Article

Classification of Defective Fabrics Using Capsule Networks

Yavuz Kahraman [1,*] and Alptekin Durmuşoğlu [2]

[1] Industrial Engineering, Adiyaman University, Adiyaman 02040, Turkey
[2] Industrial Engineering, Gaziantep University, Gaziantep 27410, Turkey; durmusoglu@gantep.edu.tr
* Correspondence: ykahraman@adiyaman.edu.tr

Abstract: Fabric quality has an important role in the textile sector. Fabric defect, which is a highly important factor that influences the fabric quality, has become a concept that researchers are trying to minimize. Due to the limited capacity of human resources, human-based defect detection results in low performance and significant loss of time. To overcome human-based limited capacity, computer vision-based methods have emerged. Thanks to new additions to these methods over time, fabric defect detection methods have begun to show almost one hundred percent performance. Convolutional Neural Networks (CNNs) play a leading role in this high-performance success. However, Convolutional Neural Networks cause information loss in the pooling process. Capsule Networks is a useful technique for minimizing information loss. This paper proposes Capsule Networks, a new generation method that represents an alternative to Convolutional Neural Networks for deep learning tasks. TILDA dataset as source data for training and testing phases are employed. The model is trained for 100, 200, and 270 epoch times. Model performance is evaluated based on accuracy, recall, and precision performance metrics. Compared to mainstream deep learning algorithms, this method offers improved performance in terms of accuracy. This method has been performed under different circumstances and has achieved a performance value of 98.7%. The main contributions of this study are to use Capsule Networks in the fabric defect detection domain and to obtain a significant performance result.

Keywords: fabric defect detection; capsule networks; deep learning; image processing

Citation: Kahraman, Y.; Durmuşoğlu, A. Classification of Defective Fabrics Using Capsule Networks. *Appl. Sci.* **2022**, *12*, 5285. https://doi.org/10.3390/app12105285

Academic Editors: Andrea Prati, Luis Javier García Villalba and Vincent A. Cicirello

Received: 17 March 2022
Accepted: 10 May 2022
Published: 23 May 2022

1. Introduction

Quality and inspection of the fabric in the textile sector is a particular phase in terms of cost and time. The quality of the fabric requires the fabric to be flawless. In this connection, defective fabrics significantly affect the quality. Fabric is fabricated from textile fibers, a commonly used material in the textile industry. Over a hundred defect types can occur in the manufactured respective fabric material [1]. Defective fabrics increase the cost price by 45–60% [2]. Due to increasing costs, textile industries are forced to create new solutions such as automatic fabric defect inspection systems that reduce fabric defects to significantly lower rates.

Traditional methods for automatic fabric defect detection explain the models in which fabric feature sets rely on human senses. Traditional methods show promising results for a particular fabric type. However, these models need to be redesigned for new fabric types. Traditional methods require expertise as they are based on human inspection. Since human-based determination has a limited capacity, traditional fabric defect detection methods result in relatively high error rates and loss of time. As an alternative to traditional approaches, modern quality inspection systems that make use of approaches such as machine learning, deep learning, and most machine vision systems have been attracting considerable attention in the scientific field, as well as the textile industry itself. These systems are generally able to provide better accuracy rates [3]. Modern quality inspection approaches have succeeded in accomplishing great results and have demonstrated their convenience for the detection of fabric defects.

Deep learning algorithms, mainly Convolutional Neural Networks (CNN), have a powerful learning ability that can learn the key depth features both adaptively and intelligently. This ability makes fabric defect detection systems more suitable for various defect types. The deep structure of CNN stores the entire input information that represents the defective fabric image. Storing entire representative information contributes to the detection of defect types with higher output performance. Although CNN has many advantages, it can still have shortcomings that sometimes affect the output. For example, it does not store important information such as orientation, rotation, dependency in related internal feature transfer, or spatial information in the image at the desired level. To overcome these shortcomings, the Hinton group invented Capsule Networks (CapsNets), a new model that develops the model hierarchical relationships inside the internal knowledge of network representation [4]. Since their introduction in 2017, there has been an increase in deep learning employing CapsNets as their core building blocks. The popular version of CapsNets uses routing by agreement algorithms between different layers. This algorithm replaces pooling in CNNs and a vector output replacing scalar outputs in CNNs. The magnitude of the output vector represents the likelihood that a feature being represented by the capsule exists in the input image, whereas the orientation of the entity represents the instantiation parameter values. CapsNets are promising in terms of improving our socio-economic activities as they can be deployed to solve real-life problems in autonomous cars [5], machine translation [6], text recognition [7], and image recognition [8]. In this paper, the main purpose is to use CapsNets algorithm in the detection of fabric defects. Depending on this purpose, CapsNets [4,9] is used to classify fabric defects as non-defective, holes, thread, stain, weave, shaded, knitting. The main contributions of this study are summarized as follows:

- We proposed the use of CapsNets for fabric defect detection purposes, which has the advantage of storing whole information about the input.
- In a similar research domain, CapsNets can be applied for various types of textures and fabric defects.
- Experimental results confirm that CapsNets has a superior accuracy rate to other well-known deep learning algorithms.

The study is organized as Section 2 summarizes the literature, Section 3 explains CapsNet implementation, and Section 4 concludes with a series of summary remarks.

2. Literature Review

Modern machine vision-based fabric defect detection systems are an attractive scientific domain as an alternative to traditional methods. In the majority of existing papers, however, homogeneous fabrics are used as input [10]. Publications in this domain can be grouped into five subtitles (Figure 1). Further information can be found in references [11,12].

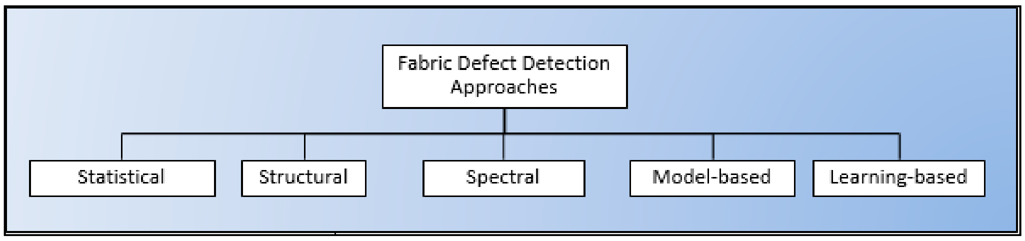


Figure 1. Fabric defect detection approaches.

