DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FABRIC DEFECT DETECTION USING DEEP NEURAL NETWORKS

by

Muhammet ÖZCAN

February, 2024

İZMİR

FABRIC DEFECT DETECTION USING DEEP NEURAL NETWORKS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for Master of Science in
Department of Mechatronics Engineering, Mechatronics Engineering Program

by

Muhammet ÖZCAN

February, 2024

İZMİR

THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"FABRIC DEFECT DETECTION USING DEEP NEURAL NETWORKS"** completed by **MUHAMMET ÖZCAN** under supervision of **ASSOC.PROF.DR. YAVUZ ŞENOL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc.Prof.Dr. Yavuz ŞENOL

Supervisor

Prof. Dr. Mustafa GÜNDÜZALP

Jury Member

Prof. Dr. Olcay AKAY

Jury Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

I would like to thank my advisor Assoc. Prof. Dr. Yavuz ŞENOL, who guided me with his experience and knowledge during my master's degree and never spared his support.

I would like to thank my wife and daughters, who are my biggest supporters.

Muhammet ÖZCAN

FABRIC DEFECT DETECTION USING DEEP NEURUL NETWORKS

ABSTRACT

Control of fabric quality is extremely important in the industry. Visual fabric defect detection causes a decrease in the defect detection rate and waste of time. In this thesis study, pre-trained convolutional neural network architectures were used to detect fabric defects using the ZJU-Leaper fabric dataset. In this study, three different fabric types were studied from the data set: plain white, thick stripe, and thin stripe. When the literature was examined, VGG16, EfficientNetB0, ResNet50, DenseNet121 and MobileNet architectures, which are the most frequently used pre-trained convolutional neural network architectures in classification problems, were selected. Data sets were divided into training and test folders, and the performance of the models was compared according to precision, recall, F1-score, and accuracy criteria.

As a result, when the performances of the models on 3 different fabric types were examined, it was observed that the most successful model was EfficientNetB0. The performance of the EfficientNetB0 model was almost 99% on all fabric types.

Keywords: ZJU-Leaper, Deep learning, Convolutional neural networks, Fabric defect detection, Google Colab

DERİN SİNİR AĞLARI KULLANILARAK KUMAŞ HATA TESPİTİ

ÖZ

Endüstride kumaş kalitesinin denetimi son derece önemlidir. Gözle yapılan kumaş hatası tespiti, hata bulma oranının düşmesine ve zaman kaybına neden olur. Bu tez çalışmasında, ZJU-Leaper kumaş veri seti kullanılarak kumaş hatalarının tespiti için önceden eğitilmiş evrişimli sinir ağı mimarileri kullanılmıştır. Bu çalışmada, veri setinden düz beyaz, kalın şerit, ince şerit olmak üzere 3 farklı kumaş türü üzerinde çalışma yapılmıştır. Literatür incelendiğinde sınıflandırma problemlerinde en sık kullanılan, önceden eğitilmiş evrişimli sinir ağı mimarileri olan VGG16, EfficientNetB0, ResNet50, DenseNet121 ve MobileNet mimarileri seçilmiştir. Veri setleri eğitim ve test klasörlerine ayrılarak, modellerin performansı kesinlik, duyarlılık, F1-skoru ve doğruluk kriterlerine göre karşılaştırılmıştır.

Sonuç olarak, modellerin 3 farklı kumaş türü üzerindeki performansları incelendiğinde en başarılı modelin EfficientNetB0 olduğu görüldü. EfficientNetB0 modelinin performansı tüm kumaş türlerinde neredeyse %99'du.

Anahtar kelimeler: ZJU-Leaper, Derin öğrenme, Evrişimli sinir ağları, Kumaş hatası tespiti, Google Colab

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

1.1 Importance and Aim of the Thesis

In this thesis, detailed information is given about the concept of deep learning, convolutional neural network architectures and hyperparameters used to evaluate the productivity of models. Additionally, the productivity of different models for detecting fabric defects were compared in the thesis.

1.2 General Structure of the Thesis

In the second chapter, an extensive literature review was conducted. In the third chapter, the definition of deep learning, the structure and types of convolutional neural network models, hyperparameters and performance evaluation metrics are mentioned. In the fourth chapter, the material part is explained. The ZJU-Leaper data set used in the thesis and the software used in training the models are mentioned. In the fifth chapter, the method and findings are shown. In this chapter, where the results obtained are shown, the method used is explained in detail. Finally, in the sixth chapter, the results obtained in the thesis study are compared and suggestions are made about future studies.

CHAPTER TWO
LITERATURE REVIEW

2.1 Convolutional Neural Networks Literature Review

Kırat and Aydın (2023) involving rail surface defect detection, they trained a data set consisting of 4 different classes with VGG16 and MobileNetV3 Small models in their study. As a result of the training, VGG16 reached 98% accuracy level.

Ekici and Ustaoğlu (2023) compared the performances of VGG16, GoogLeNet, and ResNet50 architectures using a data set consisting of a total of 7200 images for the detection of physical damage in buildings. It was observed that VGG16 and ResNet50 architectures showed high performance.

Şahin et al. (2023) created a data set called GHCD11, which includes 11 different planthopper species. In the study conducted on this data set using the convolutional neural network architectures VGG16, VGG19, ResNet50, DenseNet121, EfficientNet, and MobileNet, accuracy between 95% and 99% was achieved.

Yıldırım and Özbay (2022) used fundus images to diagnose glaucoma. In this study, where ResNet18, AlexNet, VGG16, GoogleNet, and SqueezeNet models were used, VGG16 achieved the best sensitivity value of 97.96%.

Demirci et al. (2022), in their study to predict the diseases of tomato plants, achieved 97% accuracy in the tests performed with the ResNet50 architecture for the detection of alternaria and mildew diseases.

Örenç et al. (2022) used EfficientNetB3, ResNet50 and InceptionV3 architectures for the detection of monkeypox. When the performances were compared, the accuracy of ResNet50 gave the best result with 94%.

Eliaçık et al. (2023) made use of a dataset coming out of 4 different classes to evaluate the performance of brain tumors in MRI scans. In this study, where EfficientNet, ResNet, VGG16, Inception-V3, and DenseNet architectures were used, it was observed that EfficientNet and ResNet models showed higher performance.

Eryılmaz and Karacan (2021) used lung x-ray images in detecting COVID-19. 11,293 X-Ray views with images such as Normal, COVID-19 and Lung Opacity were classified using DenseNet121, MobileNetV2, NASNetMobile, and Xception models. When the classification results were compared, the most successful result was the DenseNet121 model with an accuracy rate of 92.16%.

In his study on the classification of white blood cells, Bozkurt (2021) compared the performance of VGG19, DenseNet121, Xception, and EfficientNetB1 models on 12507 white blood cell images. The DenseNet121 model was the most successful model with an accuracy rate of 94%.

Mpinyuri and Tarambiwa (2022) used MobileNetV2 and DenseNetV121 models in their study on vehicle damage classification. In this study, where various damages were classified, MobileNetV2 showed a success rate of 78% and DenseNetV121 showed a success rate of 84%.

Gill and Gupta (2023) classified arrhythmias using ECG images. In this study, DenseNet 121 showed a success rate of more than 80%.

Şafak and Barışçı (2022) worked with MobileNet, EfficientNetB0, MobileNetV2 and NASNetMobile models using the FFHQ dataset in their study on the detection of fake face images. As a result of the study, the EfficientNetB0 model reached a 93% accuracy rate.

Goutham et al. (2022) used the EfficientNetB0 model in brain tumor detection. In the study using four different image classes, a 96% accuracy rate was achieved.

Arun and Viknesh (2022) studied 11 different classes of plant leaves. AlexNet, ResNet50, EfficientNet BO to B7, and Xception models were used. It has been observed that the productivity of the EfficientNetB5 model was higher.

Bilgin et al. (2022) used the data set called 250 Bird Species in their study on the classification of bird species. In the study carried out on this dataset, which includes 225 bird species and 31316 bird images, the performances of InceptionV3, VGG16, ResNet152V2, MobileNet, ResNet50, and DenseNet121 models were evaluated. In experimental studies, DenseNet121 was the most successful model with an accuracy value of 98.2%.

Şafak and Barışçı (2023) used different deep learning models for fire and smoke detection. In this study, where EfficientNetB0, revised MobileNet, MobileNet, MobileNetV2, and ShuffleNet models were used, the revised MobileNet network reached a success rate of 98%.

İşlek and Hamza (2023) in a study conducted using the Compound Emotion (CE) dataset, 7 different emotion expressions were used. Data augmentation was performed on 1610 images containing natural, sad, happy, angry, fearful, surprised, and disgust emotions. Successful results were obtained in the study using MobileNet and VGG19 models and various optimization algorithms.

Utomo et al. (2023) conducted a study on the determination of asphalt road quality using the VGG16 architecture. In the study conducted with 224x224 images, the accuracy vote was 92.47%.

Zhang et al. (2022) studied the classification of rubber seal defects using the VGG16 architecture. Its accuracy was 91.2%.

Wang (2020) used VGG16 architecture in his study on the classification of domestic garbage. Garbage is divided into classes as kitchen waste, hazardous garbage, recyclable garbage, and other garbage. The accuracy rate of the model was 81.1%.

Şengöz (2023) used VGG16, InceptionV3, ResNet50, Xception, DenseNet121, and EfficientNetB0 architectures with data augmentation method to detect eye diseases in cats and dogs. In the study conducted on 255 healthy and 146 diseased images, VGG16 architecture reached a success rate of 99.9%.

Cengiz et al. (2021) used VggNet, GoogleNet, and ResNet50 deep learning algorithms in their study on classification of people and vehicles. In the study conducted on the Stanford and AKÜ dataset, the ResNet50 model was more successful than other models.

Giray and Doğan (2023) used VGG16, ResNet50, InceptionV3, and Xception architectures in their study on lane detection for autonomous driving in video games. When the performances of the architectures are compared, the ResNet50 architecture shows the best performance.

2.2 Fabric Defect Detection Literature Review

Varjovi et al. (2022) studied 13820 fabric images. The performance of the pre-trained Inception V3, MobileNetV2, Xception, and ResNet50 architectures and the self-designed models were compared. As a result, the model they designed was more successful with 97%.

Şeker and Yüksek (2017) in their study on the fabric data set they created, they achieved 96% success by using the stacked autoencoder method, one of the deep learning methods.

Peng et al. (2021) analyzed the productivity of the AMTFNet model they proposed in their study on the Aitex fabric dataset with VGG19, ResNet50, ResNet101, InceptionV3, and MobileNetV2 architectures. As a result of the study, the AMTFNet model shows higher success.

Durmuşoğlu and Kahraman (2021) studied the TILDA dataset using the VGG19 architecture to detect fabric defects. High success rate was achieved in studies conducted on a total of 7 different fabric types.

Şeker (2018) used the AlexNet model in his study on the fabric data set he created. In the study conducted on 3725 images, 98% success was achieved with transfer learning.

Çelik et al. (2022) present a system in which fabric defects are automatically detected. In the study where ResNet50, AlexNet, and GoogLeNet architectures were used, they achieved 98% success with ResNet50.

Hamdi et al. (2018) studied fabric defect definition making use of the K-means clustering and texture filtering method. The success rate in this study reached 95%.

Liu et al. (2022) in their study with an unsupervised UNet model on a special fabric data set, the accuracy rate reached 99%.

Zhang et al. (2018) investigated yarn dyed fabric defect make use of different models based upon YOLOV2. Experimental results making use of YOLO-VOC have been shown to be more effective in fabric defect detection.

Yin and Li (2022) conducted a study on woven fabric defect details. The proposed model has been tested on Tianchi public data. This study, in which the YOLO-Nano model was used, was found to be suitable for defect detection.

Erdoğan (2022) used 2 different data sets in his study on fabric defects using the YOLOv4 algorithm. In this study where data augmentation was used, the YOLOv4 algorithm was quite successful.

Kahraman (2022) investigated fabric defects making use of Capsule Networks and VGG19 architecture. In this study, the autoencoder method was applied in a different way.

Çıklaçandır (2022) used different feature extraction methods in her study on fabric defects. The performances of GoogLeNet, ResNet18, ResNet50, and AlexNet architectures are compared.

