Research Reports on Computer Science
https://ojs.wiserpub.com/index.php/RRCS

Research Article

Deep Learning Based Fabric Defect Detection

Syeda Rabia Arshad* **and Muhammad Khuram Shahzad**

Faculty of Computing, National University of Sciences and Technology, Islamabad, Pakistan
Email: sarshad.msds20seecs@seecs.edu.pk

Received: 25 December 2023; **Revised:** 25 February 2024; **Accepted:** 5 March 2024

Abstract: Ensuring quality standards is a crucial stage within the textile sector. Automated classification of the fabric defects is a vital step during the fabric manufacturing process in order to prevent any faulted fabric from being supplied to the market. The defects on the surface of the fabric were manually identified by the individuals but this poses problems in terms of human-error and is also time-consuming. Efforts have been made to achieve better precision in defect detection through image processing studies, leading to the development of automated systems. In this study, some high-performing deep learning models are applied including ResNet and VGG-16 and illustrated how these algorithms can be used in the domain of textile manufacturing for fabric defect detection. A combination of images are used ranging from patterned and textured to plain for better defects recognition on any given fabric. The algorithm VGG-16 has displayed 73.91% accuracy while the ResNet algorithm has shown 67.59% accuracy.

***Keywords*:** deep learning, computer vision, ResNet, VGG-16, fabric defect detection

1. Introduction

Fabric defects are flaws in the material that diminish their quality, reducing their value and utility. Once these defects are in the market, they cannot be rectified or reversed. Hence, ensuring a flawless product is essential to deliver high-quality goods [1]. Quality control plays a pivotal role in guaranteeing customer satisfaction and reducing production costs.

Defects are caused in the fabric during the manufacturing process due to various reasons like weaving, knitting process, raw material, spinning and dying. As a result, faulted fabrics are created which may lower the customer satisfaction. Substandard products are mostly returned to the textile mill for repair or replacement which increases the overall cost apart from the production charges. Hence, it is necessary to identify the defects on the fabric (if any) because only then the manufacturers will be able to figure out how the defects are being caused and then the necessary steps could be taken to rectify this issue. Once we know how the defects are being manifested, the process of manufacturing could be improved so that the defects do not occur in the first place, instead of fixing the defects whenever it occurs. For creating high-quality products to be delivered into the market, it is crucial that the fabric manufacturing process is improved from scratch [2, 3].

Another challenge in fabric defect detection is that detecting surface defects on fabric is a manual process that necessitates the employment of an individual to physically inspect each piece of fabric, thoroughly examining it from all angles in search for potential flaws. This is a time-consuming task, as a single person cannot perform it as rapidly as needed, resulting in increased labour costs since one person alone cannot efficiently handle this labour-intensive duty. Furthermore, this manual inspection process is prone to errors, given that fabric is produced rapidly by machines, making it challenging to identify defects with the naked eye.

DOI: https://doi.org/10.37256/rrcs.3120244156

Some flaws may go unnoticed, potentially leading to the production of defective fabric. Hence, there is a compelling need for the automation of this process. Automated defect detection, without the need for human intervention, will eliminate the manual labour traditionally associated with this task. Being automated, it will exhibit a reduced susceptibility to errors, scrutinise details thoroughly, and, not least, prove to be cost-effective. Along with finding out any defects on the fabric, it is important to classify the defects into their respective types. The defects must be occurring in a specific pattern or might show some similar behaviour. If we figure out some similar types of defects, categorization can get much easier. One of the many benefits of categorization is that the manufacturing process can be improved by focusing on those particular steps that are causing some specific types of defects. With the count of each class of defect, we can know which step in the process is the most problematic.

Some other researches done in this field are those that employ the methodology of grey-level co-occurrence matrix [4, 5], Gabor filter [6, 7], CZI-Net [8] and others but these mentioned here are either too computationally heavy or are not easily scalable as the number of samples increases. In this study, an automated solution for fabric defect detection is created using deep learning algorithms like ResNet and VGG-16. This research aims to provide an efficient solution to overcome the limitations arising from a lack of data. Within this study, an automated resolution is created for the issue being discussed. Along with the detection of defects, this system also classifies the detectors into their respective categories which are vertical defects, horizontal defects, and holes. The system is trained on plain and patterned fabrics which means it is able to detect defects on any type of fabric, be it plain, textured or patterned. The system inputs the images of the fabric and it detects defects on the surface along with its respective categories. This research has produced remarkable results, characterised by a substantial enhancement in performance. Moreover, it has achieved equitable outcomes across all three categories, effectively mitigating the considerable data imbalance.