In the first group approaches, which are statistical, the focus is on some statistical information about pixels. The basic assumption in these approaches is that the defect-free regions are considered to be statistically rigid and the relevant inspection image is partially processed. Representation of this statistical information varies according to the statistical method used in [13]. Co-occurrence matrix (CM) describes the important information as prominent features by computing pixel values' dependence in a CM on each angular spatial relationship [14], where the CM is invariant under monotonic grey value transformation [15]. The mathematical morphology method is used to gain geometric representation information and regional shapes of an image [16]. Operations such as dilation, erosion, smoothing, and sharpening are used in this method. The merits of mathematical morphology can be summarized as sensitivity to defect size and shape, effectiveness in the segmentation processes, and suitability for unidirectional textures. The fractal method uses fractals that are significant in modeling roughness and self-similarity on natural textures [17].

Structural approaches are methods in which primitive elements of textures are used as the main basic characteristics. Textures are imitated by primitives as the main basic characteristics based on certain structural rules. Structural approaches are constructed based on two sequential stages: texture detection and modeling the texture pattern. As a result of the stochastic variability nature in texture patterns (e.g., noise, flexibility, yarn rotation), due to the stochastic variations in texture structures (i.e., the elasticity of yarn, fabric motion, noise, etc.), these approaches are not sufficiently reliable in general. However, these approaches are acceptable in detecting textural defects.

Spectral approaches use both spatial and frequency information to identify the presence of a fabric defect. The frequency information of a fabric image contributes to the detection of defects. Spatial information provides information about the defect location. The majority of publications consider this approach. The most common spectral methods are transform-based methods such as Fourier, Wavelet, and Gabor methods. Wavelet transform is a mapping function that is placed in Fourier and real domains to provide significant localized information by considering vertical, horizontal, and diagonal directions of the inputs [18]. The Fourier method converts the signal information from a time domain to a frequency domain [19]. The last transformation method, Gabor transform, is also used to transform an image representation to both the spatial and frequency domains. Gabor transform is a different version of the short-time Fourier transform. Gabor filters have two implementation categories: filter bank and implementation of optimal filters [20,21].

Model-based approaches build an image model describing and synthesizing textures [22]. The model stores information that represents the entire texture. These representations are provided by model parameters. The advantage of these approaches is that they create feature sets that can match the observed feature sets. These approaches are appropriate for describing stochastic surface variations. The Markov Random Field and Autoregressive models have commonly used techniques in model-based approach subtitles. The Autoregressive model represents the randomness to define computations in terms of the time domain. This method is computationally simple since it requires a linear equation solution [23]. Markov random fields (MRF) use both statistical and structural information to recognize patterns [24]. The main principle in this method is to focus on pixel intensity and its neighborhood pixel values.

Learning-based approaches are used to detect defects utilizing labeled data. This learning mechanism can be supervised and unsupervised. The main goal in these approaches is to create a model that has been trained with labeled data and then uses this model as a test for unseen images. These approaches have good defect tolerance, generalization capabilities, and self-learning capabilities. The most commonly used methods are Elo-rating and machine learning-based methods such as Artificial Neural Network (ANN), Support Vector Machines (SVM), Regression, and Nearest Neighbor (NN). Neural network models are flexible defect detection methods utilizing layer-by-layer structure to transform information forward and update the information backward to minimize network error [25]. The Elo Rating is performed to detect fabric defects and this method has high accuracy in the detection of defects on dot-and-star patterned fabrics [26]. The deep learning methods used for fabric defect detection, specific to the subject of this study, are summarized in three groups: CNN-based; Long Short Term Memory (LSTM) based; Autoencoder based publications. Table 1 gives details about discussed papers in terms of the used dataset, number of classes, and performance. The first seven rows show details of CNN-based publications. The eighth row shows details of the LSTM-based paper. The remaining rows are stands for fabric defect detection studies involving the autoencoder method. For example, Jeyaraj and Nadar performed Multi-scale CNN to categorize and localize six fabric defects by using TILDA dataset. Their study has a higher performance rate with 96.55% accuracy [27]. Zhao et al. proposed a visual long short term memory method to classify three datasets (DHU-FD-500, DHU-FD-1000, Aliyun-FD-10500) into ten and seven classes, respectively. Şeker et al. used autoencoders to classify fabric defects into two categories [28]. The purpose of the work is not only to detect fabric defects, but also to try to extract features of fabrics with unique textures correctly, to create the optimum working parameters for the autoencoder. The autoencoder network was first trained and then tested. The training phase was carried out with 153 defective and 847 defect-free fabric samples. Data were separated as 70% training set, 15% validation set, and 15% test set. The overall performance rate for the study is 88%.

Table 1. Publication details for deep-learning based fabric defect detection literature.

