

# FN-Net: A Lightweight CNN-based Architecture for Fabric Defect Detection with Adaptive Threshold-based Class Determination

Anindita Suryarasmia, Chin-Chun Chang<sup>b,1</sup>, Rania Akhmalia<sup>a</sup>, Maysa Marshalia<sup>a</sup>,  
Wei-Jen Wang<sup>a</sup>, Deron Liang<sup>a</sup>

<sup>a</sup>National Central University, No. 300, Zhongda Rd, Zhongli, Taiwan

<sup>b</sup>National Taiwan Ocean University, No. 2, Beining Rd, Keelung, Taiwan

## Abstract

Deep learning technologies based on Convolution Neural Networks (CNN) have been widely used in fabric defect detection. On-site CNN model training and defect detection offer several desirable properties for the fabric manufactures, such as better data security and less connectivity requirements, when compared with the on-cloud training approach. However, computers installed at the manufacturing site are usually industrial computers with limited computing power, which are not able to run many effective CNN models. A lightweight CNN model should be used in this scenario, in order to find a balance point among defect detection, efficiency, memory consumption and model training time. This paper presents a lightweight CNN-based architecture for fabric defect detection. Compared with VGG16, MobileNetV2, EfficientNet, and DenseNet as state-of-the-art architectures, the proposed architecture, namely FN-Net, can perform training 3 to 33 times as fast as these architectures with less graphics processing unit and memory consumption. With adaptive class determination, FN-Net has an average F1 score 0.86, while VGG16 and EfficientNet as the best and the worst among the baseline models have 0.81 and 0.50, respectively.

---

<sup>1</sup> Corresponding author

*Keywords:* artificial intelligence, lightweight Convolutional Neural Network, fabric manufacturing, AOI, defect detection

## 1. Introduction

Automatic fabric defect detection is critical in textile production for the ability to distinguish defective and normal fabrics (Fig. 1) in the production quality control. It reduces the risk of revenue loss due to the sale of defective fabrics [1]. Traditionally, the defect detection process is performed by professional inspectors who examine the surface of the fabrics manually. This process can be time-consuming, prone to human error, and leads to results with low reliability and stability [2], [3], [4]. Consequently, the demands for automatic fabric defect detection through image and video processing technologies have been increasing. Numerous approaches, including structural, statistical, spectral, and model-based approaches, have been proposed for automatic fabric defect detection to improve fabric quality and reduce labor costs [1], [5].

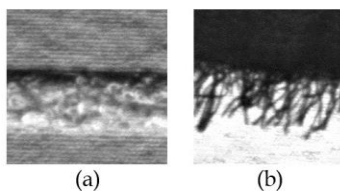


Figure 1: (a) Sample of a defective fabric and (b) sample of a normal fabric with worn-out threads.

Rule-based Automated Optical Inspection (AOI) systems have been developed and installed in the fabric production line for automatic fabric defect detection. In this AOI system, several cameras were installed to capture image samples. The images are sent to the industrial computer for defect detection. However, several types of fabric conditions are misidentified by the AOI system. Those images are considered as hard-case images, since the easy ones have been classified by the AOI system. Consequently, the misclassified images must be manually examined by the human inspector to determine whether they represent fabric defects. In addition, the on-site installation is preferred to avoid privacy leakage issue if the training is performed on cloud [6]. To address these challenges, an advanced

image processing approach, such as lightweight deep learning methods, must be developed.

Deep learning is considered the best technique in many domains for identifying complex structures in high-dimensional data [7]. Among deep learning models, CNNs are the most suitable for image recognition, classification, and detection [8]. Fabric defect detection and classification have been conducted by fine-tuning existing CNN architectures, constructing new CNN architectures, and combining two or more CNN architectures to enhance the results [2], [9], [10], [11], [12], [13].

Due to their advantages, well-known deep CNN architectures are modified to achieve satisfactory image-based defect detection results in fabric manufacturing or other domains [2], [4], [12], [13]. Various studies have implemented CNN-based architectures for fabric defect identification. Zhao et al. [2] constructed a model by combining visual perceptron, visual long-term memory, and visual short-term memory to classify defective fabrics. They achieved an accuracy of 99.47% for the DHU-DF-500 dataset. Liu et al. [4] developed an effective weakly supervised shallow network to localize the defective area of fabric with F1 score of 0.9346. Liu et al. [12] optimized VGG16 to identify defective fabrics from the Xiamen Face++ dataset, achieving an accuracy of 98.1%. Perez et al. [13] used pre-trained VGG16 with object localization for automated detection of defects and deterioration of buildings with success rate of around 91%. Table 1 provides summary of the Deep CNN recent related studies.