2. Literature Review

Talking about some of the studies related to this field, a method known as Gabor filter which utilises one-class classification (OCC) alongside Gabor filters. In OCC, a skilled classifier can define a classification boundary based solely on the information from the positive class. Consequently, OCC simplifies the problem by focusing exclusively on positive fabrics. This eliminates the need to collect and handle various fabric defects, as well as worries about imbalanced datasets or the lack of negative samples. The researchers have developed a Gabor filter bank with a range of orientations and bandwidths to analyse different fabric textures. This method facilitates the creation of customised Gabor filter designs for each fabric texture [9]. The Gabor filter acts as a strong representation of the simple cells found in the human visual cortex, effectively capturing textures. It is formed by multiplying a sinusoidal waveform with a Gaussian function. The researchers focused on assessing the model's effectiveness with plain, patterned, and rotated fabric images.

Another research done in this field is the grey-level co-occurrence matrix, also referred to as the spatial grey-level dependence method, which aggregates the probability of certain pairs of grey levels co-occurring under two criteria: the distance between pixels and their relative orientation [10]. The probability assessment is determined by tallying the occurrences of various grey levels within a defined region, while taking into account a specified displacement value. It's imperative for the statistics to remain unchanged despite shifts in grey levels, ensuring that classification is not influenced by tone variations [11]. To generate the co-occurrence data, various parameters must be defined, such as the window size, pixel distances (d), orientation (theta), and the number of quantified grey levels (G). A moving window systematically moves across the entire image, computing the GLCM at each position it covers. Subsequently, the energy feature is extracted from the GLCM. Then these energy values are compared to a reference, and if any deviation exceeding a specific threshold is detected, it is identified as a defect. The smallest window size must be designed to encompass a minimum of one periodic pattern within the pattern or the texture.

After analysing these related works in the field of fabric defect detection, it can be concluded that most of these studies use a window or a filter-based approach. The Gabor filter uses a linear filter for texture analysis. It checks for any specific frequency content in the image around the point of analysis. Due to the linear nature of the filter, it takes too much time to perform the defect detection process. It captures the features and stores them into a feature vector and the dimensions of the feature vector also keeps on increasing due to the linearity of the filter. This also causes redundancy of features which reduces the recognition rate. On the other hand, the grey-

level co-occurrence is a matrix that is defined over an image to be the distribution of co-occurring pixel values at a given offset. It converts an image into matrix of pixels and finds the combination of grey levels found within the image. However, the limitation of this approach is that it has limited capability of capturing texture information.

Our research caters for all of these gaps. It uses deep learning algorithms which do not use the linear filter, rather the ResNet uses the residual connections within the network and this helps with maintaining the information flow throughout the network. And VGG-16, which is a type of convolutional neural network (CNN), uses very small convolutional filters, usually 3x3 and this is how the problem of linear filtering is handled. These powerful deep learning architectures are also designed to deal with a huge dataset with hundreds or thousands of images, so the issue of high processing time, tardiness or scalability is also resolved. Deep learning models learn through data and since the dataset that is being used consists of a variety of fabric images including plain, patterned, and textured fabric, that's how these models can detect defects on any type of fabric. In this way, the drawback of restriction in capturing detailed texture information is also covered in this research.

3. Materials and Methods

In order to achieve the best results from a deep learning model, it is important that we feed good and relevant samples of images to our model. Then carefully selecting the best suited combination of hyperparameters of your model also plays a vital role for the training of the model. We tweaked some of the hyperparameters of our model like learning rate, number of epochs to check where our model performs its best. Then testing your model after training on unseen data is necessary to see how your model performs in a real-life situation where the data is similar to the training data but is not exactly the same.

3.1 *Dataset*

Having a good dataset is the key to achieving the optimal results in a deep learning model. In our dataset, we included a variety of sample images including varied patterns, textured and plain fabric images. The dataset is divided into three classes of defects which are vertical defects, horizontal defects and holes. Vertical and horizontal defects are those in which the fabric has a stretch mark in form of a line on top or bottom and in the left or right (respectively) of the fabric while holes are those defects in which the fabric has a circle or oval shaped flaw anywhere on the fabric.

All the images including the training and testing images have been converted to grey-scale. The benefit of working with monochrome images is that it reduces the number of colour channels which in return reduces the computation power required to pre-process the data. Less number of channels means lesser number of parameters our model needs to take into account for classification. During pre-processing, all the images were resized to 224 x 224 pixels. The dataset comprises 3630 images in total which are split into training and testing samples in a 80:20 ratio, where the 80% of our dataset is reserved for training purposes and the rest 20% is used for testing our model.

The dataset is curated from three different sources, first is One-Shot Dataset for Fabric Detection shown in figure (Figure 1), second is Fabric Defect Detection dataset from Kaggle [12] shown in figure (Figure 2), and the third is Tilda Textile Texture Database [13] which is shown in figure (Figure 3). Some data samples from all of these datasets are given below:

One-shot dataset:

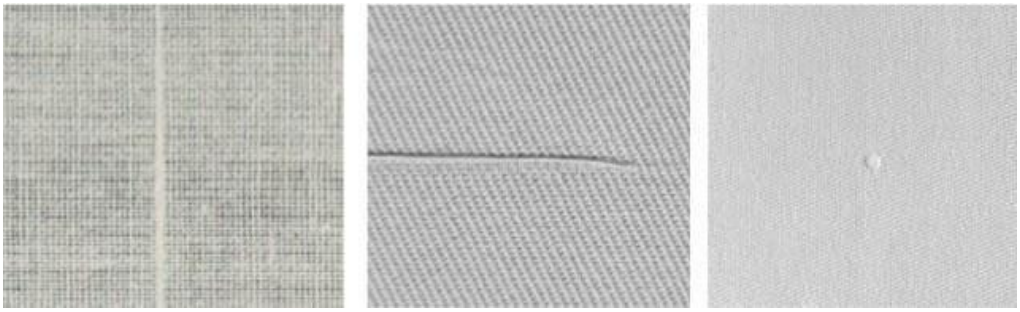

Figure 1. One-shot Fabric Defect Dataset

Fabric defect dataset:

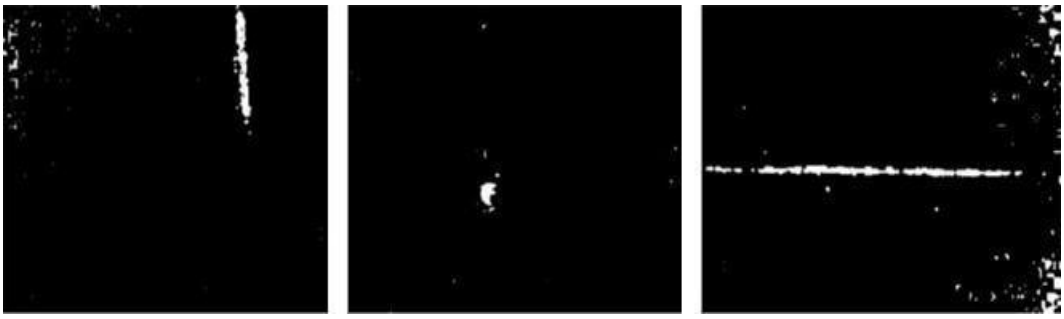

Figure 2. Kaggle Fabric Defect Dataset

Tilda:

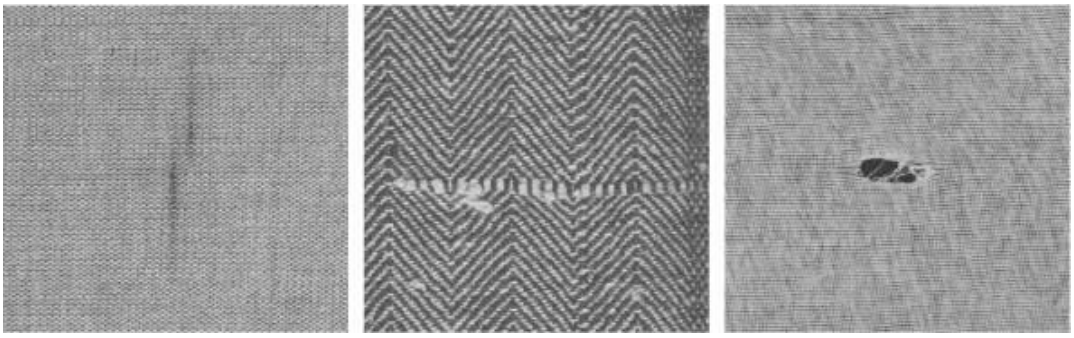

Figure 3. Tilda Fabric Defect Dataset

The types of images present in the dataset are plain, textured and patterned images. Plain fabric is a material with minimal patterning and usually solid or tonal in colour. Patterned fabrics are the ones that have a repeating of an element or motif used to create a unique decoration on fabric. Texture refers to the feel and appearance of fabric, encompassing qualities such as roughness or smoothness, coarseness or fineness, softness or stiffness etc. and these textures can be different on each fabric. The image samples of these fabric types are displayed in the image below (Figure 4).

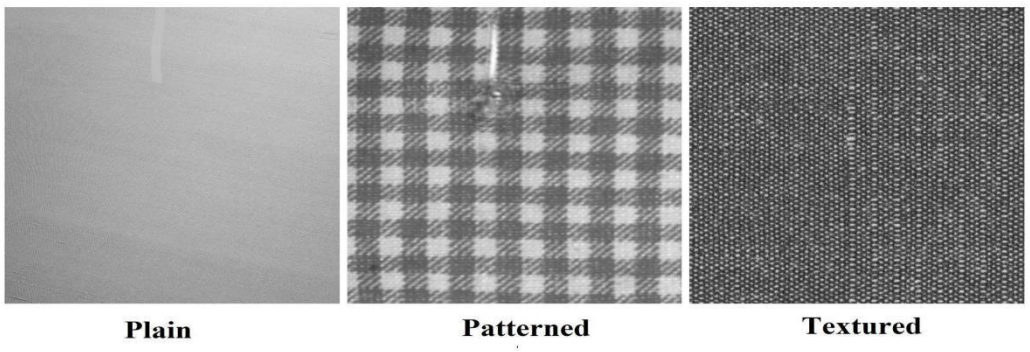

Figure 4. Plain, Patterned and Textured images

3.2 *ResNet*

A Residual Neural Network (ResNet) is a type of deep learning model where the weight layers are trained to learn residual functions with respect to the input of each layer. ResNets effectively address the well-known issue of vanishing gradients. This problem occurs in deep neural networks when, during back propagation, gradients become extremely small after multiple applications of the chain rule, leading to weight values not updating and hindering learning [14]. ResNets counter this by allowing gradients to flow directly through skip connections, which enables them to propagate backward from later layers to earlier filters, preventing the