Method	Data	Classes	Acc. [1]	Ref. [2]
Multi-scale CNN	TILDA	6: Strain, Hole, Weft formed, Slub, Stitching, Color change	95.56%	[27]
Autoencoders	Custom Dataset	2: Defective and defect-free	88%	[28]
Pretrained AlexNet	Custom Dataset	2: Defective and Defect-free	98.68%	[29]
Combination of compressive sensing and convolutional neural network (CS-CNN)	Custom Dataset	10: Normal, Mispick, Broken pick, Double flat, Slub, Fleter, Drawback, Sundries, Broken end, Oil stain	97.9%	[30]
Transfer learning and faster RCNN	Custom Dataset	2: Holes &Cuts, Stain	95.48%	[31]
U-Net and Convolutional Neural Network	Custom Dataset	2: Defective and defect-free	96.62%	[32]
Faster Regional-based CNN	Custom Dataset	6: Broken pick, Felter, Drawback, Sundries, Broken end, Oilstains	95.8%	[33]
VGG19	TILDA	7: nondefect, hole, knitting, stain, thread, weave, shaded	94.65%	[34]
Visual Long Short Term Memory (VLSTM)	DHU-FD-500, DHU-FD-1000, Aliyun-FD-10500	10 defect types for DHU-FD-500 and DHU-FD-1000: Mispick, Broken pick, Double Flat, Slub, Felter, Draw-back, Sun-dries, Broken end, and Oil stain 7 defect types for Aliyun-FD-10500: Normal, Broken end, Broken picks, Hole, Stain, Crack, Felter	99.47% 97.73% 95.73%	[35]
Fisher Criterion Based Autoencoder	Custom Dataset	2: Defect-free and defective fabrics.	98.6%	[36]
Autoencoder and MXNet	TILDA Patterned fabric	5: Broken end, Hole, Netting multiple, Thick bar, and Thin bar	98%	[37]
Deep Denoising Convolutional Autoencoder (DDCAE)	Custom Dataset	2: Defective and NonDefective	97.75%	[38]

[1] Accuracy, [2] Reference.

3. Method

Deep learning-based methods tend to trend up in fabric defect detection studies. In recent years, the majority of publications in this field have discussed defect detection strategies. In this paper, the Capsule Network, as a deep learning method proposed by Hinton et al., is used. The route followed in this paper is shown in Figure 2.

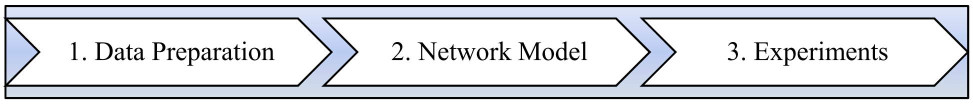


Figure 2. Followed steps.

3.1. Data

TILDA fabric dataset is created by a working group of Texture Analysis of the DFG's (Deutsche Forschungsgemeinschaft) research program [39]. TILDA is used to train, evaluate, and test the model. In TILDA, seven defect classes have been defined. There are eight defect classes in total, with a non-defective class. Each class includes 50 TIFF pictures with dimensions of 768 × 512 (grey-level). The whole dataset contains 3200 defect images. In the TILDA dataset, we used 682 images (total initial dataset before the augmentation process) that are divided into seven classes. Each of the first five classes (nondefective, hole, stain, thread, weave) has 100 pictures, and the knitting and shaded classes have 94 and 88 pictures, respectively (examples for our chosen images are shown in Figure 3).

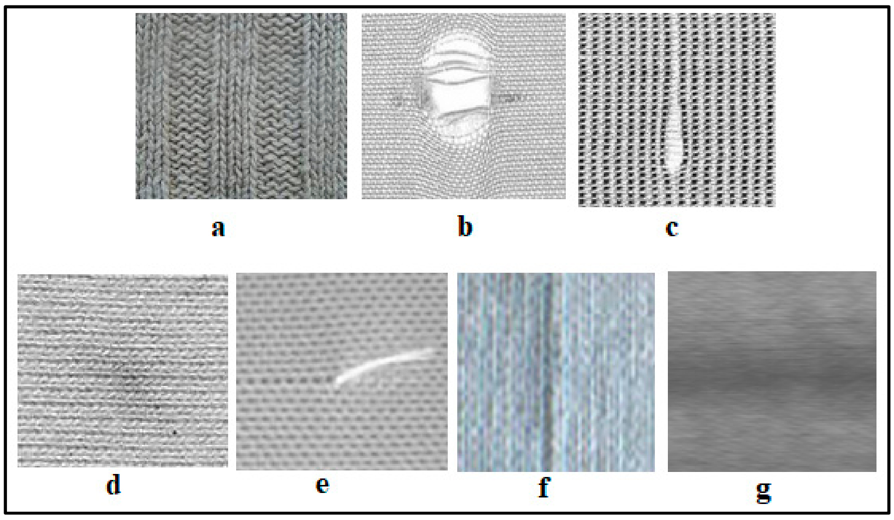


Figure 3. Examples for defect types: (**a**) nondefective; (**b**) hole; (**c**) weave; (**d**) stain; (**e**) thread; (**f**) knitting; (**g**) shaded.

Deep learning approaches provide high accuracy performance. This high performance is mostly due to data containing a large number of samples. The large dataset is of particular importance to the feature creation that represents the data well. However, small datasets may result in models having overfitting problems and result in low accuracy performance in real-life problems. To overcome such problems, data augmentation techniques have emerged. In this paper, our initial dataset consists of seven classes where all classes consist of 682 images in total. Using data augmentation techniques, we have augmented our data with the help of a script written in python software. Used augmentation techniques are flips (vertical and horizontal), different rotation angles, and zooming (in and out). These techniques provide us with the ability to hold image features and do not corrupt the data. Figure 4 shows used techniques and the changing dataset size for all fabric classes.

Experiments are carried out by using augmented data. Augmented data is still not completely ready to be input. Therefore, an operation called data pre-processing is performed. Such operations proceed as follows: dimension reduction; normalization; and data splitting. Firstly, dimension reduction is the reading of all data into a new lower dimension of 64 × 64 as resized images. Secondly, a normalization operation is performed to convert data into one reasonable scale. In other words, normalization is performed to convert data into a range of zeros to ones. Thirdly, splitting the data into two different datasets is of great importance to both the training and testing of the model. This contributes to building an efficient model. Since our network model is supervised, the model should initially be trained. After the training process is complete, the model can be tested with unseen data. Data is divided into training (27,010 images) and the test (11,581 images) image classes. Each training and testing dataset includes all image classes in random order.