Nevertheless, deep CNNs have certain disadvantages, such as their large file sizes, numerous parameters, and high computational costs. Therefore, these networks are unsuitable to be run on devices with limited computing power [14], [15], [16], [17]. Consequently, lightweight CNN architectures have been constructed to run image processing tasks, such as semantic segmentation and image classification, efficiently on mobile devices [18], [19], [20]. They successfully reduce the complexity and computational cost of the models with slight trade-off on the accuracies and error rates. Lightweight CNN architectures

use relatively few parameters, which enables them to be run faster. Moreover, the relatively small model size of these architectures enables them to be executed on

Table 1: Summary of Deep CNN Recent Related Studies

	Architecture name	Dataset	Methodology
Zhao et al. [2]	VLSTM	DHU-FD-500, DHU-FD-1000 and Aliyun-FD-10500 fabric dataset	Build visual long-short-term based deep learning for fabric defect detection.
Liu et al. [4]	DLSE-Net	DAGM2007 and private fabric dataset	Build a weakly supervised shallow network to localize the defective area of fabric.
Liu et al. [12]	LZFNet	private fabric dataset	Fine-tuned VGG16 for optimized fabric defect detection.
Perez et al. [13]	CNN-CAM	private buildings dataset	Fine-tuned VGG16 paired with object localization for buildings defect detection.

hardware with limited computing power, such as mobile devices [20], [14], [16], [17]. Several studies have constructed lightweight CNN architectures for image classification and segmentation in various domains. Yu et al. [14] developed Bi-MobileNet for remote sensing image classification. This network achieved an overall accuracy of 94.08% for the NWPU-RESISC45 dataset with a training data ratio of 20%, 7.76 million parameters, and 29.59 MB model size. Jian et al. [16] constructed a lightweight CNN model for fingerprint classification. This model achieved an accuracy of 93% with 816,261 parameters. Wang et al. [17] developed Light-AMC, which has 1.3 MB model size, for automatic modulation classification. Li et al. [20] optimized MobileNet to detect surface defects in chili filling production. They achieved an accuracy of 95.00% with a training time less than 1 day and a detection time of 0.12 seconds. Table 2 provides summary of the lightweight recent related studies.

As an addition to the lightweight CNN's accuracy compensation, imbalanced

Table 2: Summary of Lightweight CNN Recent Related Studies

	Architecture name	Dataset	Methodology
Yu et al. [14]	BiMobileNet	UCMerced, AID, and NWPU-RESISC45 dataset	MobileNetV2 paired with bilinear model for efficient and lightweight remote sensing image classification.
Jian et al. [16]	Jian et al. [16]	NIST SD4 fingerprint dataset	A lightweight CNN structure based on singularity ROI for fingerprint recognition.
Li et al. [21]	MobileNet- SSD	Private industrial chili filling image dataset	Optimize MobileNet combined with Single Shot Multibox Detector (SSD) network for surface defect detection.

datasets can also negatively affect the performance of learning algorithms [22], [23]. Various approaches have been proposed to overcome this, such as creating samples within clusters, oversampling the minority class, and using alternative metrics such as the F1 score, Area Under Curve (AUC) score, and G-mean [7], [22], [23], [24], [4]. Among these metrics, the F1 score is commonly used to assess models' performance by using the numbers of True Positives (TPs), False Positives (FPs), and False Negatives (FNs).

It is feasible to apply the state-of-the-art CNN architectures such as VGG16 [25], DenseNet [20], MobileNetV2 [18], and EfficientNet [19] for fabric defect detection. However, those architectures are usually too complex to be run and trained on an industrial computer with limited computing resources [12]. Therefore, this paper presents FN-Net, a simple lightweight CNN architecture for image-based fabric defect detection. This architecture has a low number of parameters, which leads to low computing resource demands. Furthermore, adaptive threshold-based class determination is implemented to reduce the error rate of the trained FN-Net for the defective images. This strategy allows the maximum False Negative Rate (FNR) to be set to a certain value while minimizing the False Positive Rate (FPR) by adjusting the threshold value in the class

determination process. Compared with VGG16, DenseNet, MobileNetV2, and EfficientNet, FN-Net performs training 3 to 33 times as fast as these architectures with less graphics processing unit and memory consumption. With adaptive class determination, FN-Net has an average F1 score 0.86, while VGG16 and EfficientNet as the best and the worst have 0.81 and 0.50, respectively.

The remainder of this paper is organized as follows. Section 2 specifies the datasets used in this study; Section 3 presents the proposed approach for fabric defect detection; Section 4 describes the experiments conducted in this study; and Section 5 exhibits the summary of the proposed method and the conclusions.