Guan et al. (2019) conducted a defect detection study in plain weaving. This study, in which various fabric types were used, was tested making use of the VGG model. Efficiency of the VGG model is verified.

CHAPTER THREE
DEEP LEARNING

Compact of multiple layers, deep learning is an undersphere of machine learning that works by imitating the human brain. The basis of deep learning is based upon artificial neural networks, which are machine learning algorithms. While feature extraction in machine learning is done by a human, in deep learning this process is done automatically by the algorithm. The structural difference between deep learning and machine learning is shown as an example in Figure 3.1. The biggest benefit of deep learning is that it works with big data. Since deep learning works with big data, the training period takes a long time.

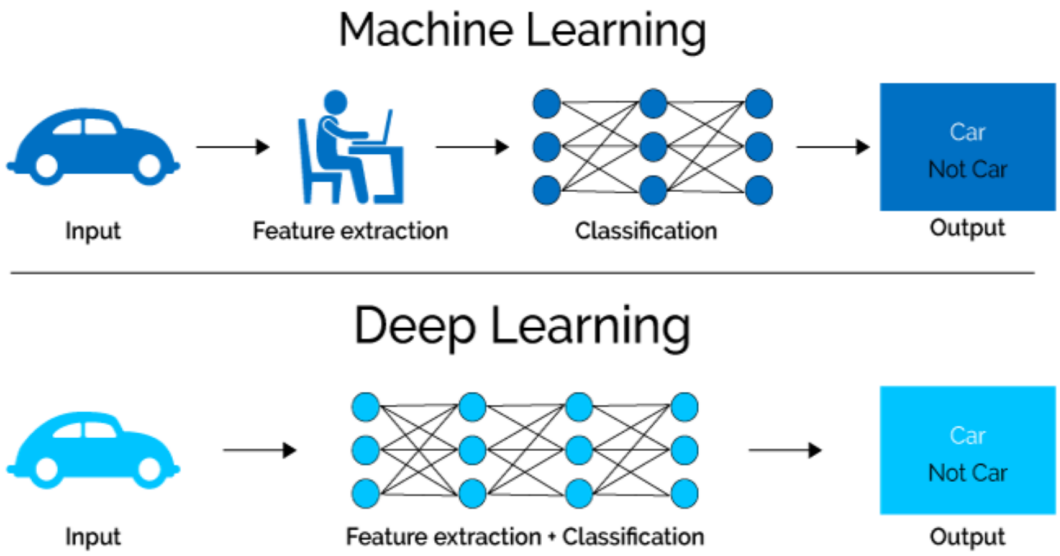

Figure 3.1 Deep learning machine learning difference (Softwaretestinghelp, 2023)

There are many different structures of deep learning algorithms. The most commonly used algorithms are convolutional neural networks, autoencoders, recurrent neural networks, and classical neural networks. In this thesis, we worked on convolutional neural network models.

3.1 Convolutional Neural Networks

These networks, a deep learning model, are created to process multidimensional data. It is commonly used in analyzing visual information. Areas of use include image classification, object detection and classification, vision systems, language processing, disease and abnormality detection. This architecture comprises of two main sections: feature extraction and classification sections, as seen in Figure 3.2. In the feature extraction section, there are convolution and pooling layers. Thanks to these layers, the feature matrix of our data is extracted. In the classification section, classification is performed by applying smoothing, full connection layer, and activation function operations.

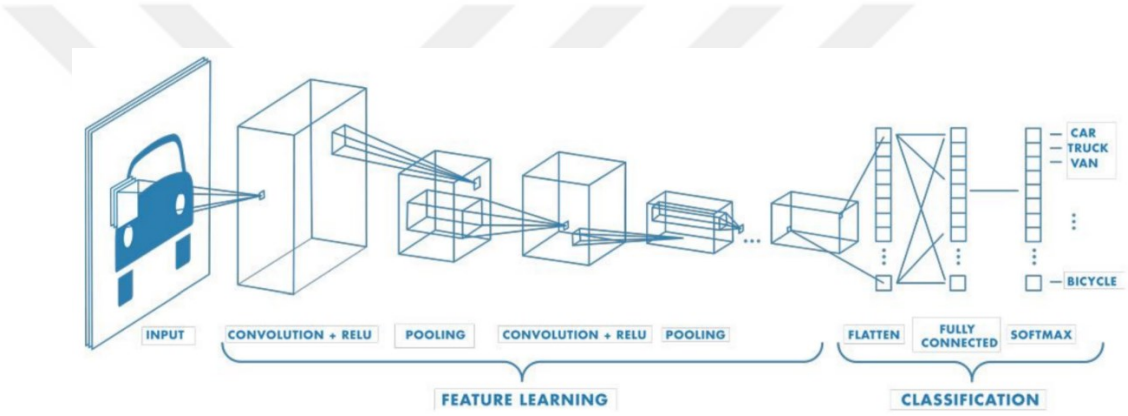

Figure 3.2 Convolutional neural networks layer structure (Mathworks, 2023)

3.1.1 Input Layer

In this layer, which is the first layer of the convolutional neural network, the image to be used in the model is transferred to the network in its raw form. The size of the data in this layer is substantial for the achievement of the current model. If the input data size is selected high, it causes high memory requirement. It also increases the training time and testing time per image. But besides all this, it can increase network success. If the input data size is selected low, it results in low memory requirement. It also shortens the training time and testing time per image. But besides all this, it can reduce the success of the network.

3.1.2 Convolution Layer

In this layer, which is the major building block of the convolutional neural network, the basic feature map of the image data is extracted. By applying filtering operations, high and low level features of the image are determined. In this layer, the output matrix is created by applying the correlation kernel on the matrix of the image data. Edges, stains, curves and smooth areas of the image can be learned in this layer. Generally, filters of size 3x3, 5x5, 7x7 are applied on the image matrix. A mathematical example of the convolution process is shown in Figure 3.3. Here, a 2x2 sized convolution filter was applied to the 4x4 sized image matrix to obtain a 3x3 sized matrix.

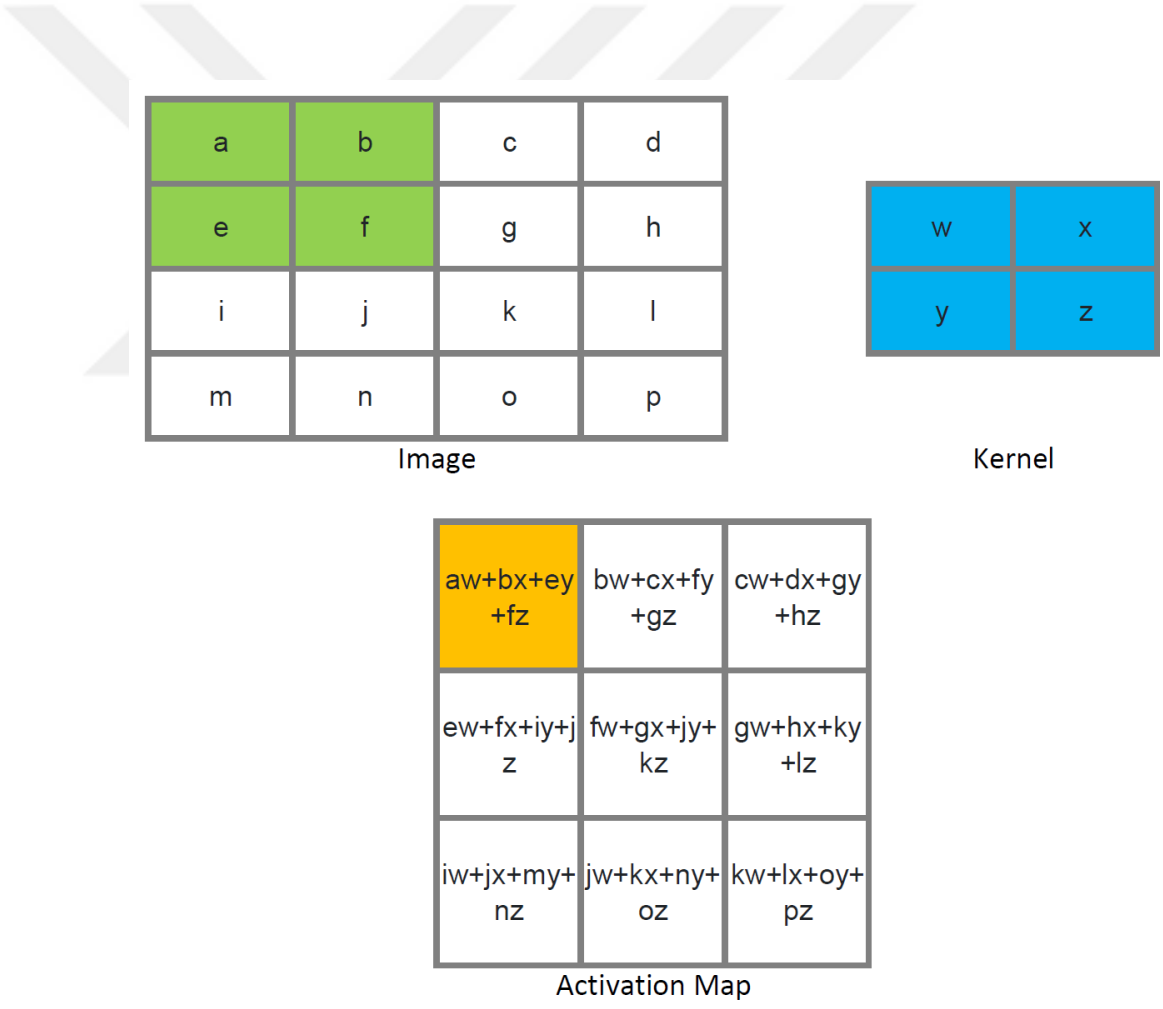

Figure 3.3 Convolution Operation

3.1.3 Pooling Layer

After the activation layer, a pooling layer is applied on the obtained data to reduce the size and processing power of large image data in the network. With this method, the learning process will occur faster and at less cost. In this layer, where no learning process takes place, features continue to be preserved.

There are two species of pooling layer methods. These are maximum pooling and average pooling methods. Maximum pooling is the process of selecting the maximum item in the area enveloped by the filter from the feature map of the image. In this way, the output after the max-pooling layer contains the best salient properties of the previous feature map. Figure 3.4 shows an example of maximum pooling applied to 2x2 sized data on 4x4 sized data (Geeksforgeeks, 2023).

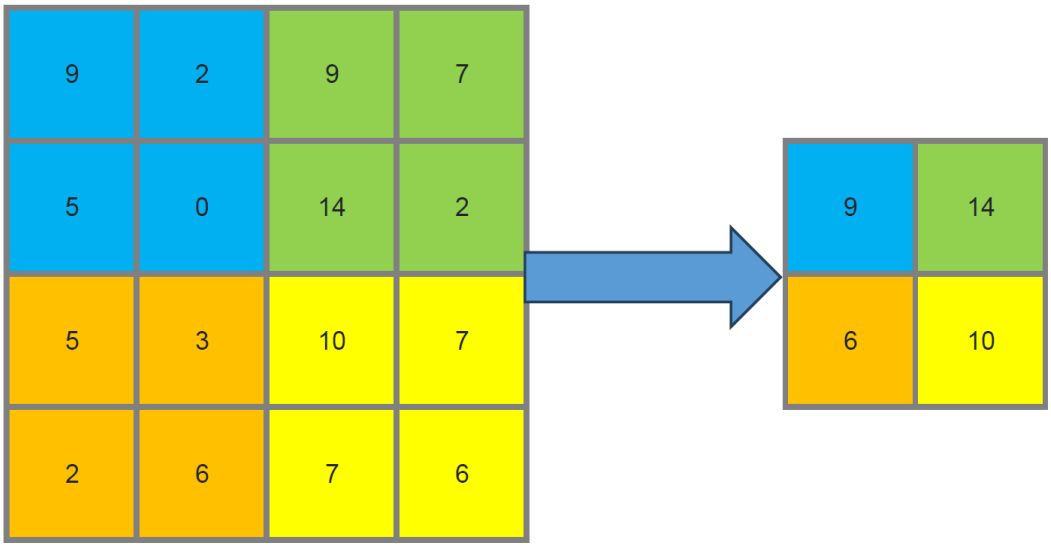

Figure 3.4 Max Pooling

Average pooling is the process of selecting the image from the feature map by averaging the elements in the area enveloped by the filter. In this way, the output after the average pooling layer contains the average features of the previous feature map. Figure 3.5 shows an example of average pooling applied to 2x2 sized data on 4x4 sized data (Geeksforgeeks, 2023).