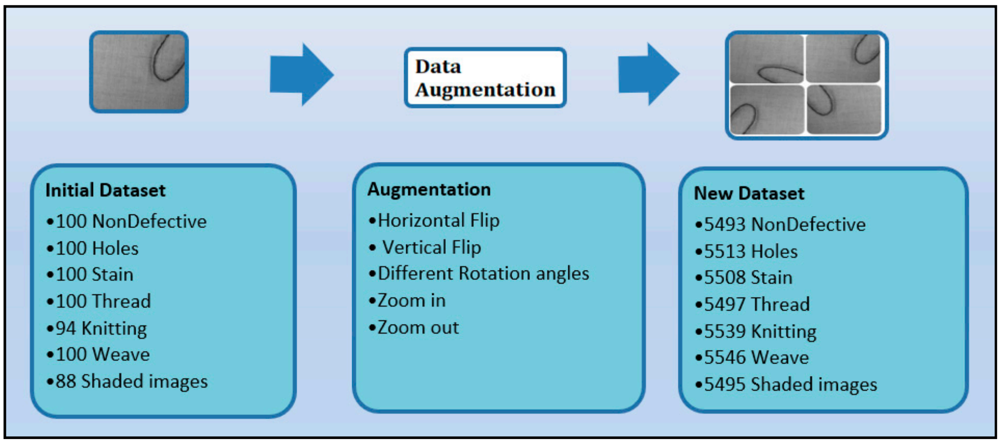


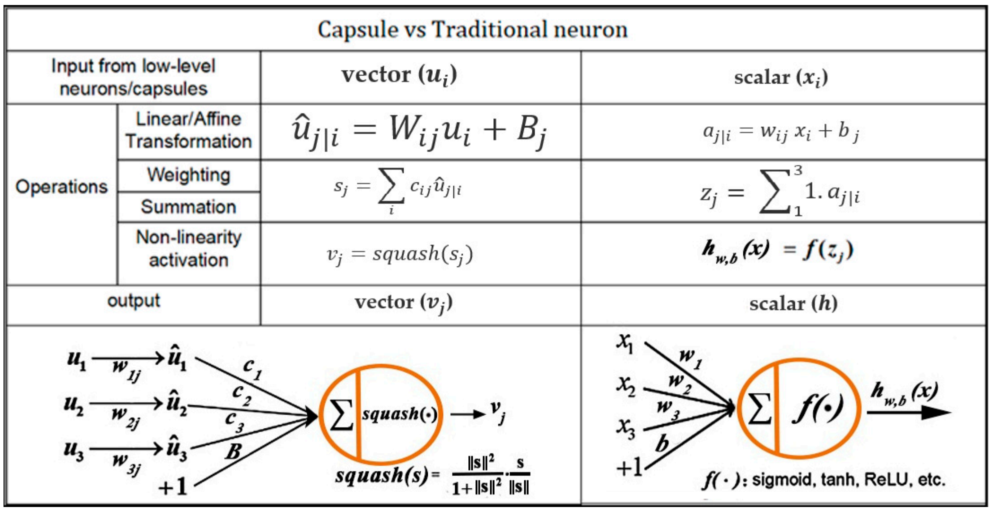
Figure 4. Representative data augmentation process.

3.2. *Network Model*

In this section, procedures relating to Capsule Networks are discussed. The first part of this section considers the original CapsNets explanation, whilst the second is about the implementation of CapsNets for fabric defect detection problems.

3.2.1. Capsule Networks

Convolutional neural networks (CNN) have created great contributions to the machine learning domain. In many classification task domains, CNN's have proven very successful. However, CNN's have some shortcomings that should be taken into account [40]. The basic problem in CNN is that they are translation invariant. This translation invariance problem means keeping the information in a spatial hierarchical manner. More simply, the problem in CNN is to keep the dependent components of an image that must be stored together and in a particular hierarchical form in a random structure. This basic problem is also called the lack of translation equivariance. The lack of translation equivariance in CNN's case is manifest when the network incorrectly assigns the input to a different category label, resulting in false negatives. The second problem in CNN is that they require a large dataset to generalize the network. This is due to the fact that CNN requires a large amount of training data to compensate for the information loss caused by the pooling process [40]. The third problem is the poor imitation of the human visual system. As presented by Bhowmik et al., the closer the simulations of artificial systems to the human visual system, the better the system will perform [41]. To overcome all these problems, CapsNets is an alternative to CNN. As presented by Hinton, CapsNet is different from traditional CNN. These differences lie in the use of vectors instead of scalar values, no pooling operations, a novel activation function for capsules, and the feed-forward training of capsule parameters unlike the back-propagation in CNN. A comparison between capsules and traditional neurons is given in Figure 5.



Capsule vs Traditional neuron			
Input from low-level neurons/capsules		vector $(\boldsymbol{u}_i)$	scalar $(\boldsymbol{x}_i)$
Operations	Linear/Affine Transformation	$\hat{u}_{j\|i} = W_{ij}u_i + B_j$	$a_{j\|i} = w_{ij}\,x_i + b_j$
	Weighting Summation	$s_j = \sum_i c_{ij}\hat{u}_{j\|i}$	$z_j = \sum_1^3 1.\,a_{j\|i}$
	Non-linearity activation	$v_j = squash(s_j)$	$\boldsymbol{h}_{w,b}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{z}_j)$
output		vector $(\boldsymbol{v}_j)$	scalar $(\boldsymbol{h})$

u_1, w_{1j}, $\hat{u}_1$, c_1
u_2, w_{2j}, $\hat{u}_2$, c_2
u_3, w_{3j}, $\hat{u}_3$, c_3
$+1$, B
$\sum$ $squash(\cdot)$ $\rightarrow v_j$
$squash(s) = \frac{\|s\|^2}{1+\|s\|^2} \cdot \frac{s}{\|s\|}$
x_1, w_1, x_2, w_2, x_3, w_3, $+1$, b
$\sum$ $f(\cdot)$ $h_{w,b}(x)$
$f(\cdot)$: sigmoid, tanh, ReLU, etc.

Figure 5. Comparison of capsules and traditional neurons.