vanishing gradient problem and facilitating learning [15]. A pre-trained ResNet model is used by TensorFlow. In the below figure (Figure 5), the summary of layers in ResNet model is shown:

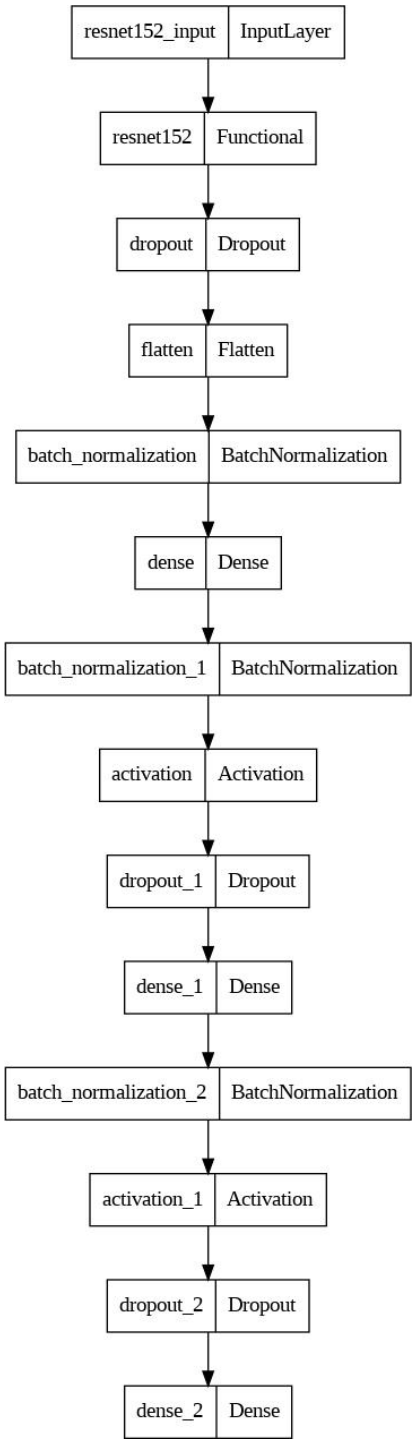

Figure 5. ResNet Layers Summary

Before training our model we first pre-processed our dataset. All images underwent resizing to dimensions of 224 x 224 pixels, and a batch size of 10 was chosen. Given that there were multiple classes to predict, the class mode is configured as "sparse." Our dataset pre-processing began with the utilisation of Keras ImageDataGenerator [16]. This tool was employed to transform the data based on the specified parameters. All images underwent resizing to dimensions of 224 x 224 pixels, and a batch size of 10 was chosen.

The total parameters of our model are 266,411,910 out of which 207,834,118 are the trainable parameters and 58,577,792 are non-trainable parameters. The ResNet architecture comprises an initial convolution and pooling step, succeeded by four consecutive layers that exhibit similar behaviour. Each of these layers adheres

to an identical pattern. They carry out a 3x3 convolution with a consistent feature map dimension of [64, 128, 256, 512] respectively. Additionally, they skip the input every two convolution operations. Notably, the width and height remain unchanged throughout the layer notably, the width and height remain unchanged throughout the layer. The optimiser used in this model is 'Adam' optimizer and the activation function used is 'ReLU'. The 'learning rate' chosen for this model is 0.001, the number of epochs is 50 and the output size is 3. In the following image (Figure 6), ResNet architecture can be seen.