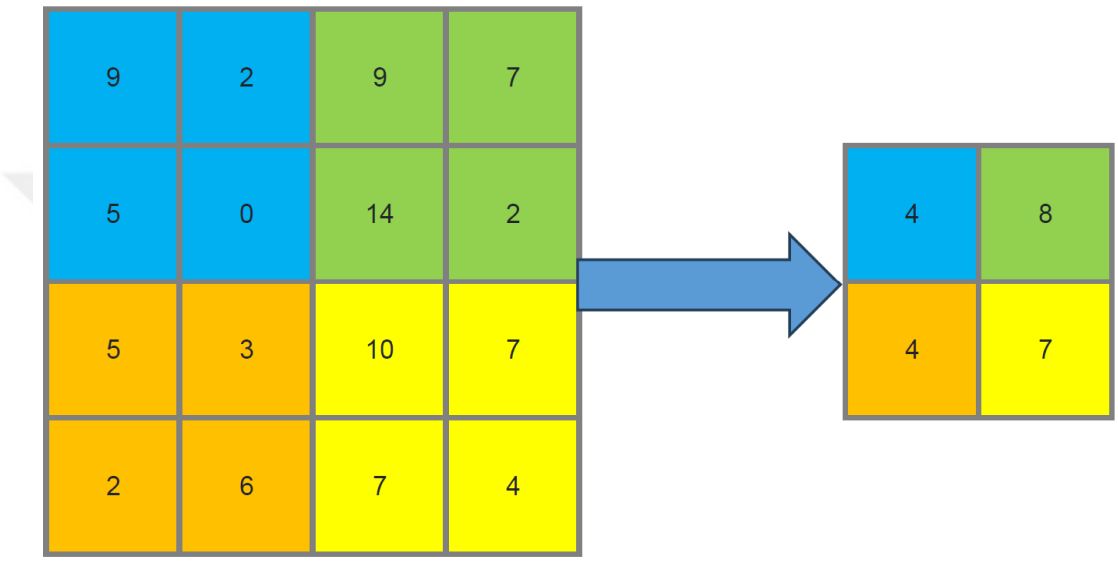

Figure 3.5 Average Pooling

3.1.4 Flattening Layer

The flattening layer takes the multidimensional data produced in the pooling layer and transforms it into one-dimensional data. The one-dimensional data to be processed in the full connection layer is obtained thanks to the flattening layer. Figure 3.6 shows the structure of the flattening layer (Jin et al., 2014).

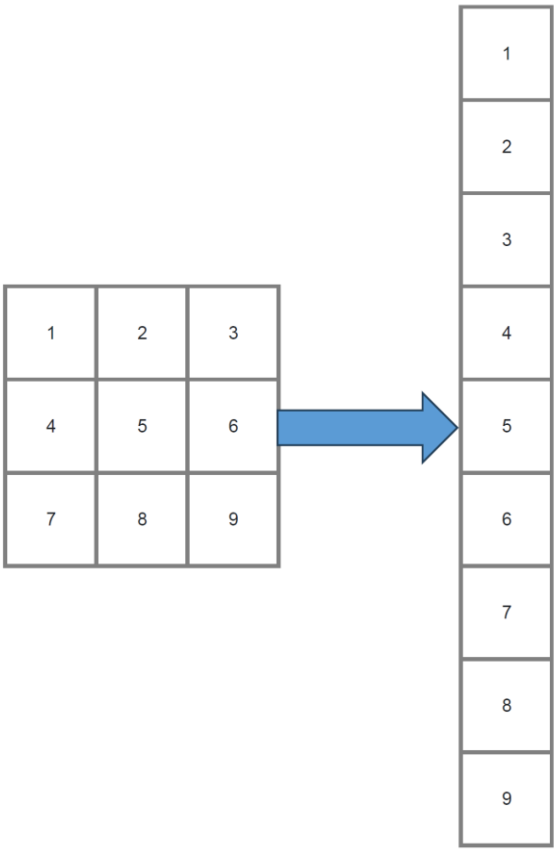

Figure 3.6 The structure of the flatten layer

3.1.5 Fully Connected Layer

After the flattening layer comes the this layer. Every neuron in this layer is linked to the neurons in the following layer by all possible connections. Here, each input data affects the output data. This situation is similar to the traditional neural network layout. This layer contains the most used parameters in the convolutional neural network. For this reason, it requires more a time in training. The structure of the full link layer is shown in Figure 3.7 (Unzueta, 2023).

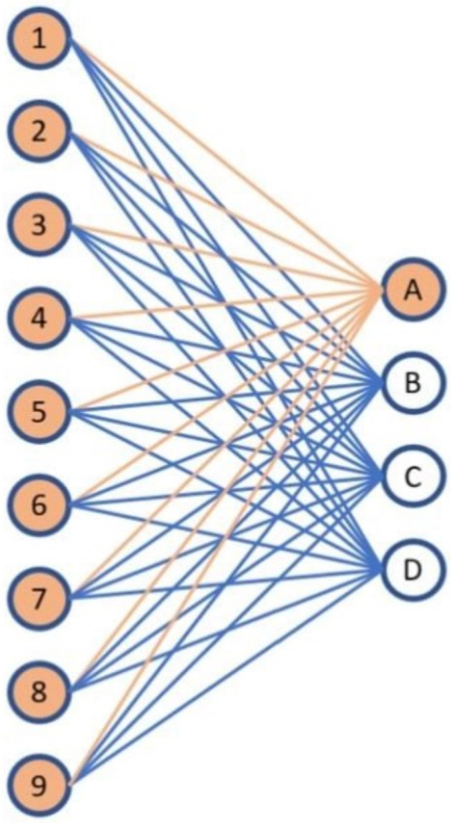

Figure 3.7 Illustration of a fully connected layer (Unzueta, 2023)

3.1.6 Dropout Layer

This layer, which is used to hamper the network from memorizing information, eliminates some random nodes in the network. A situation called overlearning occurs on the test data set due to overtraining of the model. This reduces the success rate and causes the network to memorize. For this reason, this layer is used in the convolutional neural network. Figure 3.8 shows the Standard Neural Network and after applying dropout (Srivastava et al., 2014).

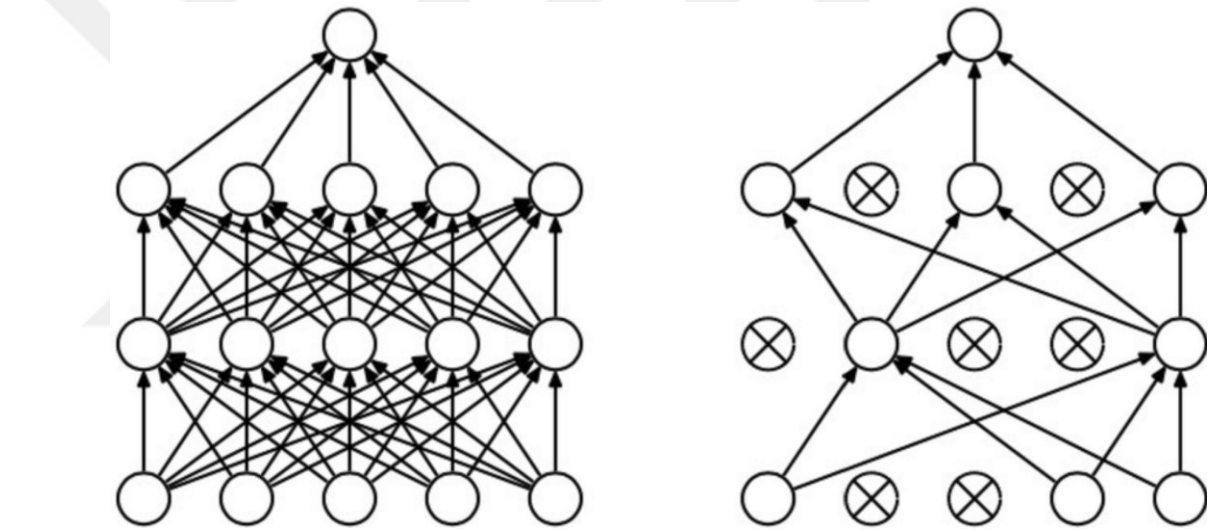

Figure 3.8 Standard Neural Net and After applying dropout (Srivastava et al., 2014)

3.2 Convolutional Neural Network Models

3.2.1 VGGNet

VGGNet is a model introduced by Karen S. and Andrew Z., which investigates the impact of increasing the depth of the convolutional network on trueness by utilizing an architecture with 3×3 dimensional convolution filters (Simonyan and Zisserman, 2014). VGG16 architecture will be used in this thesis. The VGG16 architecture has an input image size of 224x224x3. The number 16 in VGG16 indicates the number of layers. VGG16 architecture consists of 13 convolutional layers, 3 connected full layers. Figure 3.9 shows the layers of VGG16 model architecture.

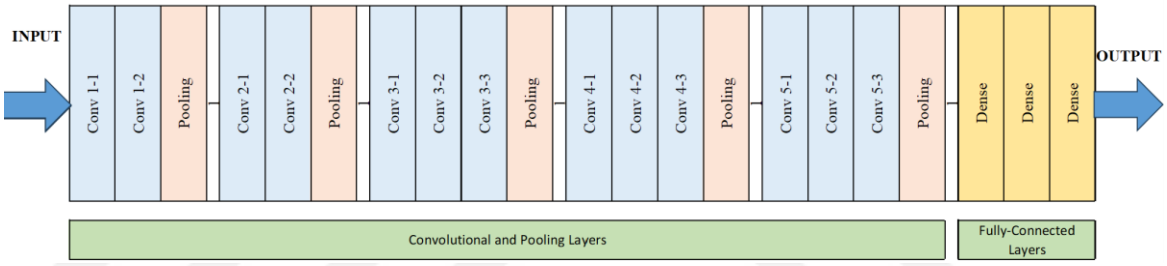

Figure 3.9 VGG16 Model Architecture

There are various VGGNet architectures. The most popular VGGNet architectures are VGG16 and VGG19. Almost similar results were obtained in studies conducted on the ImageNet dataset with VGGNet architectures. Comparison of the performance and features of VGG16 and VGG19 architectures is given in Table 3.1.

Table 3.1 Comparison of VGG16 and VGG19 architectures (Simonyan and Zisserman, 2014)

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528	71.3%	90.1%	138.4M	16
VGG19	549	71.3%	90%	143.7M	19

3.2.2 EfficientNet

EfficientNet is a model introduced by Tan and Le in 2019, aiming to create a more successful model by balancing the width, depth, and resolution of a network (Tan and Le., 2019). Introducing the concept of compound scaling, the EfficientNet model scales width, depth, and resolution dimensions equally. Major building block of the network is the mobile inverted bottleneck (MBConv) layer, first introduced in MobileNetV2 (Sandler et al., 2018). MBConv blocks form a layer that sequentially becomes wide and jams the channels. Thus, the layers are connected by skipping with fewer channels.

EfficientNetB0 architecture will be used in this thesis. EfficientNetB0 architecture has an input image size of 224x224x3. EfficientNetB0 architecture has less depth and parameters compared to other EfficientNet architectures. Distribution of layers and filters of the EfficientNetB0 structure is shown in Figure 3.10.

Figure 3.10 EfficientNetB0 baseline network

EfficientNet models consist of 8 different models, starting with B0 and going up to B7. In studies conducted on the ImageNet dataset, as the model number increases, the number of parameters and depth increase. However, the success rate is also increasing. Comparison of the performance and features of EfficientNet architectures are given in Table 3.2.