## **2. Datasets**

Two sets of Greige fabrics images from a fabric company in Taiwan are used in this study. The images have gone through a rule-based AOI and misclassified as defective. Each fabric image was acquired by capturing an area of a fabric roll by using cameras set up under two lighting scenarios: reflected light and transmitted light. Three cameras were installed for each lighting scenario. Cameras 1–3 were used for reflected light imaging, and cameras 4–6 were used for transmitted light imaging. The aforementioned six cameras were located inside glass boxes marked with red and yellow lines (Fig. 2). In the reflected light scenario, the light sources and cameras were located above the fabric roll, which allowed light to be projected approximately at the same angles as the cameras. In the transmitted light scenario, the cameras were located above the fabric roll, with the light sources were located under the fabric roll, which allowed light to be projected through the fabric. The distributions and samples of both datasets are presented in TABLE 3 and TABLE 4. Both datasets contain different type of defect and normal classes. Additionally, both datasets suffer from imbalance condition in a different way. The majority samples of Dataset 1 belong to normal class, which make up to 99.75% of total samples, while in Dataset 2, the majority class is the defect class with 91.63% of total samples.

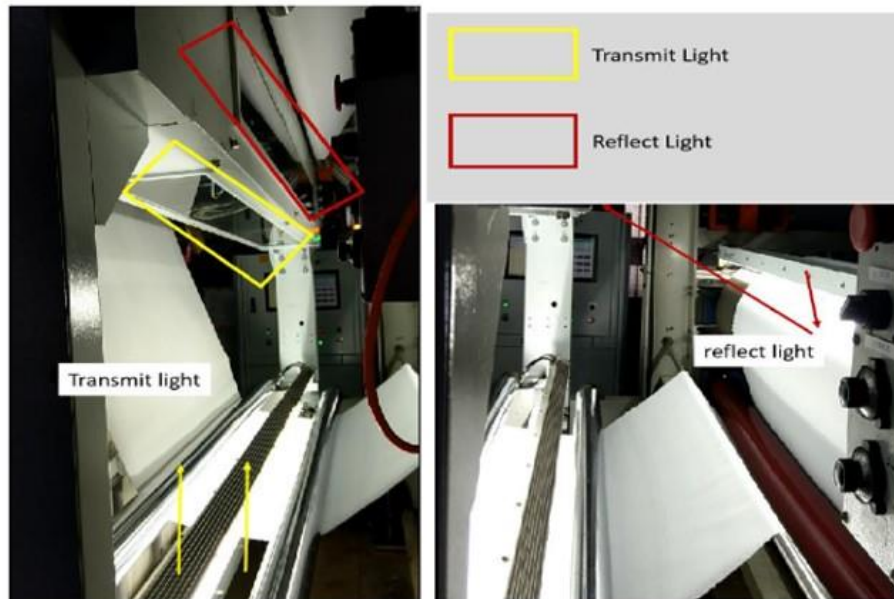


Figure 2: Fabric images captured using two lighting scenarios: transmitted light and reflected light. Three cameras were used in each lighting scenario to capture fabric images.

The typical defect detection system requires one defect class and one normal class as the input. However, since each normal classes have distinctive features, and they came from AOI system's false positive detection, they are easy to be recognized as defective. Merging them into one normal class would result in the loss of potentially useful information for classification. Therefore, a multiclass classification was performed by retaining multiple normal classes and one defect class. An additional procedure was required after multiclass classification to ensure that the final classification was a binary classification.

All image samples are greyscale images with pixel values between 0 and 255, and various sizes around  $128 \times 128$  pixels (e.g.,  $130 \times 130$  and  $132 \times 135$  pixels). Before being input into the proposed network, the images were preprocessed by being cropped to  $128 \times 128$  pixels and the values were normalized between 0 and 1.

Table 3: Image Classes and Samples in Dataset 1

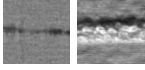
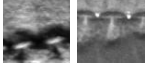

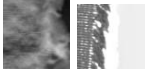
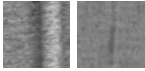
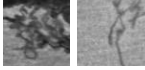
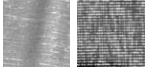

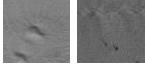
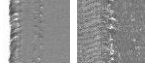
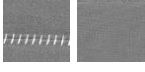
Classes	Number of images	Image samples
Defect	183	
Normal - Seam	1,685	
Normal - Dirt	32,621	
Normal - Fringe	1,221	
Normal - Fold	10,602	
Normal - Thread-off	3,187	
Normal - White-spot	23,159	
Total number of defect images	179	0.25%
Total number of normal images	72,475	99.75%
Total number of images	72,654	