A CapsNet is a new CNN form used to provide information about the spatial/presence relationship of features (such as scales, locations, orientations, brightness, etc.) in an image. The foundation of this network is based on the addition of concepts called "capsules" to CNN. A capsule is a group of neurons that include not only the probability of a particular object's presence but also different informative values related to instantiation parameters such as rotation, pose, posture, slope, position, direction, thickness, scale. Unlike CNN components like pooling and convolution, which cause information loss, capsules are capable of holding more information. The innovation in CapsNets is the "routing by agreement" concept, which is replaced with pooling. Based on this concept, outputs are sent to all parent capsules in the next layer, however, their coupling coefficients are not the same. Each capsule tries to estimate the output of the parent capsules, and if this estimate matches the actual output of the parent capsule, the coupling coefficient between these two capsules increases. Considering u_i as the output of capsule i, its prediction for parent capsule j is computed as

$$\hat{u}_{j|i} = W_{ij}u_i \tag{1}$$

where $\hat{u}_{j|i}$ is the prediction vector of the output of the jth capsule in a higher level computed by capsule i in the layer below, and W_{ij} is the weighting matrix that needs to be learned in the backward pass. Based on the degree of confirmation between the capsules in the layer below and the parent capsules, coupling coefficients c_{ij} are calculated using the following softmax function

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ij})} \tag{2}$$

where b_{ij} is the log probability that whether capsule i should be coupled with capsule j and its initially set to zero at the beginning of the routing by agreement process. Hence, the input vector to the parent capsule j is computed as follows

$$s_j = \sum_i a_j \cdot \hat{u}_{j|i} \tag{3}$$

Finally, the following no-linear squashing function is used to prevent the output vectors of capsules from exceeding one and forming the final output of each capsule based on its initial vector value given in Equation (4)

$$\vartheta_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2}\frac{s_j}{\|s_j\|} \tag{4}$$

This new architecture achieves significantly better, state-of-the-art performance accuracy with MNIST data [42]. The CapsNets architecture using MNIST data is shown in Figure 6 as an example.

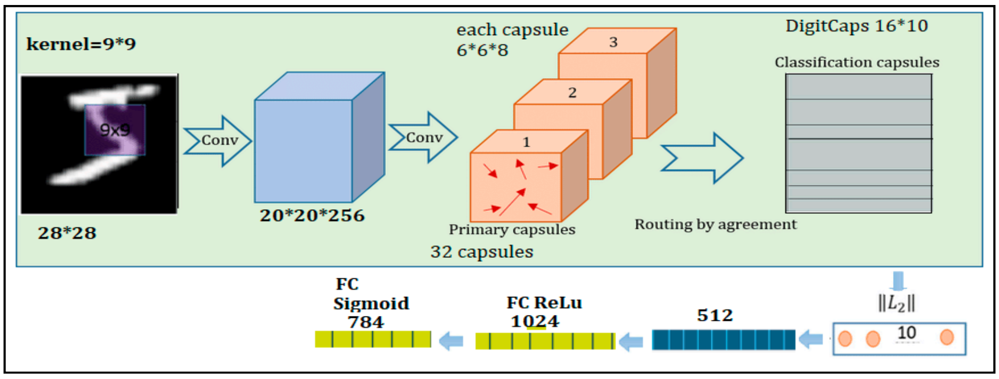


Figure 6. Capsule network architecture-MNIST.

To summarize the capsule network procedure, the capsules determine the parameters of the features in a particular object. During the process of identification of such an object, capsules not only determine the presence or absence of the object's features but also take into account the respective parameters, in which the object's features are organized. This means the system will only detect the object if the features detected by capsules are present in the correct order. The work procedure of the capsules is as follows:

- Initial capsules proceed with the matrix multiplication of the input vectors with weight matrices, which represent the spatial relationships of low-level features with high-level features;
- Capsules decide their parent capsule. This is achieved through the use of the dynamic routing algorithm. For example, suppose the system tries to recognize a picture of a house from different viewpoints. Capsules extract the information about the roof and walls, but this does not mean any roof can be a house. Therefore, the capsules analyze the consistent part in an image. To decide if the object is a house (or otherwise), predictions are made by using both roofs and walls. These predictions are then sent to the higher-level capsule. If the estimation is correct, the object is assigned as a house;
- After the parent capsule's decision, the process proceeds by summing all vectors that will be ultimately squashed to between zero and one while retaining their directions. Squashing is performed using a cosine distance as a measure of the agreement and norm calculations as the probability of presence.

3.2.2. Capsule Networks Implementation

This section describes the work procedure of our model framework, the hyperparameter settings, and the training/testing process. The model framework is designed in a similar way, but is not identical to the original Hinton model framework. The model attempts to detect and categorize six different defect classes (holes, stain, thread, knitting, weave, and shade) and one defect-free class.

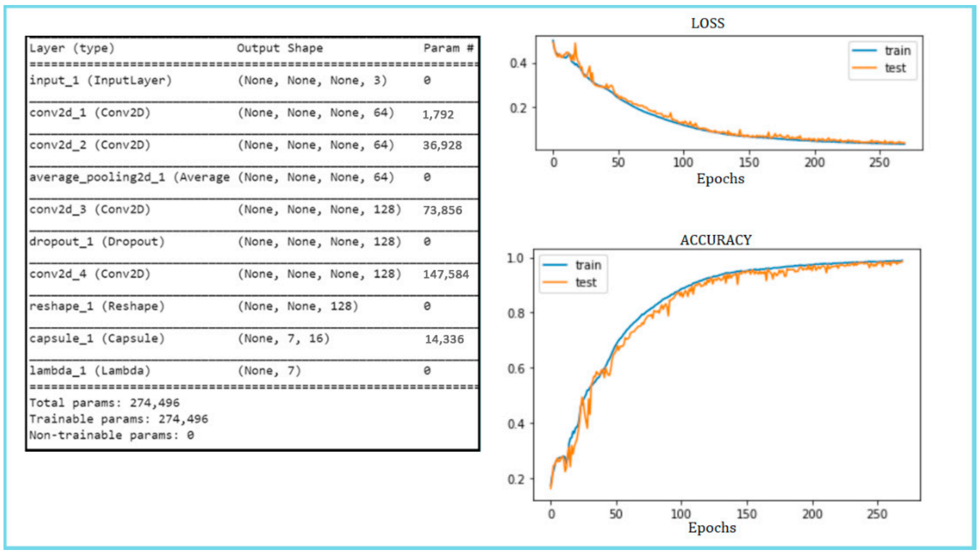
The CapsNet framework consists of two parts: the encoder and the decoder. The encoder includes a convolutional layer (256 channels each made up of 6 $\times$ 6 filters with a stride of one and a ReLu activation function applied to a 64 $\times$ 64 $\times$ 1 fabric image), PrimaryCaps layer (convolutional capsule layer with 6 $\times$ 6 $\times$ 32 capsules each outputting an 8D vector. At a stride of one, each primary capsule has eight convolutional units operating with a 3 $\times$ 3 kernel), and DigitCaps layer (this layer is a fully connected layer with seven 16D capsules created by applying squashing function, each digit caps the receiving input from all capsules from the layer below to perform classification based on seven classes); the decoder part (the last part of the structure that determines the length of each capsule in the previous layer necessary to obtain the probability that entity is present. Decoder past is also a reconstruction of input made up of fully connected layers), which contain a fully connected layer1, fully connected layer2, and fully connected layer3, which are DigitCaps outputs, fully connected layer1 outputs, and fully connected layer2 outputs, respectively.