Table 3.2 Comparison of EfficientNet architectures (Tan and Le., 2019)

Model	**Size (MB)**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
EfficientNetB0	29	77.1%	93.3%	5.3M	132
EfficientNetB1	31	79.1%	94.4%	7.9M	186
EfficientNetB2	36	80.1%	94.9%	9.2M	186
EfficientNetB3	48	80.6%	95.7%	12.3M	210
EfficientNetB4	75	82.9%	96.4%	19.5M	258
EfficientNetB5	118	83.6%	96.7%	30.6M	312
EfficientNetB6	166	84%	96.8%	43.3M	360
EfficientNetB7	256	84.3%	97%	66.7M	438

3.2.3 ResNet

ResNet, also called Residual Network, is a model inspired by the VGGNet architecture and introduced by He Kaiming and colleagues in 2015 (He et al., 2016). The benefit of ResNet architecture compared toother deep learning models is that its performance does not decrease even as the model deepens.

To avoid the gradient problem seen in traditional and flat neural networks, ResNet architecture emerged. Increasing the number of layers in the models causes training and testing errors. This situation is called the gradient problem. This leads to slow or incomplete learning in models (Sharma, 2023). ResNet architecture consists of residual blocks, which are considered as the main building blocks to solve this problem. In the residual block, input values are taken from one layer and added to the next layer. Residual connections can jump to one or more layers.

ResNet50 architecture will be used in this thesis. ResNet50 architecture has an input image size of 224x224x3. ResNet50, the first developed model, has fewer layers compared to other ResNet architectures. ResNet50 architecture consists of 50 layers. Although the ResNet50 architecture is much deeper than the VGG16 architecture, its model size is 98MB. The layers and filters of ResNet50 model architectures are shown in Figure 3.11.

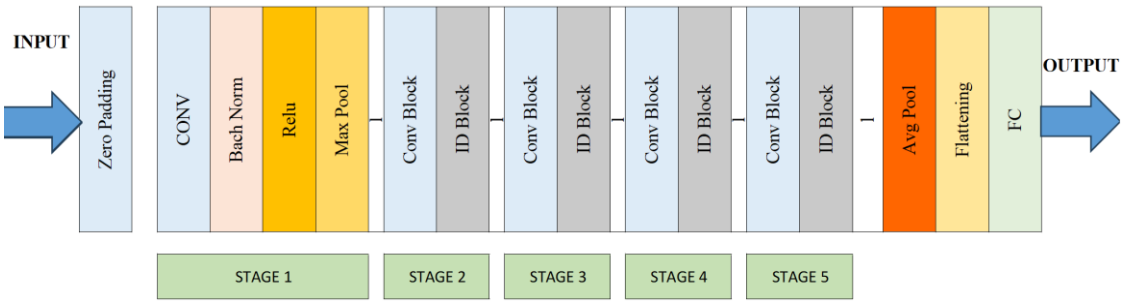

Figure 3.11 ResNet50 Model Architecture

There are various ResNet architectures. These are ResNet50, ResNet50V2, ResNet101, ResNet101V2, ResNet152, ResNet152V2. In studies conducted on the ImageNet dataset, it has been observed that number of layers is directly proportional to the performance, size, and depth of ResNet architectures. Comparison of the performance and features of ResNet architectures is given in Table 3.3.

Table 3.3 Comparison of ResNet architectures (He et al., 2016)

Model	**Size (MB)**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
ResNet50	98	74.9%	92.1%	25.6M	107
ResNet50V2	98	76%	93%	25.6M	103
ResNet101	171	76.4%	92.8%	44.7M	209
ResNet101V2	171	77.2%	93.8%	44.7M	205
ResNet152	232	76.6%	93.1%	60.4M	311
ResNet152V2	232	78%	94.2%	60.4M	307

3.2.4 DenseNet

DenseNet, which stands for Densely Connected Convolutional Networks, is a convolutional neural network proposed by Huang et al. in 2017 (Huang at el., 2017). DenseNet, introduced in the article "Densely Connected Convolutional Networks", has been quite successful in image classification.

This model, proposed to solve the vanishing gradient problem, ensures that every layer takes additional input from all previous layers and conveys its feature maps to the next layers to maximize the information flow.

There are various DenseNet architectures. These are DenseNet121, DenseNet169, and DenseNet201. In studies conducted on the ImageNet dataset, it has been observed that the number of layers is directly proportional to the performance, size and, depth of DenseNet architectures. Comparison of the performance and features of DenseNet architectures is given in Table 3.4.

Table 3.4 Comparison of DenseNet architectures (Huang et al., 2017)

Model	**Size (MB)**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
DenseNet121	33	75%	92.3%	8.1M	242
DenseNet169	57	76.2%	93.2%	14.3M	338
DenseNet201	80	77.3%	93.6%	20.2M	402

DenseNet121 architecture will be used in this thesis. DenseNet121 architecture has an input image size of 224x224x3. DenseNet121, the first developed model, has fewer layers compared to other DenseNet architectures. DenseNet121 architecture consists of 121 layers. The layers and filters of DenseNet121 architecture are shown in Figure 3.12.

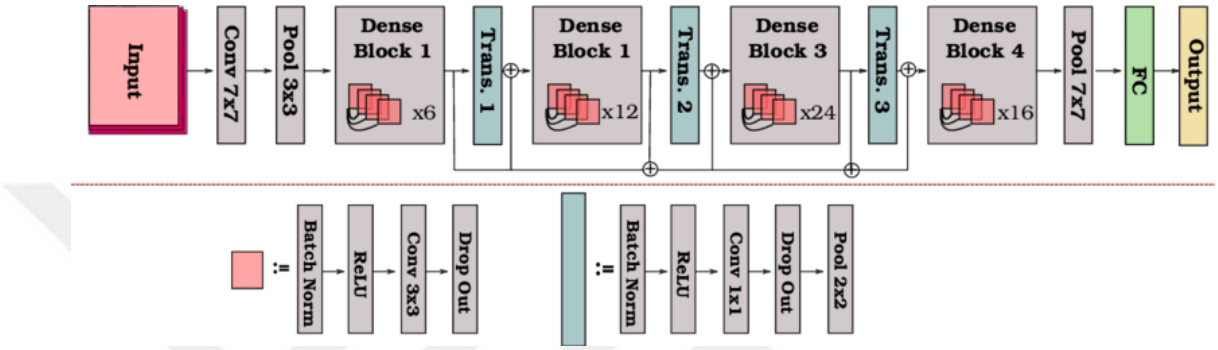

Figure 3.12 DenseNet121 Model Architecture (Radwan, 2019)

3.2.5 MobileNet

MobileNet, designed for mobile and embedded vision applications, was introduced by Howard et al. in 2017 (Howard et al., 2017). MobileNet has been quite successful in object and face classification.

It uses an architecture with depth-separable convolutions to build a low-dimensional and lightweight neural network. Two simple hyperparameters are used, namely width multiplier and stability multiplier, which efficiently balance between latency and accuracy.

There are various MobileNet architectures. These are MobileNet and MobileNetV2. A comparison of the performance and features of MobileNet architectures is given in Table 3.5.

Table 3.5 Comparison of MobileNet architectures (Howard et al., 2017)

MobileNet	**Size (MB)**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
MobileNet	16	70.4%	89.5%	4.3M	55
MobileNetV2	14	71.3%	90.1%	3.5M	105

MobileNet architecture will be used in this thesis. MobileNet architecture has an input image size of 224x224x3. MobileNet, the first developed model, has fewer layers compared to the MobileNetV2 architecture. MobileNet architecture consists of 28 layers. Figure 3.13 shows the layers and filters of the MobileNet architecture.

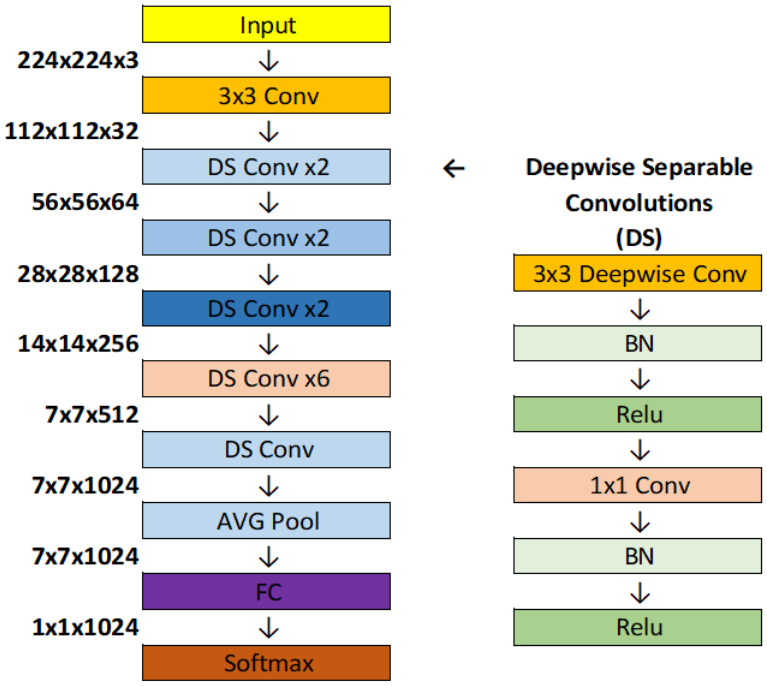

Figure 3.13 MobileNet Model Architecture

3.3 Hyperparameters

In deep learning, there are some parameters that need to be adjusted before starting the training of the model. These parameters directly affect the performance of the model. These are activation function, optimization function, batch size, epoch, and learning rate.

3.3.1 Activation Function

Activation functions convert input signals into an output signal after applying various mathematical operations. The basic function of the activation function is to limit the output value of the neuron and ensure that it operates within a certain range. In this way, it will ensure that the output signal produces a non-linear output. Otherwise, the output signal will produce a linear output and provide limited learning (Nwankpa at el., 2018). The most commonly used activation functions in convolutional neural networks are softmax, tanh, relu, and sigmoid.

Softmax is the activation function that compresses multiple input values to an output value between 0 and 1. Figure 3.14 shows the graph of the Softmax function.

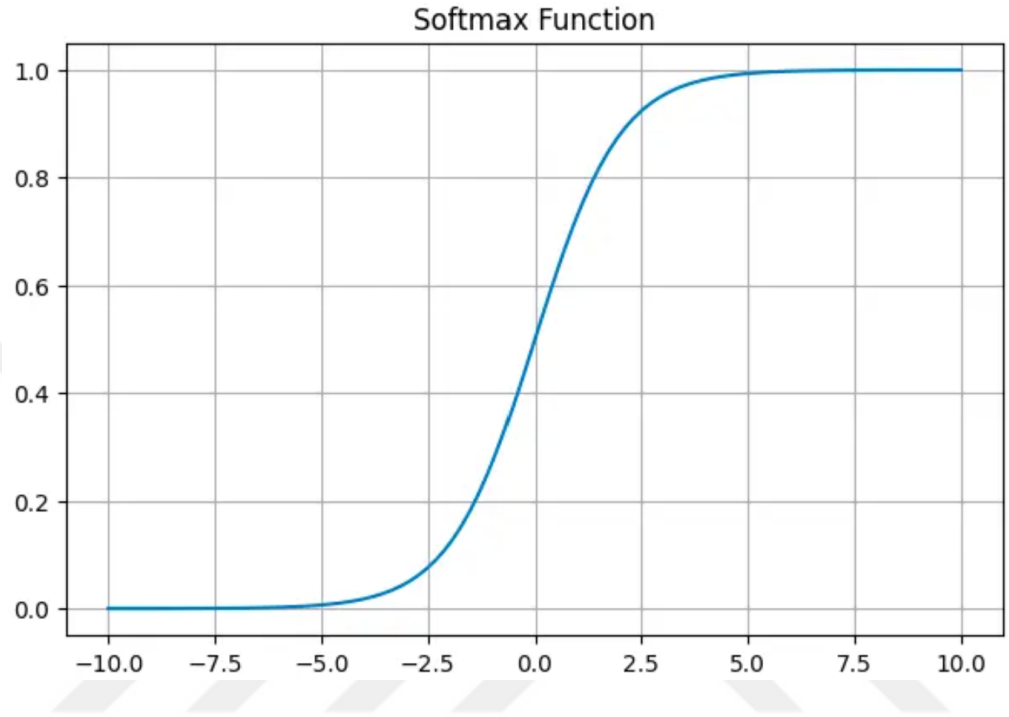

Figure 3.14 Softmax Graph

Tanh is the activation function that produces an output value between -1 and 1. Figure 3.15 shows the graph of the Tanh function.