Table 4: Image Classes and Samples in Dataset 2

Classes	Number of images	Image samples
Defect	11,502	
Normal - Uneven cloth	19	
Normal - Selvage	943	
Normal - Seam	88	
Total number of defect images	11,502	91.63%
Total number of normal images	1,050	8.37%
Total number of images	12,552	



### 3. Proposed Architecture and Data Processing Workflow

This paper proposes FN-Net, a CNN-based architecture, for fabric defect detection. FN-Net has a low computational cost because it uses relatively few parameters; thus, this network has a short runtime and can be executed on hardware devices with relatively low computing power. However, FN-Net might have a low defect detection performance due to its lightness. Moreover, the presence of unbalanced data between normal and defect classes might further reduce the performance. To overcome these problems, an adaptive threshold-based class determination is conducted on the probability values generated by FN-Net. By implementing these methods, a lightweight CNN-based architecture can be constructed without compromising the detection accuracy.

#### 3.1. Architecture of FN-Net

FN-Net receives  $128 \times 128$ -pixel grayscale images as inputs and produces multiclass outputs, which are then converted into binary outputs (defect and normal). Fig. 3 displays the architecture of FN-Net. It consists of four convolutional layers with 16, 32, 64, and 96 channels. Each convolutional layer has a  $3 \times 3$  receptive field with zero padding, a stride of 1, and a Rectified Linear Unit (ReLU) activation function. The stride and padding values are selected to preserve the spatial size before max pooling is performed, and the ReLU activation is used to enable the network to learn faster [26]. Each convolutional layer in the proposed architecture is followed by a max pooling layer. The first two max pooling layers have windows of  $3 \times 3$  with a stride of 3, and the final two layers have windows of  $2 \times 2$  with a stride of 2. The windows and stride values are selected such that the spatial size of the last max pooling layer is  $3 \times 3$ . To visualize the ability of FN-Net's convolution layers to locate the defective areas of fabrics, gradient-based localization [27] is performed for the activation layer following the final convolutional layer. The heat maps of the localization (Fig. 4) indicate the defective areas of a fabric. This information is used by the following layers to determine whether an image can be considered to represent a defective fabric.

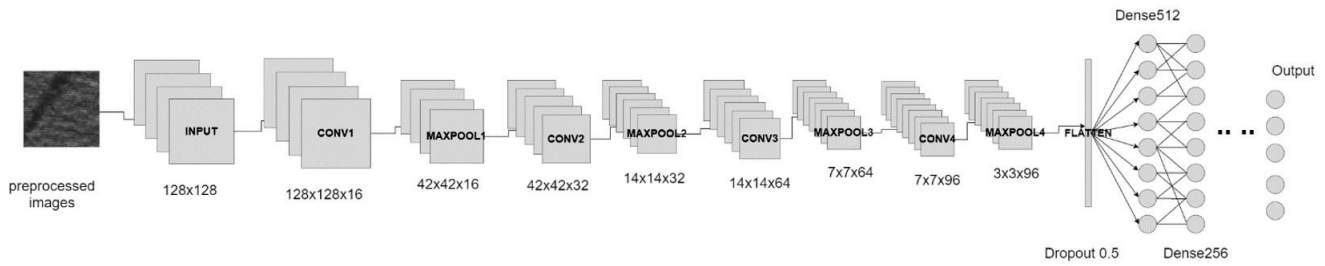


Figure 3: Architecture of FN-Net, which consists of four convolution layers, four max pooling layers, and three dense layers. The final dense layer acts as an output layer.

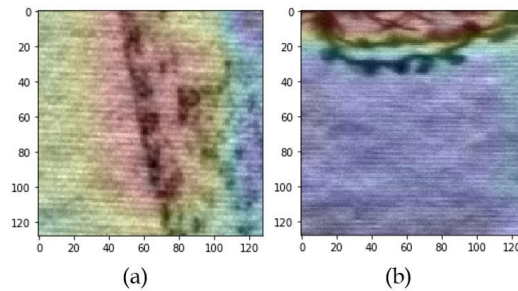


Figure 4: Heat maps of (a) Image number 1,000 and (b) Image number 11,000, which indicate the ability of FN-Net to detect the defective areas of fabrics.