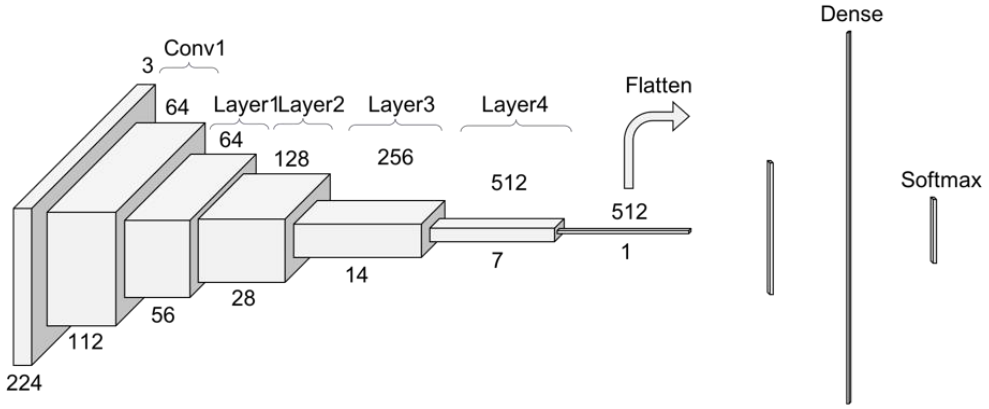

Figure 6. ResNet Architecture Diagram

3.3 *VGG-16*

Visual Geometry Group (VGG) is 16 layers in depth. This model is also based on the Convolutional Neural Network (CNN). A convolutional neural network consists of an input layer, an output layer, and several hidden layers in between. This model comprises a total 13 convolutional layers, 5 max pooling layers, and 3 dense layers, adding up to 21 layers in terms of architecture. However, it has only 16 weight layers, meaning there are 16 layers with learnable parameters. VGG-16 expects an input tensor with dimensions of 224 pixels in height, 224 pixels in width, and 3 channels for RGB colour information. One of the distinctive features of VGG16 is its emphasis on simplicity and uniformity in architectural design. Instead of employing a multitude of hyperparameters, VGG-16 predominantly utilises 3x3 convolution layers with a stride of 1, consistently applying the same padding technique. Additionally, it employs max pooling layers with 2x2 filters and a stride of 2. This consistent and straightforward design is a notable characteristic of VGG16. This model's design adheres to a uniform arrangement of convolutional and max- pooling layers across its entire architecture. Following the stack of convolutional layers in VGG16, there are three Fully-Connected (FC) layers. The final layer is the softmax layer.

We pre-processed the data by resizing it into 224 x 224 pixels. A little scaling is applied between the range of 0.8 and 1.0. Then image normalisation was applied. Image normalisation is a crucial process that ensures accurate comparisons across various data acquisition methods and different instances of textures. Normalising pixel values is particularly advised for imaging modalities that don't directly correspond to absolute physical quantities. After that, we have used a pre-trained model by PyTorch [17] which had 135,310,918 total parameters out of which 1,050,374 are trainable while 134,260,544 are non-trainable parameters. The batch size was set to 128, Learning rate to 0.005, dropout rate to 0.4 [18] and the output size is 3. The summary of our VGG-16 model can be seen in the figure (Figure 7) and the architecture diagram of the VGG-16 model is displayed in (Figure 8).