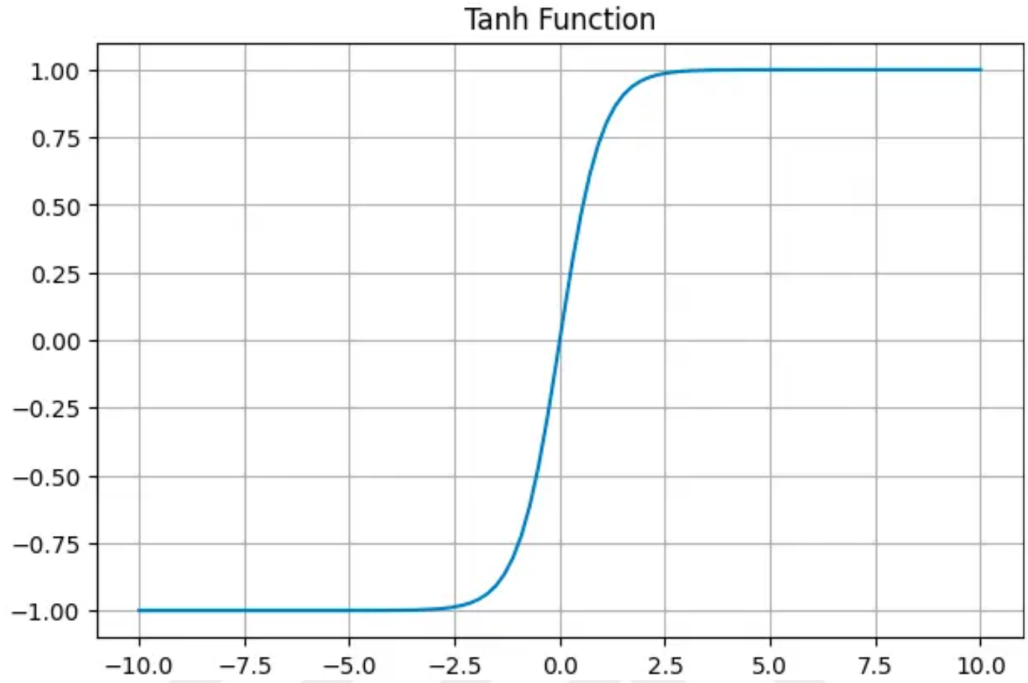

Figure 3.15 Tanh Graph

Relu sets the output value equal to 0 if the input value is less than 0. In other cases, it is the activation function that sets the output value equal to the input value. Figure 3.16 shows the graph of the Relu function.

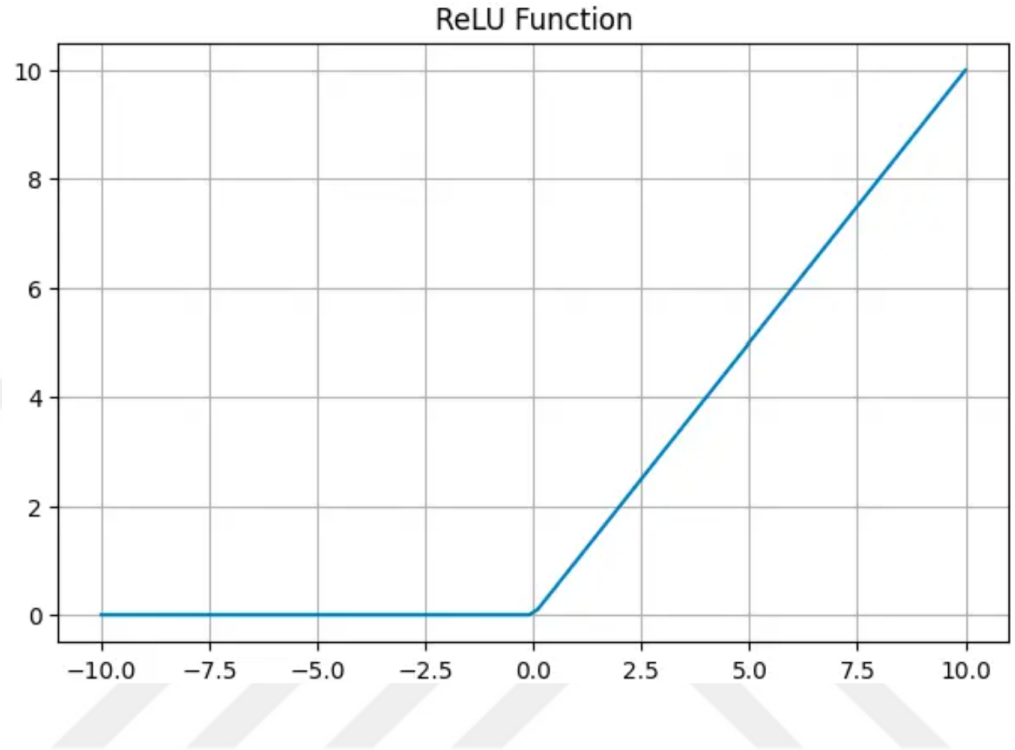

Figure 3.16 Relu Graph

Sigmoid is an activation function that compresses a single input value into an output value between 0 and 1. Figure 3.17 shows the graph of the Sigmoid function.

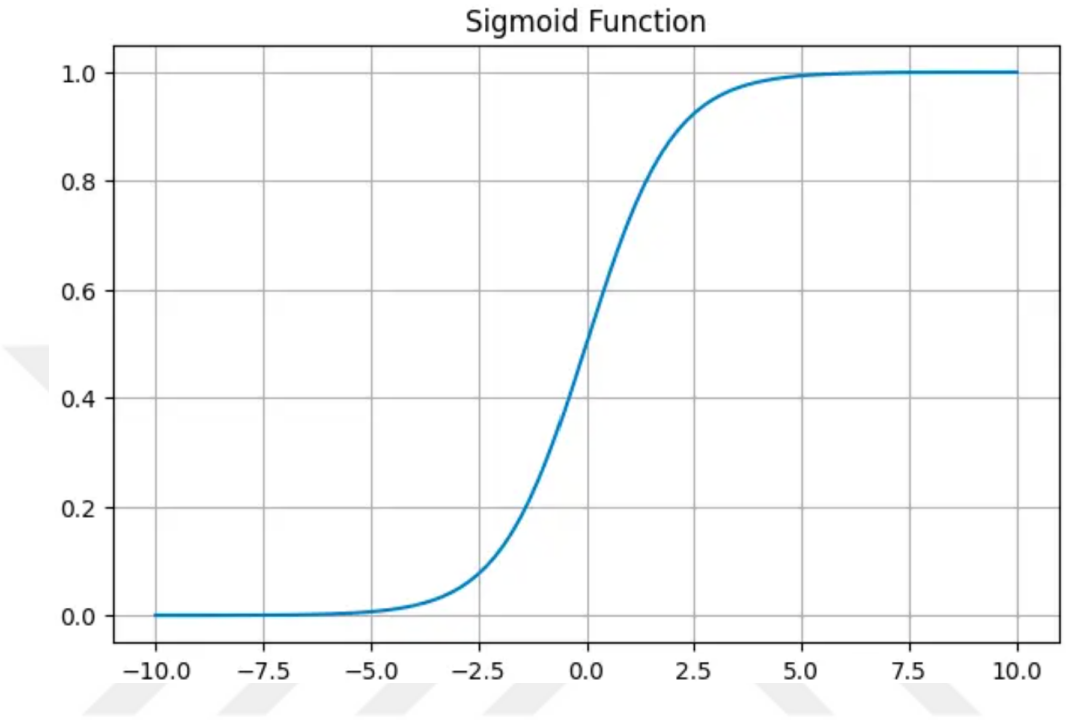

Figure 3.17 Sigmoid Graph

3.3.2 Optimization Function

Optimization functions adjust model parameters to minimize the loss function, which measures how well the model can make predictions on the data set. There are many optimization algorithms available. Which of these will be used can significantly affect the performance of the model. ADAM, stochastic gradient descent, Adagrad, RMSprop, and Adadelta optimization techniques are generally used in deep learning applications (Seyyarer at el., 2020).

ADAM, which is short for adaptive moment estimation, is an optimization algorithm that compounds the advantages of RMSprop and momentum methods. In this thesis study, ADAM algorithm was used.

Stochastic gradient descent (SGD), an optimization algorithm, involves updating parameters based on a small randomly selected dataset rather than the full dataset.

Momentum is the optimization algorithm recommended to reduce the excessive oscillation and increase the speed of reaching the target while searching for the optimum point in SGD.

The optimization algorithm, Adagrad, can work well with small amounts of data, making use of an adaptive learning rate per parameter to eliminate the problem of fixed learning coefficients.

RMSprop was proposed to solve the constant learning coefficient problem, which is a similar problem in Adagrad. However, instead of taking the squares of the gradients, it uses the squares of the momentum gradients.

AdaDelta is an optimization algorithm like RMSProp. The hyperparameter uses the exponentially decaying mean and squares of the gradients to determine the updated scale.

3.3.3 Loss Function

It is a function that measures the success of the model between the actual and predicted value during the training process. The result of the loss function is inversely proportional to the success of the model. The lower the loss function, the more successful the model produced (Erikçi, 2023).

Cross entropy function was used in this study. Cross entropy compares predicted probability values with actual values, producing a probability value whose result varies between 0 and 1. Cross-entropy loss reduces as the estimated possibility value approximates the true value.

3.3.4 Learning Rate

Learning rate is defined as the step size to be taken to reach the optimum point in the loss function. Learning rate was used as 0.00005 for all deep learning architectures in this study.

3.3.5 Batch Size

It is a hyperparameter that divides the data into small pieces and gives it to the network during the training period of the model, instead of giving the entire data set as input to the network. In this study, Batch Size 32 was used for all deep learning architectures.

3.3.6 Epoch

Processing all data once by the model network is called epoch. In this study, epoch 50 was used for all deep learning architectures.

3.4 Evaluation Metrics

The confusion matrix was used to evaluate the productivity of the models used to determine whether the fabric images were normal or faulty. Precision, recall, F1 score, and accuracy values were acquired using the results in the confusion matrix.

3.4.1 Confusion Matrix

With the confusion matrix, comparisons can be made between the actual values of our model and the predicted values. Confusion matrix for binary classification is shown in Figure 3.18.

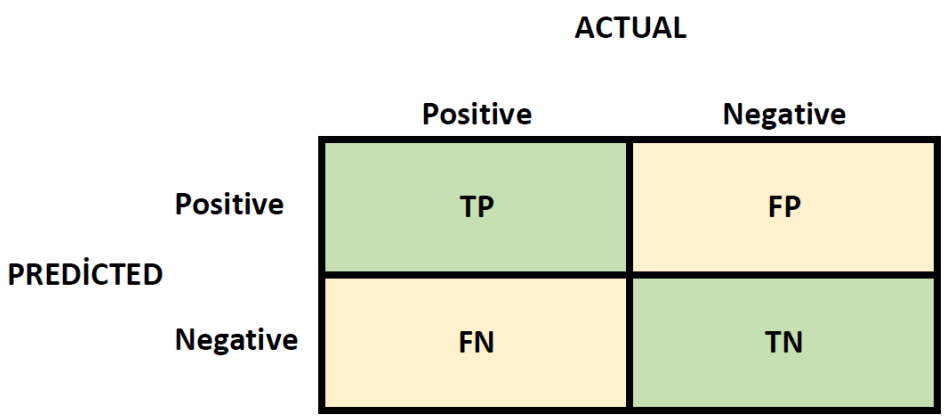

Figure 3.18 Confusion matrix for binary classification

• True Positive (TP): It expresses the numbers that we actually classify as positive and whose predictive value is positive.

• True Negative (TN): It expresses the numbers that we actually classify as negative and whose predictive value is negative.

• False Positive (FP): It expresses the numbers that we actually classify as positive and whose predictive value is negative.

• False Negative (FN): It expresses the numbers that we actually classify as negative and whose predictive value is positive.