The stacks of convolutional and max pooling layers are followed by two fully connected layers (i.e., dense layers) and an output layer. The first fully connected layer derives the layers' weights by flattening the nodes of the final max pooling layer with a dropout of 0.5. This value means that half the nodes are randomly selected and ignored during training to prevent overfitting. The first and second fully connected layers possess 512 and 256 channels, respectively. Finally, the output layer is responsible for performing classification. The number of channels in the output layer depends on the number of datasets' image classes. With such configurations, FN-Net uses considerably fewer parameters than other architectures as indicated in Table 5.

The output layer of FN-Net uses a softmax activation function to provide the probabilities of each class (ranging from 0 to 1), with the sum of the probabilities of all classes being 1. The probability values are commonly used to determine the class of a given input.

Table 5: Number of Parameters Used by Different Architectures

Architecture	Number of parameters
VGG16	63,989,365
MobileNetV2	2,401,351
EfficientNet	4,058,538
DenseNet	7,152,199
FN-Net	623,975

Because the dataset comprises multiple normal classes and only one defect class as the output, the probabilities of the normal classes are combined to get one probability for each defect and normal class. Furthermore, the imbalanced nature of the dataset allows the prediction accuracy to be high even if all the minority samples are misclassified. For example, if all the defect images in Dataset 1 are misclassified as normal images, the classification accuracy is still 99.75%. Therefore, an additional data processing must be conducted to interpret the CNN probability values for increasing the prediction quality without CNN retraining and F1 score should be used instead of accuracy to evaluate the proposed architecture.

### 3.2. Class Determination with Adaptive Threshold

Typically, the threshold values are the averages of adjacent probability values for a given set of input images. The number of thresholds and their values can vary between different prediction results. To avoid this situation, a set of threshold values ranging from 0 to 1 in increments of 0.0001 is used. The results of the comparisons between the threshold and defect probabilities are then used to calculate the numbers of TPs, True Negatives (TNs), FPs, and FNs. In this paper, defective images are labeled as positive and normal images are labeled as negative.

Since both datasets suffer from imbalance problem, the proposed model tends to minimize either FPR or FNR even if the other is significantly higher so that a good prediction accuracy can be achieved. To prevent this situation, usually, Equal

Error Rate (EER) point, where the FPR is equal or close to the FNR, can be used as a solution. However, using similar FPR and FNR values may not be the optimal approach, especially in the case of an extremely imbalanced dataset. As an alternative, a maximum FNR can be defined according to company's requirements, and the solution will be found by finding the minimum FPR value. However, if the maximum FNR value is not specified, the solution can be selected based on the best F1 score from various FNR values. The steps to perform the adaptive threshold approach is listed as follows:

1. Set initial threshold values with 0 to 1 in increments of 0.0001.
2. Compare validation's data defect probabilities with each threshold value.
3. Calculate F1 scores for each threshold value.
4. Select threshold that returns the specified FNR value or highest F1 score.
5. Apply threshold to the testing data.

#### **4. Experiment Results and Discussion**

Two experiments for two different datasets were conducted to demonstrate the lightweightness and defect detection performance of FN-Net by comparing it with VGG16, DenseNet, MobileNetV2, and EfficientNet. These four baseline architectures were implemented using Keras modules in Python 3.7.7 and TensorFlow 2.3.1. We also demonstrated the ability of the adaptive threshold-based class determination to tune the proposed model's performance on the basis of different thresholds.

In each experiment, the datasets were split into 90% and 10% for training and testing, respectively. It ensures the training data not to be included in the testing phase. Additionally, the validation sets were split from the training set and consisted of as much as 10% of the training data. Hyperparameters were selected on the basis of the validation data, and the presented results were obtained from the testing data. Each experiment was conducted 5 times and the average results were considered.

This section is divided into two parts. Section 4.1 presents runtime and resource consumption results of FN-Net and the baseline CNN architectures. Section 4.2 presents the results of adaptive data processing and the thresholds selection according to the F1 scores.

To compare the five considered architectures, their F1 scores, along with the corresponding accuracies and FNR were determined as shown in (1 to 3). By applying multiple threshold values to the prediction results of validation data, various FNR values were obtained for the calculation of the F1 score. The highest F1 scores were then selected, and the corresponding thresholds were applied to the testing data to obtain the prediction results.

$$F1 = \frac{TP}{TP+0.5(FP+FN)} \quad (1)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$FNR = \frac{FN}{TP+FN} \quad (3)$$

#### 4.1. Lightweightness Evaluation

The computational cost experiment was conducted to prove that FN-Net can perform efficiently with a low computation cost, short execution time, and small model size. We recorded the Graphics Processing Unit (GPU) utilization, memory, and runtime in both the training and testing phases with Datasets 1 and 2 by using the wandb.ai module [28]. The computing host consisted of an Intel® Core™ i7-900 CPU, an NVIDIA GeForce RTX 2080 Ti GPU, and 64 GB of RAM. The operating system of this host was 64-bit Microsoft Windows 10 Enterprise.