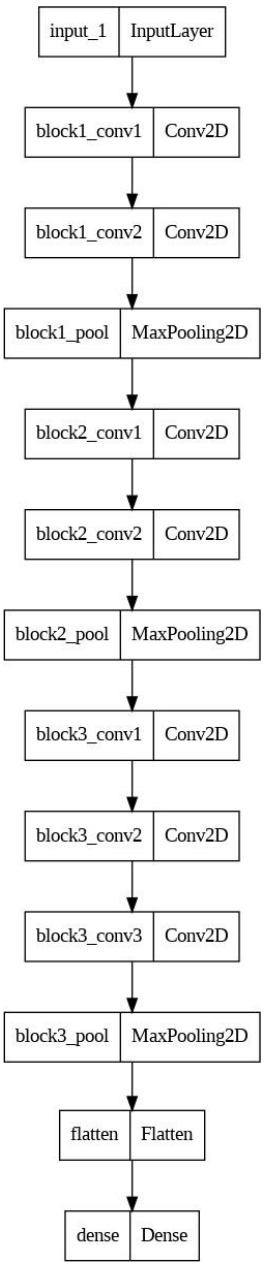

Figure 7. VGG-16 Layers Summary

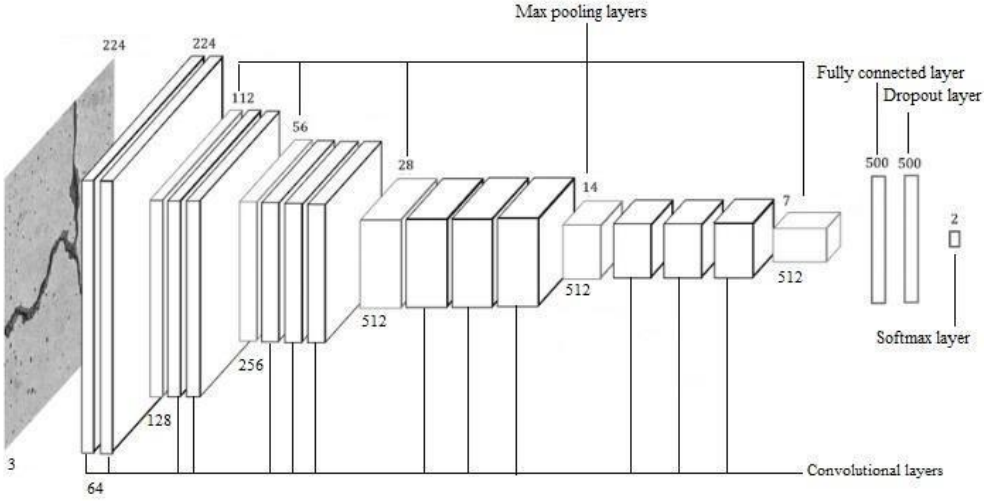

Figure 8. VGG-16 architecture

4. Results and Discussion

ResNet is a very complex deep learning model. It requires high computational resources and has a large number of parameters to train. That's why it requires a large number of data samples categorised in a large number of classes. If the dataset is large and varied enough, then the ResNet does a great job in identifying patterns and classifying the images. The accuracy achieved from the ResNet model is 67.59%. The model was executed for a total of 50 epochs. The graph of loss recorded during training and testing of the model can be seen in figure (Figure 9). Likewise, the training and testing accuracies over the training and testing period of the ResNet model can be seen in figure (Figure 10).

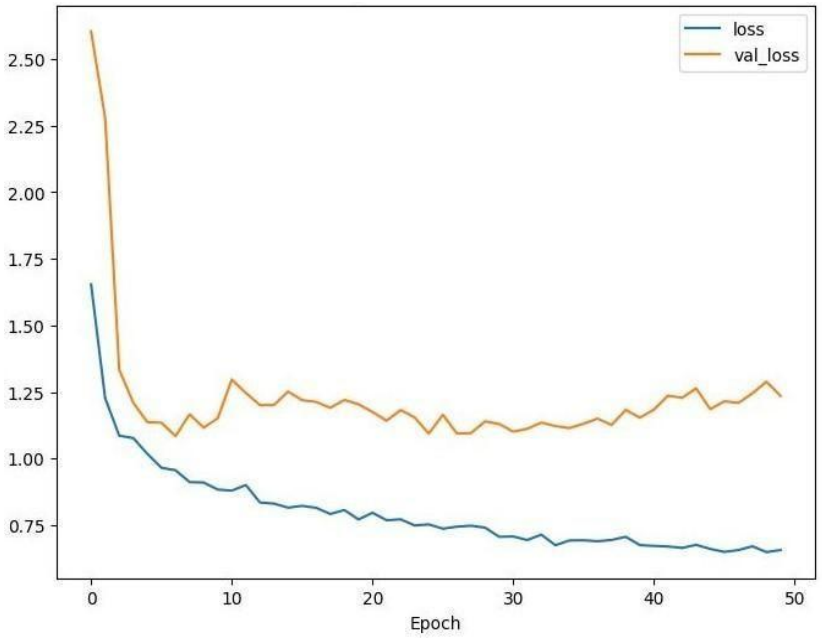

Figure 9. ResNet Training and Testing Loss

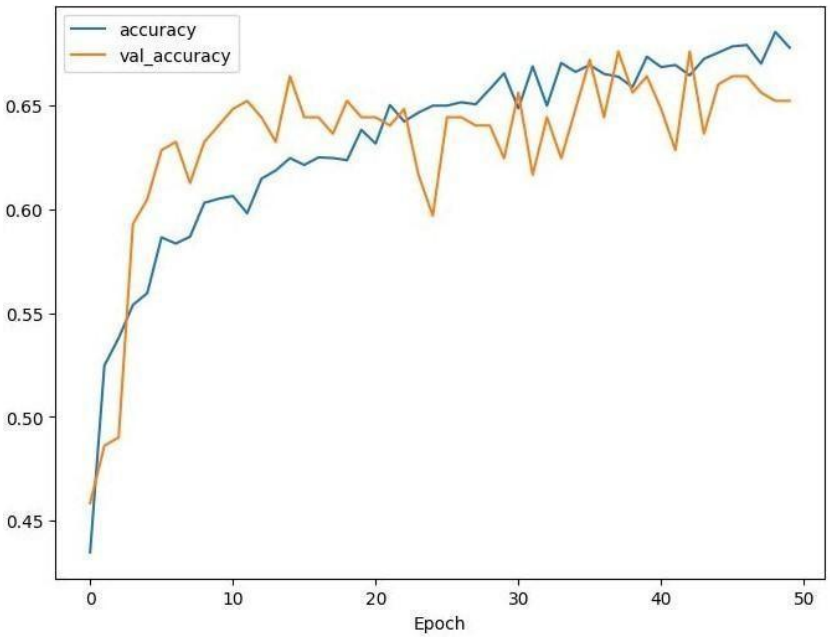

Figure 10. ResNet Training and Testing Accuracy

The graph of loss recorded during the training and testing phase of the VGG-16 model is depicted in figure (Figure 11). Similarly, the training and testing accuracies over the training and testing period of the VGG-16 model can be seen in figure (Figure 12).