3.4.2 Accuracy

The accuracy value indicates how often a classification model is correct. It is calculated by splitting TP and TN values by TP+TN+FP+FN values, as shown in Equation 3.1.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

3.4.3 Precision

The precision value gives the proportion of data classified correctly. It is calculated by dividing the TP value by TP+FP values, as shown in Equation 3.2.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3.2}$$

3.4.4 Recall

The recall value gives the proportion of correctly classified ones consisting of only positive values. It is calculated by dividing the TP value by TP+FN values, as shown in Equation 3.3.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3.3}$$

3.4.5 F1-Score

F1-Score value is figured out by getting the harmonic mean of recall and precision values as shown in Equation 3.4.

$$\text{F1-Score} = \frac{2 \text{ x Precision x Recall}}{\text{Precision + Recall}} \tag{3.4}$$

CHAPTER FOUR
MATERIALS

4.1 Dataset

In this study, ZJU-Leaper fabric dataset was used. The ZJU-Leaper fabric dataset has images with size of 512x512x3. Images were resized to 224x224x3 pixels before being processed in the model. Each fabric type consists of 2 classes: normal and defective. In this study, three different fabrics in Group 1 were studied.

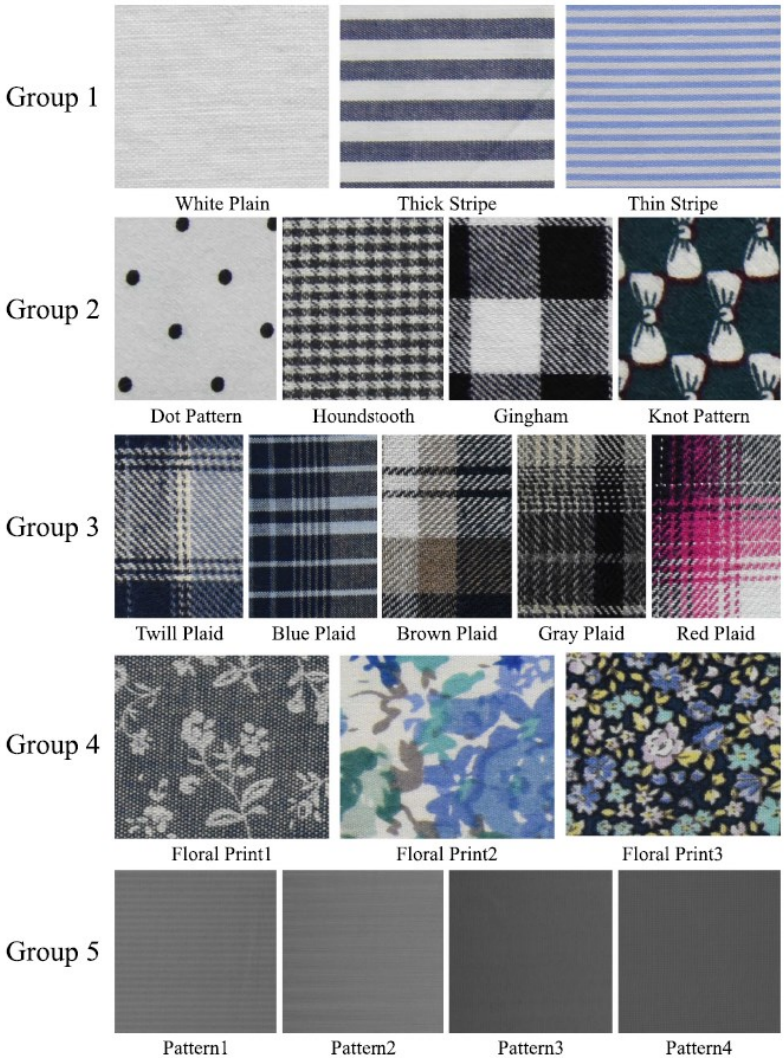

Figure 4.1 ZJU-Leaper Fabric Dataset (Zhang et al., 2020)

ZJU-Leaper fabric dataset contains a total of 19 different fabric samples in 5 particular groups (Zhang et al., 2020). Figure 4.1 shows the fabric samples in the ZJU-Leaper fabric dataset. There are total of 98777 fabric image samples, containing 71127 normal images and 27650 defective images. There are 3 different fabric types in Group 1. There are 4 different fabric types in Group 2. There are 5 different fabric types in Group 3. There are 3 different fabric types in Group 4. There are 4 different fabric types in Group 5.

Figure 4.2 shows fabric samples of the plain white fabric type in Group 1. In white plain fabric, there are a total of 3564 fabric image samples, 2673 normal and 891 defective images.

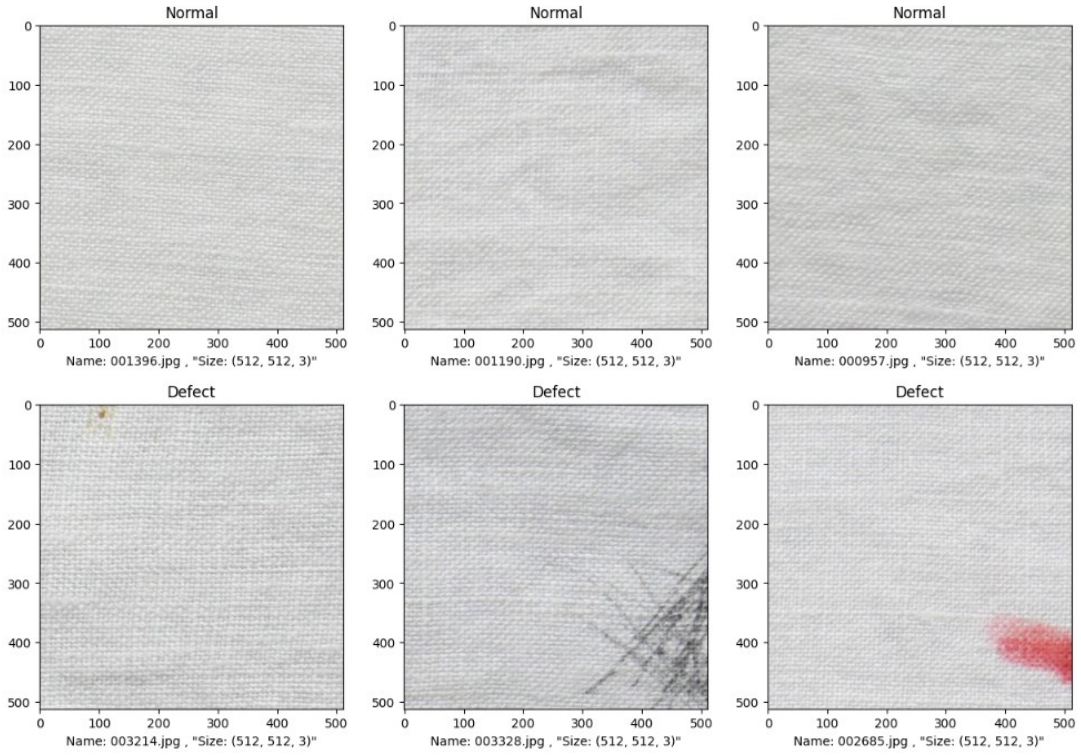

Figure 4.2 White plain fabric samples

Figure 4.3 shows fabric samples of the thick stripe fabric type in Group 1. In the thick stripe fabric, there are a total of 4106 fabric image samples, 3080 normal and 1026 defective images.

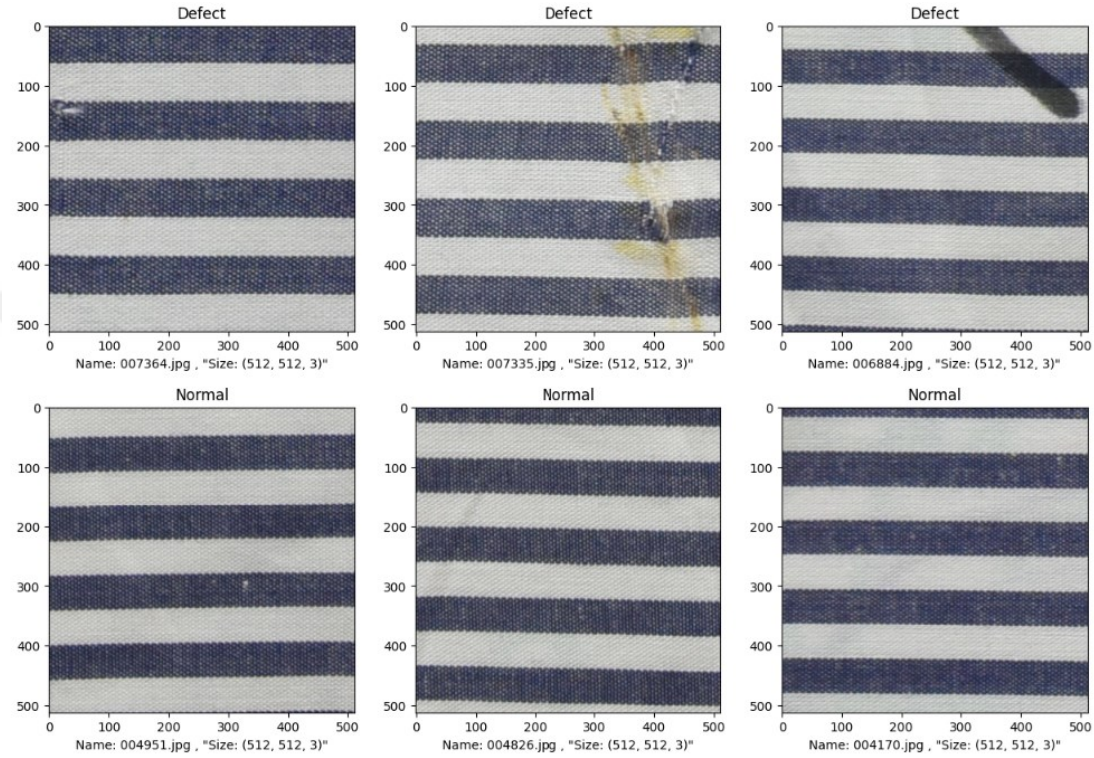

Figure 4.3 Thick stripe fabric samples

Figure 4.4 shows fabric samples of the thin stripe fabric type in Group 1. In the thin stripe fabric, there are a total of 4106 fabric image samples, 3080 normal and 1026 defective images.

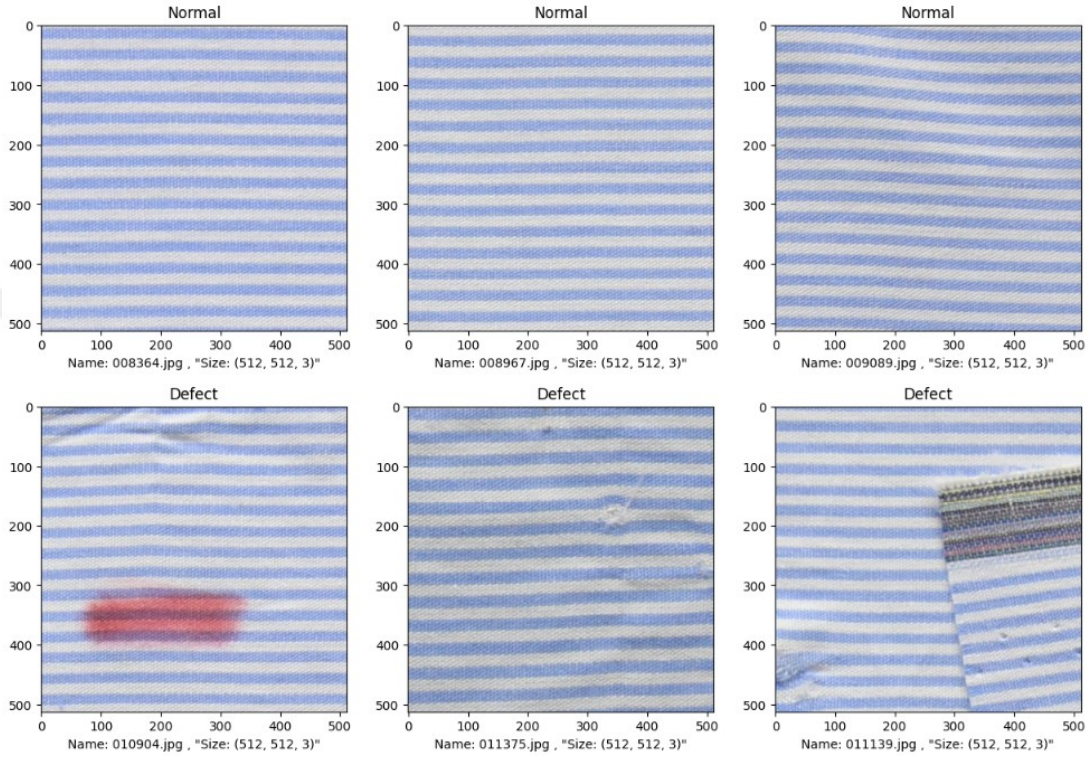

Figure 4.4 Thin stripe fabric samples

4.2 Software and Libraries

In this study, training and testing phases were realized via Google Colab using Python language. Google Colab is a cloud-based programming environment developed for those working on artificial intelligence projects. It has become a very popular tool today because it is free and works without installation. With Google Colab, we can develop free deep learning applications via GPU or TPU. In this study, Google Colab Tesla T4 GPU was used. Tesla T4 GPU has 16 GB GDDR3 memory. You can see the Google Colab working environment in Figure 4.5.