TABLE 6 presents the model sizes of the architectures. It shows that FN-Net is only 25% the size of MobileNetV2, which was the smallest baseline models. As

Table 6: Model Sizes of the Compared Architectures

Architecture	Model size (MB)
VGG16	500
MobileNetV2	28
EfficientNet	48
DenseNet	84
FN-Net	7

for the computational cost, TABLE 7 indicates that DenseNet had the longest training time, while MobileNetV2 had the shortest training time among the baselines. The runtime of FN-Net was 73% and 6% of MobileNetV2 for Datasets 1 and 2, respectively. In the testing phase, FN-Net consistently had the shortest runtime, with an average of 0.17 minutes for both datasets. This runtime was 37% the average runtime of the baseline models.

The highest GPU utilization of the training and testing phase was obtained by VGG16 with average values of 94% and 90%, respectively; while the lowest GPU utilization of baseline architecture was obtained by DenseNet with 76.01% for training and MobileNetV2 with average value of 78.5%. Compared with the DenseNet, FN-Net had less 4% GPU utilization. As for the memory utilization, FN-Net had 78% of EfficientNet as the lowest among the baselines. The results confirmed that the proposed FN-Net was considerably smaller, faster, and required less computing power than the baseline models.

#### 4.2. Evaluation of Class Determination with Adaptive Threshold

An experiment was conducted to demonstrate the suitability of adaptive threshold-based class determination. It allows the selection of certain FNR values and the determination of the corresponding threshold and mean F1 score for evaluating the selected solution. To illustrate the multiple solutions obtained by compared networks, Fig. 5 displays the F1 scores for different FNR values when using validation data from Datasets 1. The highest F1 score for Dataset 1 (0.78) was obtained when the FNR was 0.18 and FPR was 0.00.

Table 7: Runtime, GPU Utilization, and Memory Utilization of the Compared Architectures

	Dataset 1						Dataset 2					
	Training			Testing			Training			Testing		
	Runtime GPU Memory			Runtime GPU Memory			Runtime GPU Memory			Runtime GPU Memory		
	(min.)	(%)	(MB)	(min.)	(%)	(MB)	(min.)	(%)	(MB)	(min.)	(%)	(MB)
VGG16	30.2	94	59.1	1.18	90	74.42	3.9	94	35.93	0.22	90	42.96
MobileNetV2	13.4	93	57.61	0.74	79	60.16	1.97	93	39.3	0.21	78	40.35
EfficientNet	29.4	89	50.96	1.07	81	59.65	4.6	89	42.45	0.31	82	41.51
DenseNet	43.6	87	57.68	1.28	75	58.71	4.06	87	36.5	0.36	86	40.81
FN-Net	9.8	83	43.56	0.26	52	40.63	0.11	83	29.48	0.08	42	36.53

The preferred solutions in this paper are presented by selecting the best F1 scores points (marked with x) within the graphs displayed in Fig. 5. Nevertheless, any points in the graph are feasible solutions that can be selected according to the company’s requirements as described in Section 3.2.

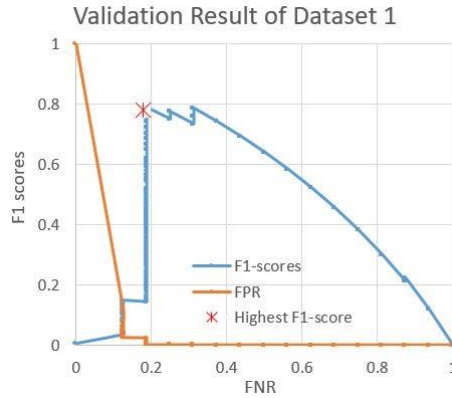


Figure 5: Validation results of FN-Net for Dataset 1 at various FNR values. The best F1 scores (red crosses) are the commonly used points in threshold selection. However, any points in the graph are feasible solutions.