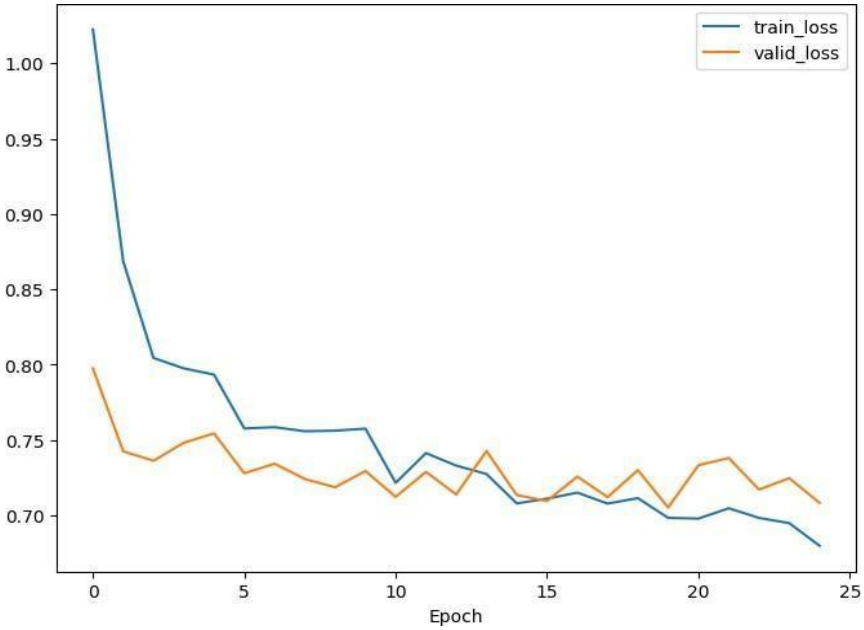

Figure 11. VGG-16 Training and Testing Loss

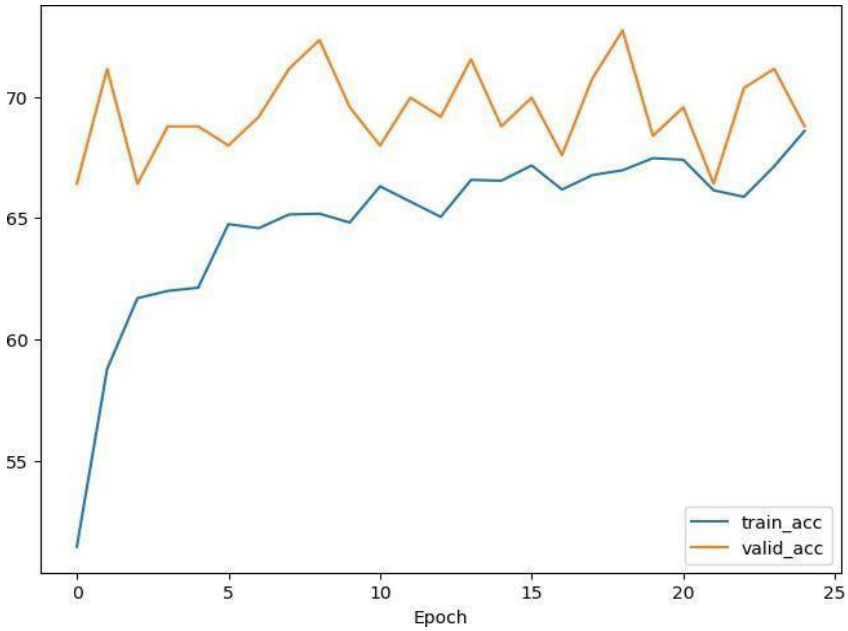

Figure 12. VGG-16 Training and Testing Accuracy

VGG-16 has more parameters than ResNet and therefore is a deeper architecture and it has shown better performance than ResNet with an overall accuracy of 73.91% achieved at 13th epoch. The model was executed for a total of 30 epochs. This transfer learning model re-used the knowledge of pre-trained weights and therefore, lesser parameters were trained due to which, the accuracy achieved was higher than ResNet. Transfer learning made the model size substantially smaller, because it used global average pooling instead of the fully connected layers. Since the weights are not updated and the knowledge is transferred throughout the model, the use of global average pooling is a smarter option than fully connected layers. This gives the model an opportunity to effortlessly flow the knowledge through the model.

The comparison of accuracy, precision, recall, and F1 Score of both the models can be seen in the below table (Table 1).

Table 1. Result comparison of models

Model	ResNet	VGG-16
Accuracy	67.59 %	73.91 %
Precision	0.26	0.64
Recall	0.27	0.65
F1 Score	0.26	0.64

5. Conclusion

Artificial Intelligence finds applications in a wide range of industries, and it has demonstrated impressive achievements in the textile sector. Computer Vision (CV) is harnessed to spot imperfections on fabric surfaces by recognizing patterns and singling out any unusual designs, deeming them as defects. These defects may take the form of horizontal or vertical flaws, or even holes. This research showcases the advantages of employing various deep learning methods for the identification of defects on fabric surfaces. Additionally, the study categorises these defects into different types, such as horizontal, vertical, and holes. The accuracy obtained from ResNet was 67.59% while the accuracy obtained from VGG-16 was 73.91% which was higher than ResNet.