Figure 4.5 Google Colab working environment

Many Python libraries were used in this study, including NumPy, Matplotlib, Pandas, Tensorflow, and Keras. Numpy is a Python library that allows us to easily operate on multidimensional matrices and arrays. Matplotlib is a Python library we use to represent data in graphs. Pandas is a Python library used to analyze and process data. Tensorflow is an open source artificial intelligence library developed by Google. Training and development of deep learning networks takes place thanks to this library. Keras is a library that allows us to work on artificial neural networks with Python, similar to Tensorflow.

CHAPTER FIVE
METHOD AND FINDINGS

The performances of deep learning architectures were evaluated on 3 different fabric types in the ZJU-Leaper data set: plain white, thick stripe, and thin stripe. The images in the data set used are splitted into 2 classes: normal and defective. Training and testing phases were realized with the Tesla T4 GPU in Google Colab. Normal and defective images in the fabric data set were divided into folders as 80% training data and 20% test data.

These images, with dimensions of 512x512x3 pixels, were resized to 224x224x3 pixels and applied as input to convolutional neural network models. In this study, five different convolutional neural networks were studied: VGG16, ResNet50, EfficientNetB0, DenseNet121, and MobileNet.

ImageNet dataset weights, which have millions of images, were used. ADAM was chosen as the optimization algorithm to minimize the error of the model. As a result of the preliminary studies, the most suitable batch size for all architectures was determined to be 32, Epoch 50, and learning rate 0.00005. As a result, the model was trained by selecting the most appropriate hyperparameters.

During the training of the models, graphs showing the progress of accuracy and loss values at each number of cycles were created. Precision, recall, F1-Score, and accuracy results were obtained by creating a confusion matrix to evaluate the performance of convolutional neural networks for normal and defective classes.

The flow model we propose to classify normal and defective images is shown in Figure 5.1.

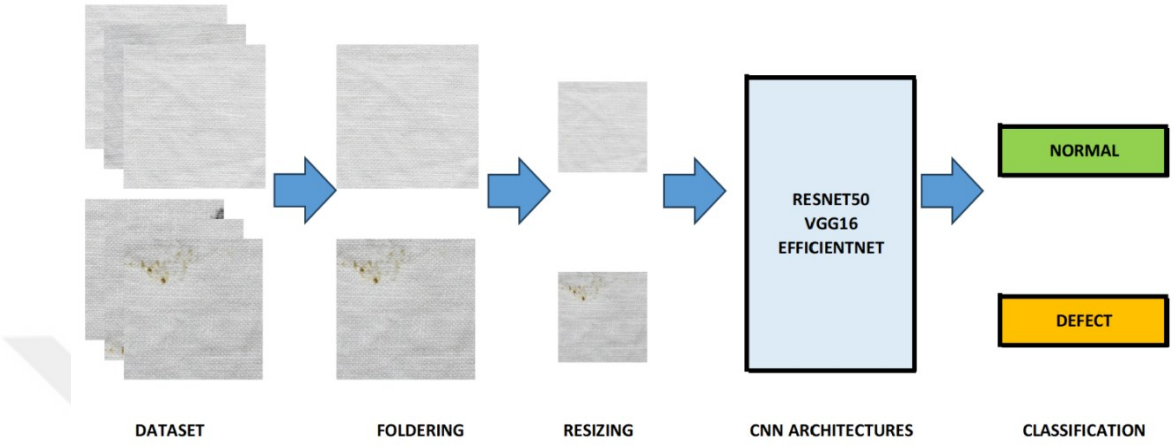

Figure 5.1 Flow Model

5.1 VGG16 Model Test Results

The graph of accuracy and loss values at each epoch number for the white plain fabric data set of the VGG16 model is shown in Figure 5.2.

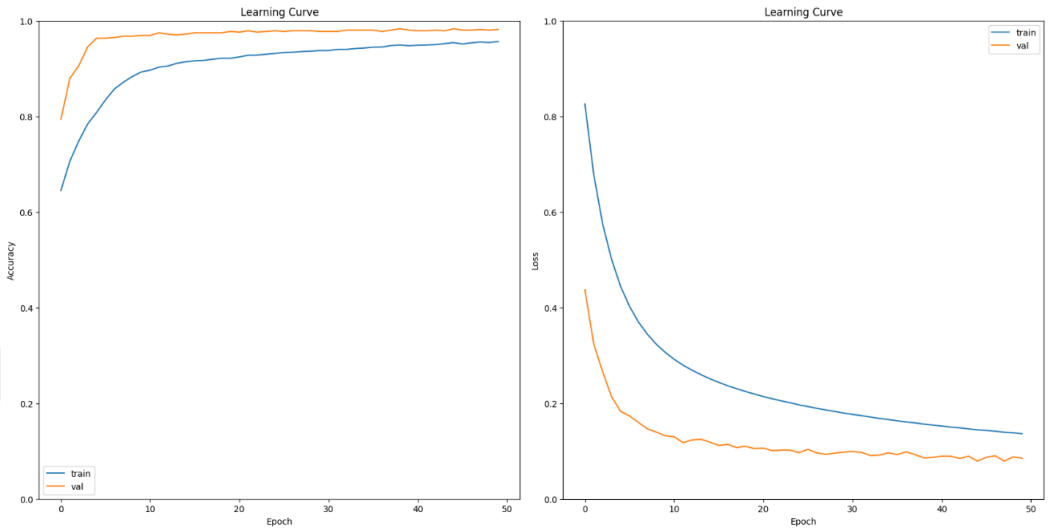

Figure 5.2 VGG16 accuracy and loss graphs (white plain fabric)

When the confusion matrix performance evaluation of the white plain fabric image in Figure 5.3 is examined, in the VGG16 architecture, 522 normal fabrics in the plain white fabric type were detected correctly and 12 normal fabrics were detected incorrectly. 177 defective fabrics were detected correctly, 1 defective fabric was detected incorrectly.

White Plain		
NORMAL	522	12
DEFECT	1	177
	NORMAL	DEFECT

Figure 5.3 Confusion matrix for VGG16 (white plain fabric)

The graph of accuracy and loss values at each epoch number for the thick stripe fabric data set of the VGG16 model is shown in Figure 5.4.

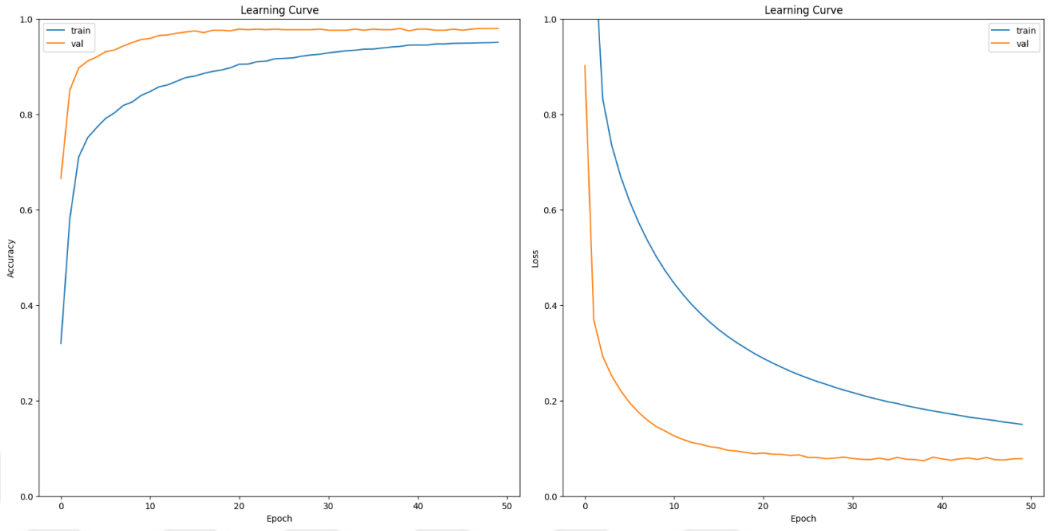

Figure 5.4 VGG16 accuracy and loss graphs (thick stripe fabric)

When the confusion matrix performance evaluation of the thick stripe fabric image in Figure 5.5 is examined, in the VGG16 architecture, 608 normal fabrics in the thick stripe fabric type were detected correctly and 8 normal fabrics were detected incorrectly. 197 defective fabrics were detected correctly, 9 defective fabrics were detected incorrectly.

Thick Stripe		
NORMAL	608	8
DEFECT	9	197
	NORMAL	DEFECT

Figure 5.5 Confusion matrix for VGG16 (thick stripe fabric)

The graph of accuracy and loss values at each epoch number for the thin stripe fabric data set of the VGG16 model is shown in Figure 5.6.

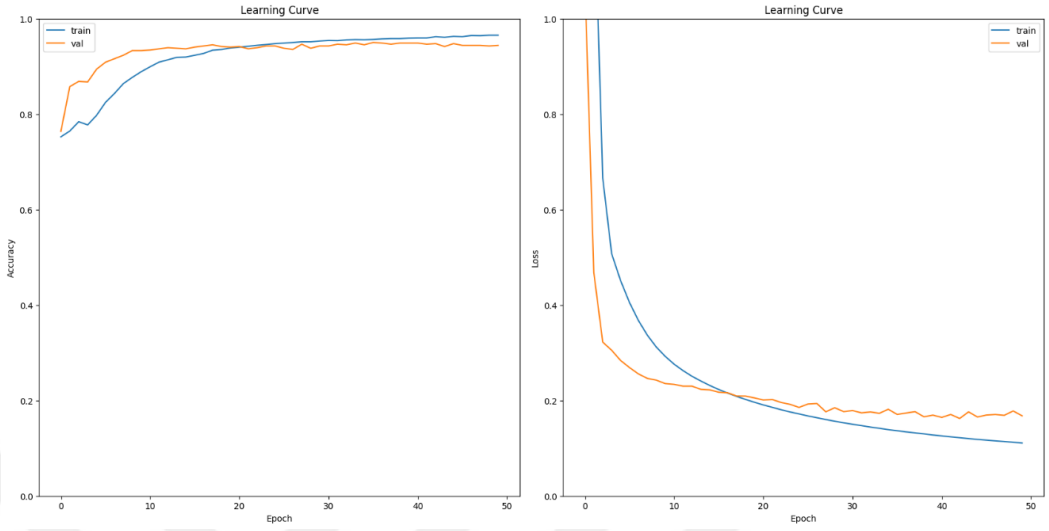

Figure 5.6 VGG16 accuracy and loss graphs (thin stripe fabric)

When the confusion matrix performance evaluation of the thin stripe fabric image in Figure 5.7 is examined, in the VGG16 architecture, 585 normal fabrics in the thin stripe fabric type were detected correctly and 31 normal fabrics were detected incorrectly. 194 defective fabrics were detected correctly, 12 defective fabrics were detected incorrectly.

Thin Stripe		
NORMAL	585	31
DEFECT	12	194
	NORMAL	DEFECT

Figure 5.7 Confusion matrix for VGG16 (thin stripe fabric)

5.2 EfficientNetB0 Model Test Results

The graph of accuracy and loss values at each epoch number for the white plain fabric data set of the EfficientNetB0 model is shown in Figure 5.8.