Hyperparameter settings have a very significant effect on the accurate prediction of the model. Optimal hyperparameter settings generally vary for various datasets, and should ideally be tuned for each specific dataset. CapsNets have their parameter-tuning algorithms that are run during the training process. Table 2 reports the hyperparameters that we used in our model. The parameters given in italics in Table 2 are the parameter settings that have been changed, different from the original model (i.e., while using 9 × 9 filters with a stride of one setting in the original basic structure of the capsule network, we used 3 × 3 and 6 × 6 filters with stride of one). Non-italic parameters in Table 2 are considered as constant/fixed for all experimental runs, and training and testing processes are carried out.

Table 2. Hyperparameter settings (Changed setting: Batch size, kernel size, and length of digitcaps).

Parameter	Setting
Activation	ReLu
Batch Size	*64*
Epoch Number	*270*
Optimizer	Adam
Kernel Size	*3 × 3, 6 × 6*
Routing Time	3
Dimension of Each Capsule	8
Length of Primary Caps	7
Length of DigitCaps	*7*

Detection accuracy and model deep dimensions are two important terms in deep learning algorithms. Studies have shown that one of the more significant ways to improve accuracy is to enlarge the network's depth and width [43]. Deep network architectures require significant computational resources, but CapsNets have provided high accuracy results and deeper network structures. In this paper, the training and testing process is performed using the TILDA dataset. Defect recognition is based on seven categories, namely nondefective, holes, stains, knitting, threads, weaves, and shade categories. The training process is performed three times (100, 200, 270) based on epoch number changes. Details for these changes are given in Section 3.3. The ideal model parameters are given in Table 2. The structure, loss and accuracy graphics per epoch are illustrated in Figure 7.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 64)    1,792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 64)    36,928
_________________________________________________________________
average_pooling2d_1 (Average (None, None, None, 64)    0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 128)   73,856
_________________________________________________________________
dropout_1 (Dropout)          (None, None, None, 128)   0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, None, None, 128)   147,584
_________________________________________________________________
reshape_1 (Reshape)          (None, None, 128)         0
_________________________________________________________________
capsule_1 (Capsule)          (None, 7, 16)              14,336
_________________________________________________________________
lambda_1 (Lambda)            (None, 7)                 0
=================================================================
Total params: 274,496
Trainable params: 274,496
Non-trainable params: 0
```

Figure 7. Model Structure, loss-accuracy graphs.

3.3. Experiments

Experiments on the CapsNet model are reported by considering key terms, such as evaluation criteria, like performance metrics, performance results, novel algorithm comparisons, and model execution properties as environmental configurations or implementation details.

The performance evaluation was carried out by utilizing performance determination through the correct classification rate (CCR as accuracy), recall, and precision. These metrics are formulated in Equations (5)–(7), respectively.

$$CCR = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

where true positive, true negative, false positive, and false negative are *TP*, *TN*, *FP*, and *FN*, respectively. Table 3 reports the metrics utilized in the accuracy criteria.

Table 3. Accuracy metric explanations.

Term	Explanation
True Positive (TP)	The actual sample is positive and is predicted to be positive
True Negative (TN)	The actual sample is negative and is predicted to be negative
False Positive (FP)	Actual is negative, but it is predicted to be positive
False Negative (FN)	Actual is positive, but it is predicted to be negative

The performance verification of our model is compared with three algorithms, namely the Multiscale CNN, AlexNet [44], VGG19, and VGG16 [45]. Since these methods use TILDA data, it is appropriate to compare our model with these methods. In addition, how our model gave performance results in three different epoch numbers (100, 200, 270) has also been examined. As for the compared models and our model with different epoch numbers, the results were evaluated separately. The multi-scale CNN is analyzed in two different classes, defective and nondefective, and a performance result value (CCR) of 95.56% was obtained. The VGG19 model has the same characteristics as our model's class number and type and produced a result value of 94.65%. The VGG16 model was analyzed in two different classes, defective and nondefective, and a result value of 96.38% was obtained. The AlexNet model was evaluated for two different classes, defective and nondefective, and gave a performance result value of 95.42%. The evaluation of our model at different epoch numbers is as follows. The performance of the model was evaluated separately for 100, 200 and 270 epochs, with the hyperparameter settings remaining the same. The model resulted in 85.82% performance for 100 epoch number. For 200 epoch number, this performance value is 96.22%. The performance value with the best accuracy value of the model was obtained at the epoch number of 270. Considering all these evaluations, our model was found to perform better, with the model's testing accuracy achieving 98.71% as reported in Table 4. Furthermore, Table 5 explains our model's performance by adopting CCR, recall, and precision metrics based on 100, 200, and 270 epoch times, respectively.

