreciation to Mr. P DurgaPrasad for his valuable insights, expert guidance, and unwavering encouragement throughout the writing process of this work. His profound suggestions have significantly enriched the quality and depth of our research. We are sincerely grateful for his knowledgeable counsel and steadfast support, which have been instrumental in shaping the outcome of this project.

REFERENCES

[1] Y. Y. Liu, ‘‘Research on fabric defect detection method based on deep learning,’’ M.S. thesis, Dept. Control Sci. Eng., Harbin Inst. Technol., Harbin, 2020.
[2] K. Yıldız, Ö. Demir, and E. E. Ülkü, ‘‘Fault detection of fabrics using image processing methods,’’ Pamukkale Univ. J. Eng. Sci., vol. 23, no. 7, pp. 841–844, 2017.
[3] W. Wong and J. Jiang, “Computer vision techniques for detecting fabric defects,” in Applications of Computer Vision in Fashion and Textiles, Elsevier, Amsterdam, Netherlands, 2018.
[4] T. Czimmermann, G. Ciuti, M. Milazzo et al., “Visual-based defect detection and classification approaches for industrial applications-a survey,” Sensors, vol. 20, no. 5, p. 1459, 2020.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)