6. Limitations and Future Work

In future, we would like to employ better deep learning techniques like MobileNet, VGG-19 etc. so that better performance results can be achieved. Right now this research is mainly on image based dataset, in future we would like to extend the work to videos as well so that real-time fabric defect detection can be made possible. This study is also limited to one defect per image, further enhancements can be made which detect multiple types of defects in an image if there are any.

Acknowledgments

This acknowledgment serves as a tribute to everyone who contributed to the meaningfulness of my research journey.

Conflict of Interest

There is no conflict of interest for this study.

References

[1] Karayiannis, Y.A.; Stojanovic, R.; Mitropoulos, P.; Koulamas, C.; Stouraitis, T.; Koubias, S.; Papadopoulos, G. Defect detection and classification on web textile fabric using multiresolution decomposition and neural networks. In Proceedings of the IEEE International Conference on Electronics, Circuits and Systems, Pafos, Cyprus, 5–8 September 1999, https://doi.org/10.1109/ICECS.1999.813221.

[2] Chan, C.-H.; Pang, G. Fabric defect detection by Fourier analysis. *IEEE Trans. Ind. Appl.* **2000**, *36*, 1267–1276, https://doi.org/10.1109/28.871274.

[3] Jain, A.K.; Tuceryan, M. *The Handbook of Pattern Recognition and Computer Vision*. Chapter 11, Texture analysis. World Scientific Publishing: Singapore, 1992.

[4] Chellappa, R.; Manjunath, B.S. Texture classification and segmentation: tribulations, triumphs and tributes. In *Foundations of Image Understanding*, pp. 219–240, Springer: Boston, MA, USA, 2001. https://doi.org/10.1007/978-1-4615-1529-6_8.

[5] Zhou, H.; Chen, Y.; Troendle, D.; Jang, B. One-Class Model for Fabric Defect Detection. *arXiv preprint*, **2022**, arXiv:2204.09648, https://doi.org/10.48550/arXiv.2204.09648.

[6] Tong, L.; Wong, W.; Kwong, C. Differential evolution-based optimal Gabor filter model for fabric inspection. *Neurocomputing* **2016**, *173*, 1386–1401, https://doi.org/10.1016/j.neucom.2015.09.011.

[7] Mak, K.-L.; Peng, P.; Yiu, K.-F.C. Fabric defect detection using multi-level tuned-matched Gabor filters. *J. Ind. Manag. Optim.* **2012**, *8*, 325–341, https://doi.org/10.3934/jimo.2012.8.325.

[8] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110, https://doi.org/10.1023/b:visi.0000029664.99615.94.

[9] Raheja, J.L.; Ajay, B.; Chaudhary, A. Real time fabric defect detection system on an embedded DSP platform. *Optik* **2013**, *124*, 5280–5284, https://doi.org/10.1016/j.ijleo.2013.03.038.

[10] Kula, J.; Tunak, M.; Linka, A. Real-time quality control of fabric based on multivariate control charts. In Proceedings of the 7th international conference TEXSCI 2010, Liberec, Czech Republic, 6–8 September 2010.

[11] Ranathunga, S. Fabric Defect Dataset. Kaggle. Available online: https://www.kaggle.com/datasets/rmshashi/fabric-defect-dataset (accessed on 24 December 2023).

[12] Ben Salem, Y.; Abdelkrim, M.N. Texture classification of fabric defects using machine learning. *Int. J. Electr. Comput. Eng. (IJECE)* **2020**, *10*, 4390–4399, https://doi.org/10.11591/ijece.v10i4.pp4390-4399.

[13] Targ, S.; Almeida, D.; Lyman, K. Resnet in resnet: Generalizing residual architectures. *arXiv preprint*, **2016**, arXiv:1603.08029, https://doi.org/10.48550/arXiv.1603.08029.

[14] He, F.; Liu, T.; Tao, D. Why ResNet Works? Residuals Generalize. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *31*, 5349–5362, https://doi.org/10.1109/tnnls.2020.2966319.

[15] Gulli, A.; Pal, S. *Deep learning with Keras*. Packt Publishing Ltd: Birmingham, UK, 2017.

[16] Tammina, S. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *Int. J. Sci. Res. Publ. (IJSRP)* **2019**, *9*, 143–150, https://doi.org/10.29322/ijsrp.9.10.2019.p9420.

[17] Arshad, S.R.; Obaid, I.; Gull, R.; Shahzad, M.K. Steel Defect Classification Using Machine Learning. In Proceedings of 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, South Korea, 3–5 January 2022, https://doi.org/10.1109/IMCOM53663.2022.9721728.

[18] Israr, H.; Khan, S.A.; Tahir, M.A.; Shahzad, M.K.; Ahmad, M.; Zain, J.M. Neural Machine Translation Models with Attention-Based Dropout Layer. *Comput. Mater. Contin.* **2023**, *75*, 2981–3009, https://doi.org/10.32604/cmc.2023.035814.