Table 4. Performance comparisons.

Model	CCR	Target Classes	Class Names
Multi-scale CNN	95.56%	2	Defective and Nondefective
VGG19	94.65%	7	Nondefective, Holes, Knitting, Weave, Shaded, Stain, Thread
VGG16	96.38%	2	Defective and Nondefective
AlexNet	95.42%	2	Defective and Nondefective
CapsNets 100	85.82%	7	Nondefective, Holes, Knitting, Weave, Shaded, Stain, Thread
CapsNets 200	96.22%	7	Nondefective, Holes, Knitting, Weave, Shaded, Stain, Thread
Proposed CapsNet	***98.71%***	*7*	Nondefective, Holes, Knitting, Weave, Shaded, Stain, Thread

Table 5. Performance evaluation.

Models	CCR (Accuracy)	Recall	Precision
CapsNets 100	85.82%	85.24%	87.12%
CapsNets 200	96.22%	95.8%	96.40%
Proposed CapsNet	***98.71%***	***97.82%***	***98.18%***

4. Conclusions

A supervised novel deep learning model—CapsNets—has been used to classify fabric defects into seven categories. TILDA images were used as source data for model training and testing. The proposed model was trained over 270 epochs. Hyperparameter tuning was also performed and contributed in terms of performance metrics. The experimental results have been presented and discussed. The accuracies show that CapsNet algorithm gives significant performance results. Compared to some CNN based algorithms, CapsNets have higher performance results. Experimental results indicate that it has obtained an accuracy of 98.71%.

For future studies, two scenarios will be considered: the first is to apply the model in real-time applications; and the second is to compare the model with different fabric types such as carpet, canvas, woven, and twill. Future research zones for CapsNets could be model parameter optimization, network structure modification, investigation of relationships of inner concepts (i.e., capsules, layers, routing algorithm, squashing).

Author Contributions: Conceptualization, Y.K. and A.D.; methodology, Y.K. and A.D.; software, Y.K. and A.D.; validation, A.D.; formal analysis, Y.K. and A.D.; investigation, Y.K.; resources, Y.K. and A.D.; data curation, Y.K.; writing—original draft preparation, Y.K. and A.D.; writing—review and editing, A.D.; visualization, Y.K. and A.D.; supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ngan, H.Y.T.; Pang, G.K.H.; Yung, N.H.C. Automated Fabric Defect Detection—A Review. *Image Vis. Comput.* **2011**, *29*, 442–458. [CrossRef]
2. Tiwari, V.; Sharma, G. Automatic Fabric Fault Detection Using Morphological Operations on Bit Plane. *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)* **2015**, *15*, 30.
3. Susan, S.; Sharma, M. Automatic Texture Defect Detection Using Gaussian Mixture Entropy Modeling. *Neurocomputing* **2017**, *239*, 232–237. [CrossRef]
4. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. *Adv. Neural Inf. Processing Syst.* **2017**, *2017*, 3857–3867.
5. Kumar, A.D. Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks. *arXiv* **2018**, arXiv:1805.04424.

6. Wang, M.; Xie, J.; Tan, Z.; Su, J.; Xiong, D.; Li, L. Towards Linear Time Neural Machine Translation with Capsule Networks. *arXiv* **2018**, arXiv:1811.00287.
7. Mandal, B.; Dubey, S.; Ghosh, S.; Sarkhel, R.; Das, N. Handwritten Indic Character Recognition Using Capsule Networks. In Proceedings of the 2018 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, India, 7–9 December 2018; pp. 304–308.
8. Chao, H.; Dong, L.; Liu, Y.; Lu, B. Emotion Recognition from Multiband EEG Signals Using CapsNet. *Sensors* **2019**, *19*, 2212. [CrossRef]
9. Hinton, G.; Sabour, S.; Frosst, N. Matrix Capsules with EM Routing. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018—Conference Track Proceedings, Vancouver, BC, Canada, 30 April–3 May 2018.
10. Bennamoun, M.; Bodnarova, A. Automatic Visual Inspection and Flaw Detection in Textile Materials: Past, Present and Future. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA, USA, 11–14 October 1998; Volume 5, pp. 4340–4343.
11. Mahajan, P.M.; Kolhe, S.R.; Patil, P.M. A Review of Automatic Fabric Defect Detection Techniques. *Adv. Comput. Res.* **2009**, *1*, 18–29.
12. Hanbay, K.; Talu, M.F.; Özgüven, Ö.F. Fabric Defect Detection Systems and Methods—A Systematic Literature Review. *Optik* **2016**, *127*, 11960–11973. [CrossRef]
13. Jain, A.K.; Duin, R.P.W.; Mao, J. Statistical Pattern Recognition: A Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 4–37. [CrossRef]
14. Fanizzi, A.; Basile, T.M.; Losurdo, L.; Bellotti, R.; Bottigli, U.; Campobasso, F.; Didonna, V.; Fausto, A.; Massafra, R.; Tagliafico, A.; et al. Ensemble Discrete Wavelet Transform and Gray-Level Co-Occurrence Matrix for Microcalcification Cluster Classification in Digital Mammography. *Appl. Sci.* **2019**, *9*, 5388. [CrossRef]
15. Armi, L.; Fekri-Ershad, S. Texture image analysis and texture classification methods—A review. *Int. Online J. Image Processing Pattern Recognit.* **2019**, *2*, 1–29.
16. Shih, F.Y. *Image Processing and Mathematical Morphology: Fundamentals and Applications*; CRC Press: Boca Raton, FL, USA, 2017. [CrossRef]