Table 8: Highest F1 scores Obtained for Various Architectures

	Dataset 1						Dataset 2					
	Highest F1 score			Argmax			Highest F1 score			Argmax		
	F1 scores	Corr. ACC	Corr. FNR	F1 scores	Corr. ACC	Corr. FNR	F1 scores	Corr. ACC	Corr. FNR	F1 scores	Corr. ACC	Corr. FNR
VGG16	0.62	0.99	0.40	0.60	0.99	0.40	0.99	0.99	0	0.99	0.99	0
MobileNetV2	0.05	0.04	0.01	0.00	0.99	1	0.98	0.95	0	0.97	0.96	0
EfficientNet	0.04	0.40	0.38	0.00	0.99	1	0.97	0.95	0	0.97	0.96	0
DenseNet	0.33	0.99	0.68	0.62	0.99	0.44	0.98	0.97	0	0.97	0.96	0.03
FN-Net	0.72	0.99	0.36	0.60	0.99	0.33	0.99	0.99	0	0.99	0.99	0

To demonstrate the effectiveness of the adaptive threshold-based class de termination, the approach was applied to all five architectures, and the results are presented in Table 8. Among the baseline models, the best F1 scores of Dataset 1 was obtained by VGG16 with 0.62, while the worse was EfficientNet with 0.04. Even though the corresponding accuracies of the VGG16, DenseNet, and FN-Net reached 0.99, with lower FNR of 0.36, FN-Net achieved better F1 score by 0.1 compared to VGG16. When the adaptive approach was not applied, the accuracies of all the models reaches optimum value by 0.99. However, MobileNetV2 and EfficientNet return FNR value of 1, which implies that the whole defect samples were mispredicted as normal. As for the Dataset 2, both VGG16 and FN-Net had 0.99 F1 scores with corresponding accuracies of 0.99 and corresponding FNR of 0. The results confirmed that the proposed FN-Net was able to obtain better classification results while maintaining less computing resources than the baseline models.



## 5. Conclusions

This paper presents FN-Net, a lightweight CNN-based architecture for fabric defect detection. This architecture is currently the best solution for overcoming the hardware limitations of on-site industrial computers in fabric manufacturing. Compared with VGG16, MobileNetV2, EfficientNet, and DenseNet, the proposed architecture enables image-based defect detection to be performed with a considerably lower computational cost and significantly higher speed. Moreover, the results of the proposed method are better compared to the aforementioned state-of-the-art lightweight architectures. We demonstrated that the adaptive-threshold-based class determination can dynamically adjust the prediction results of existing trained models without the need for retraining.

## Acknowledgement

This study is conducted under the “Research & Development Project for IntelliGEMt Cloud Service Platform for Machinery Industry” of the Institute for Information Industry which is subsidized by the Ministry of Economic Affairs of the Republic of China.

## References

- [1] K. Hanbay, M. F. Talu, O. F. “Ozgu”ven, “ Fabric defect detection systems and methods—A systematic literature review, *Optik (Stuttg)*. 127 (24) (2016) 11960–11973. doi:10.1016/j.ijleo.2016.09.110.  
URL <http://dx.doi.org/10.1016/j.ijleo.2016.09.110>
- [2] Y. Zhao, K. Hao, H. He, X. Tang, B. Wei, A visual long-short-term memory based integrated CNN model for fabric defect image classification, *Neurocomputing* 380 (2020) 259–270. doi:10.1016/j.neucom.2019.10.067.  
URL <https://doi.org/10.1016/j.neucom.2019.10.067>
- [3] A. Z. da Costa, H. E. Figueroa, J. A. Fracarolli, Computer vision based detection of external defects on tomatoes using deep learning, *Biosyst. Eng.*

190 (2020) 131–144. doi:10.1016/j.biosystemseng.2019.12.003.

URL <https://doi.org/10.1016/j.biosystemseng.2019.12.003>

- [4] Z. Liu, Z. Huo, C. Li, Y. Dong, B. Li, DLSE-Net: A robust weakly supervised network for fabric defect detection, *Displays* 68 (May) (2021) 102008. doi:10.1016/j.displa.2021.102008.

URL <https://doi.org/10.1016/j.displa.2021.102008>

- [5] H. Y. Ngan, G. K. Pang, N. H. Yung, Automated fabric defect detection A review, *Image Vis. Comput.* 29 (7) (2011) 442–458. doi:10.1016/j.imavis.2011.02.002.

URL <http://dx.doi.org/10.1016/j.imavis.2011.02.002>

- [6] G. Shu, W. Liu, X. Zheng, J. Li, IF-CNN: Image-Aware Inference Framework for CNN With the Collaboration of Mobile Devices and Cloud, *IEEE Access* 6 (2018) 68621–68633. doi:10.1109/ACCESS.2018.2880196.

- [7] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, *Nature* 521 (7553) (2015) 436–444. doi:10.1038/nature14539.

- [8] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* 86 (11) (1998) 2278–2323. doi:10.1109/5.726791.