17. Marusina, M.Y.; Karaseva, E.A. Automatic Analysis of Medical Images Based on Fractal Methods. In Proceedings of the 2019 International Conference” Quality Management, Transport and Information Security, Information Technologies” (IT&QM&IS), Sochi, Russia, 23–27 September 2019; pp. 349–352.
18. Yang, L.; Su, H.; Zhong, C.; Meng, Z.; Luo, H.; Li, X.; Tang, Y.Y.; Lu, Y. Hyperspectral Image Classification Using Wavelet Transform-Based Smooth Ordering. *Int. J. Wavelets Multiresolut. Inf. Processing* **2019**, *17*, 1950050. [CrossRef]
19. Osgood, B.G. *Lectures on the Fourier Transform and Its Applications*; American Mathematical Soc.: Providence, RI, USA, 2019; Volume 33, ISBN 1470441918.
20. Mak, K.L.; Peng, P. An Automated Inspection System for Textile Fabrics Based on Gabor Filters. *Robot. Comput.-Integr. Manuf.* **2008**, *24*, 359–369. [CrossRef]
21. Bodnarova, A.; Bennamoun, M.; Latham, S. Optimal Gabor Filters for Textile Flaw Detection. *Pattern Recognit.* **2002**, *35*, 2973–2991. [CrossRef]
22. Hanbay, K.; Talu, M. Kumaş Hatalarının Online/Offline Tespit Sistemleri ve Yöntemleri. *Sak. Univ. J. Sci.* **2014**, *18*, 49–69. [CrossRef]
23. Zhang, Y.; Liu, B.; Ji, X.; Huang, D. Classification of EEG Signals Based on Autoregressive Model and Wavelet Packet Decomposition. *Neural Processing Lett.* **2017**, *45*, 365–378. [CrossRef]
24. Smii, B. Markov Random Fields Model and Applications to Image Processing. *AIMS Math.* **2022**, *7*, 4459–4471. [CrossRef]
25. Gurney, K. *An Introduction to Neural Networks*; CRC Press: Boca Raton, FL, USA, 2018; ISBN 1315273578.
26. Tsang, C.S.C.; Ngan, H.Y.T.; Pang, G.K.H. Fabric Inspection Based on the Elo Rating Method. *Pattern Recognit.* **2016**, *51*, 378–394. [CrossRef]
27. Jeyaraj, P.R.; Samuel Nadar, E.R. Computer Vision for Automatic Detection and Classification of Fabric Defect Employing Deep Learning Algorithm. *Int. J. Cloth. Sci. Technol.* **2019**, *31*, 510–521. [CrossRef]
28. Şeker, A.; Peker, K.A.; Yüksek, A.G.; Delibaş, E. Fabric Defect Detection Using Deep Learning. In Proceedings of the 2016 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, Turkey, 16–19 May 2016; pp. 1437–1440.
29. Şeker, A. Evaluation of Fabric Defect Detection Based on Transfer Learning with Pre-Trained AlexNet. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–4.
30. Wei, B.; Hao, K.; Tang, X.S.; Ding, Y. A New Method Using the Convolutional Neural Network with Compressive Sensing for Fabric Defect Classification Based on Small Sample Sizes. *Text. Res. J.* **2019**, *89*, 3539–3555. [CrossRef]
31. Siegmund, D.; Prajapati, A.; Kirchbuchner, F.; Kuijper, A. An Integrated Deep Neural Network for Defect Detection in Dynamic Textile Textures. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11047, pp. 77–84.
32. Kopaczka, M.; Saggiomo, M.; Güttler, M.; Kielholz, K.; Merhof, D. Detection and Classification of Faulty Weft Threads Using Both Feature-Based and Deep Convolutional Machine Learning Methods. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11351, pp. 141–163.

33. Wei, B.; Hao, K.; Tang, X.S.; Ren, L. Fabric Defect Detection Based on Faster RCNN. In *Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 849, pp. 45–51.
34. Durmusoglu, A.; Kahraman, Y. Detection of Fabric Defects Using Convolutional Networks. In Proceedings of the 2021 Innovations in Intelligent Systems and Applications Conference (ASYU), Elazig, Turkey, 6–8 October 2021; pp. 1–5. [CrossRef]
35. Zhao, Y.; Hao, K.; He, H.; Tang, X.; Wei, B. A Visual Long-Short-Term Memory Based Integrated CNN Model for Fabric Defect Image Classification. *Neurocomputing* **2020**, *380*, 259–270. [CrossRef]
36. Li, Y.; Zhao, W.; Pan, J. Deformable Patterned Fabric Defect Detection with Fisher Criterion-Based Deep Learning. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 1256–1264. [CrossRef]
37. Tian, H.; Li, F. Autoencoder-Based Fabric Defect Detection with Cross-Patch Similarity. In Proceedings of the 16th International Conference on Machine Vision Applications, MVA 2019, Tokyo, Japan, 27–31 May 2019.
38. Zhang, H.; Tang, W.; Zhang, L.; Li, P.; Gu, D. Defect Detection of Yarn-Dyed Shirts Based on Denoising Convolutional Self-Encoder. In Proceedings of the 2019 IEEE 8th Data Driven Control and Learning Systems Conference, DDCLS 2019, Dali, China, 24–27 May 2019; pp. 1263–1268.
39. Schulz-Mirbach, H. TILDA-Ein Referenzdatensatz zur Evaluierung von Sichtprüfungsverfahren für Textiloberflächen. *Interner Ber.* **1996**, *4*, 96.
40. Kwabena Patrick, M.; Felix Adekoya, A.; Abra Mighty, A.; Edward, B.Y. Capsule Networks—A Survey. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 1295–1310. [CrossRef]
41. Bhowmik, P.; Pantho, M.J.H.; Bobda, C. Bio-Inspired Smart Vision Sensor: Toward a Reconfigurable Hardware Modeling of the Hierarchical Processing in the Brain. *J. Real-Time Image Processing* **2021**, *18*, 157–174. [CrossRef]
42. Rajasegaran, J.; Jayasundara, V.; Jayasekara, S.; Jayasekara, H.; Seneviratne, S.; Rodrigo, R. Deepcaps: Going Deeper with Capsule Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 10725–10733.
43. Li, Y.; Zhang, D.; Lee, D.J. Automatic Fabric Defect Detection with a Wide-and-Compact Network. *Neurocomputing* **2019**, *329*, 329–338. [CrossRef]
44. Arora, P.; Hanmandlu, M. Detection of Defects in Fabrics Using Information Set Features in Comparison with Deep Learning Approaches. *J. Text. Inst.* **2021**, *113*, 266–272. [CrossRef]
45. Dong, Y.; Wang, J.; Li, C.; Liu, Z.; Xi, J.; Zhang, A. Fusing Multilevel Deep Features for Fabric Defect Detection Based NTV-RPCA. *IEEE Access* **2020**, *8*, 161872–161883. [CrossRef]