- [9] R. Wang, Q. Guo, S. Lu, C. Zhang, Tire Defect Detection Using Fully Convolutional Network, *IEEE Access* 7 (2019) 43502–43510. doi:10.1109/ACCESS.2019.2908483.

- [10] H. Xie, Y. Zhang, Z. Wu, Fabric Defect Detection Method Combing Image Pyramid and Direction Template, *IEEE Access* 7 (2019) 182320–182334. doi:10.1109/ACCESS.2019.2959880.

- [11] S. Mei, Y. Wang, G. Wen, Automatic fabric defect detection with a multiscale convolutional denoising autoencoder network model, *Sensors (Switzerland)* 18 (4) (2018) 1–19. doi:10.3390/s18041064.

- [12] Z. Liu, C. Zhang, C. Li, S. Ding, Y. Dong, Y. Huang, Fabric defect recognition using optimized neural networks, *J. Eng. Fiber. Fabr.* 14 (41) (2019). doi:10.1177/1558925019897396.
- [13] H. Perez, J. H. Tah, A. Mosavi, Deep learning for detecting building defects using convolutional neural networks, *Sensors (Switzerland)* 19 (16) (2019). doi:10.3390/s19163556.
- [14] D. Yu, Q. Xu, H. Guo, C. Zhao, Y. Lin, D. Li, An efficient and lightweight convolutional neural network for remote sensing image scene classification, *Sensors (Switzerland)* 20 (7) (2020). doi:10.3390/s20071999.
- [15] J. Shen, N. Liu, H. Sun, H. Zhou, Vehicle Detection in Aerial Images Based on Lightweight Deep Convolutional Network and Generative Adversarial Network, *IEEE Access* 7 (2019) 148119–148130. doi:10.1109/ACCESS.2019.2947143.
- [16] W. Jian, Y. Zhou, H. Liu, Lightweight Convolutional Neural Network Based on Singularity ROI for Fingerprint Classification, *IEEE Access* 8 (2020) 54554–54563. doi:10.1109/ACCESS.2020.2981515.
- [17] Y. Wang, J. Yang, M. Liu, G. Gui, LightAMC: Lightweight Automatic Modulation Classification via Deep Learning and Compressive Sensing, *IEEE Trans. Veh. Technol.* 69 (3) (2020) 3491–3495. doi:10.1109/TVT.2020.2971001.
- [18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (2018) 4510–4520 arXiv:1801.04381, doi:10.1109/CVPR.2018.00474.
- [19] M. Tan, Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, *36th Int. Conf. Mach. Learn. ICML 2019 2019-June* (2019) 10691–10700. arXiv:1905.11946.

- [20] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017 2017-Janua (2017) 2261–2269. arXiv: 1608.06993, doi:10.1109/CVPR.2017.243.
- [21] Y. Li, H. Huang, Q. Xie, L. Yao, Q. Chen, Research on a surface defect detection algorithm based on MobileNet-SSD, Appl. Sci. 8 (9) (2018). doi: 10.3390/app8091678.
- [22] S. Vluymans, Learning from imbalanced data, Stud. Comput. Intell. 807 (9) (2019) 81–110. doi:10.1007/978-3-030-04663-7\_4.
- [23] J. M. Johnson, T. M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (1) (2019). doi:10.1186/s40537-019-0192-5. URL <https://doi.org/10.1186/s40537-019-0192-5>
- [24] W. Ouyang, B. Xu, J. Hou, X. Yuan, Fabric Defect Detection Using Activation Layer Embedded Convolutional Neural Network, IEEE Access (2019). doi:10.1109/ACCESS.2019.2913620.
- [25] K. Simonyan, A. Zisserman, Very deep convolutional networks for largescale image recognition, 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. (2015) 1–14 arXiv:1409.1556.
- [26] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM (2017). doi:10.1145/3065386.
- [27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, Int. J. Comput. Vis. 128 (2) (2020) 336–359. arXiv:1610.02391, doi:10.1007/s11263-019-01228-7.
- [28] L. Biewald, Experiment Tracking with Weights & Biases, Softw. available from wandb.com (January) (2020) 1–5.

## **List of Acronyms and Abbreviations**

**AOI** Automated Optical Inspection. 2

**AUC** Area Under Curve. 5

**CNN** Convolutional Neural Network. 1

**EER** Equal Error Rate. 12

**FN** False Negative. 5

**FNR** False Negative Rate. 6

**FP** False Positive. 5

**FPR** False Positive Rate. 6

**GPU** Graphics Processing Unit. 13

**ReLU** Rectified Linear Unit. 9

**SSD** Single Shot Multibox Detector. 5

**TN** True Negative. 11

**TP** True Positive. 5