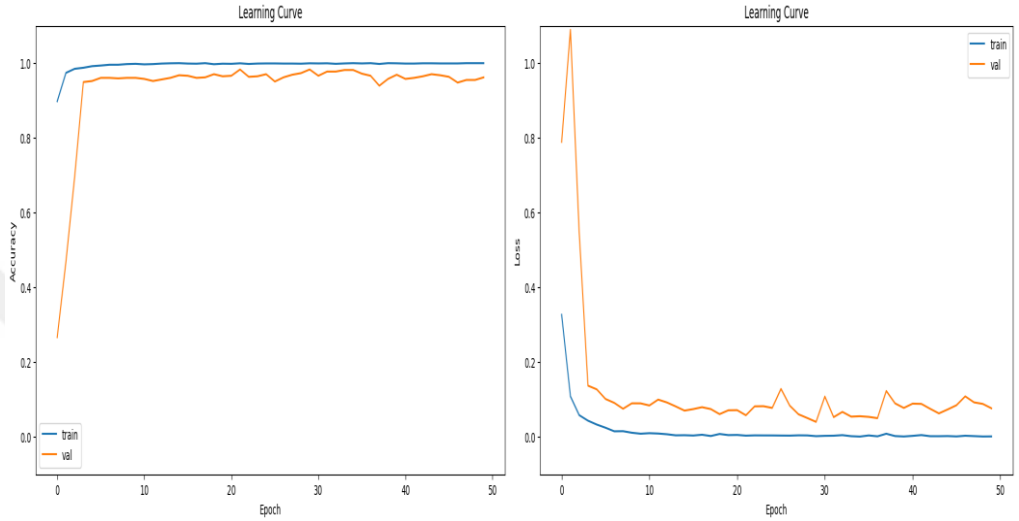

Figure 5.8 EfficientNetB0 accuracy and loss graphs (white plain fabric)

When the confusion matrix performance evaluation of the white plain fabric image in Figure 5.9 is examined, in the EfficientNetB0 architecture, 531 normal fabrics of the plain white fabric type were detected correctly and 3 normal fabrics were detected incorrectly. 169 defective fabrics were detected correctly, 9 defective fabrics were detected incorrectly.

White Plain		
NORMAL	531	3
DEFECT	9	169
	NORMAL	DEFECT

Figure 5.9 Confusion matrix for EfficientNetB0 (white plain fabric)

The graph of accuracy and loss values at each epoch number for the thick stripe fabric data set of the EfficientNetB0 model is shown in Figure 5.10.

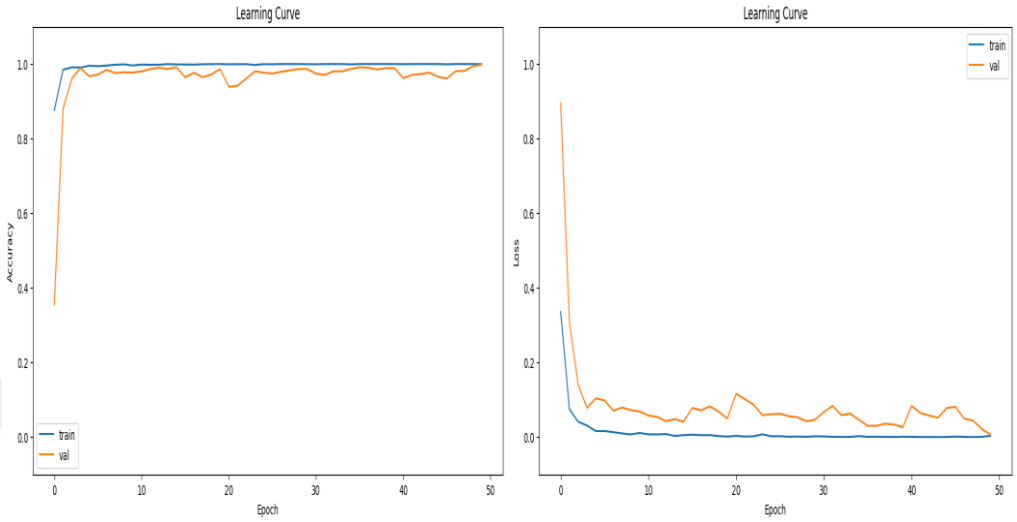

Figure 5.10 EfficientNetB0 accuracy and loss graphs (thick stripe fabric)

When the confusion matrix performance evaluation of the thick stripe fabric image in Figure 5.11 is examined, in the EfficientNetB0 architecture, 615 normal fabrics in the thick stripe fabric type were detected correctly and 1 normal fabric was detected incorrectly. All 206 defective fabric types were detected correctly.

Thick Stripe		
NORMAL	615	1
DEFECT	0	206
	NORMAL	DEFECT

Figure 5.11 Confusion matrix for EfficientNetB0 (thick stripe fabric)

The graph of accuracy and loss values at each epoch number for the thin stripe fabric data set of the EfficientNetB0 model is shown in Figure 5.12.

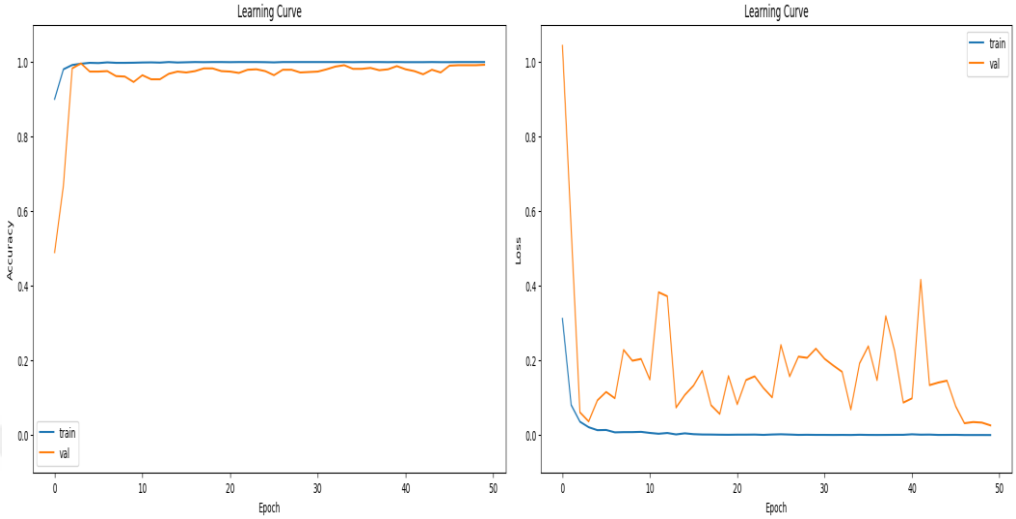

Figure 5.12 EfficientNetB0 accuracy and loss graphs (thin stripe fabric)

When the confusion matrix performance evaluation of the thin stripe fabric image in Figure 5.13 is examined, in the EfficientNetB0 architecture, 613 normal fabrics in the thin stripe fabric type were detected correctly and 3 normal fabrics were detected incorrectly. 203 defective fabrics were detected correctly, 3 defective fabrics were detected incorrectly.

Thin Stripe		
NORMAL	613	3
DEFECT	3	203
	NORMAL	DEFECT

Figure 5.13 Confusion matrix for EfficientNetB0 (thin stripe fabric)

5.3 ResNet50 Model Test Results

The graph of accuracy and loss values at each epoch number for the white plain fabric data set of the ResNet50 model is shown in Figure 5.14.

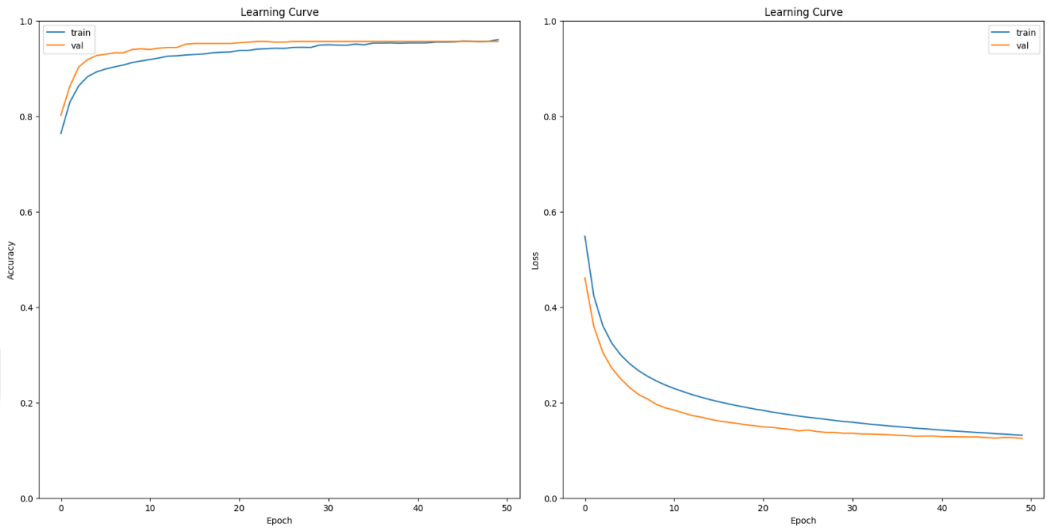

Figure 5.14 ResNet50 accuracy and loss graphs (white plain fabric)

When the confusion matrix performance evaluation of the white plain fabric image in Figure 5.15 is examined, in the ResNet50 architecture, 533 normal fabrics of the white plain fabric type were detected correctly and 1 normal fabric was detected incorrectly. 148 defective fabrics were detected correctly, 30 defective fabrics were detected incorrectly.

White Plain		
NORMAL	533	1
DEFECT	30	148
	NORMAL	DEFECT

Figure 5.15 Confusion matrix for ResNet50 (white plain fabric)

The graph of accuracy and loss values at each epoch number for the thick stripe fabric data set of the ResNet50 model is shown in Figure 5.16.

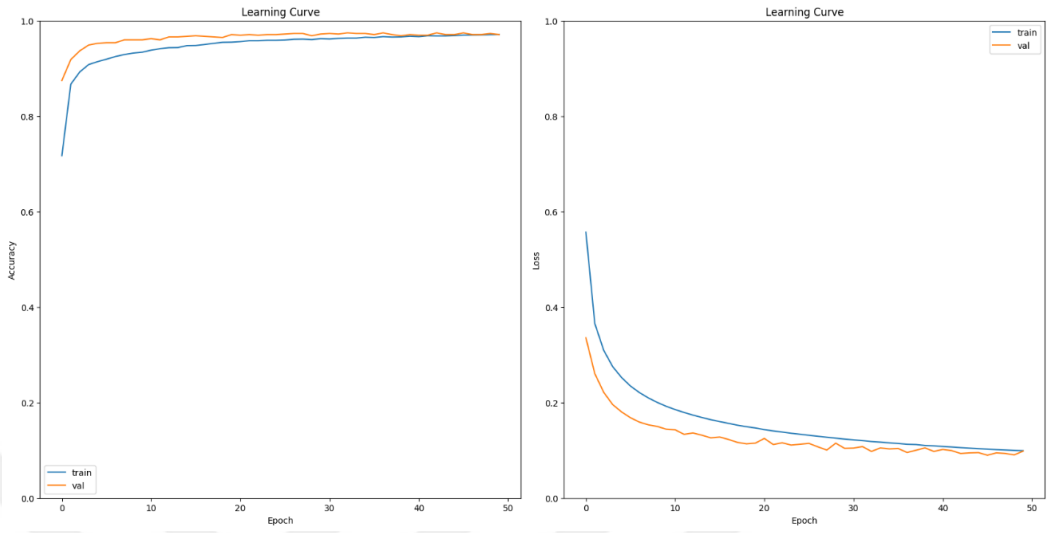

Figure 5.16 ResNet50 accuracy and loss graphs (thick stripe fabric)

When the confusion matrix performance evaluation of the thick stripe fabric image in Figure 5.17 is examined, in the ResNet50 architecture, 604 normal fabrics in the thick stripe fabric type were detected correctly and 12 normal fabrics were detected incorrectly. 197 defective fabrics were detected correctly, 9 defective fabrics were detected incorrectly.

Thick Stripe		
NORMAL	604	12
DEFECT	9	197
	NORMAL	DEFECT

Figure 5.17 Confusion matrix for ResNet50 (thick stripe fabric)

The graph of accuracy and loss values at each epoch number for the thin stripe fabric data set of the ResNet50 model is shown in Figure 5.18.

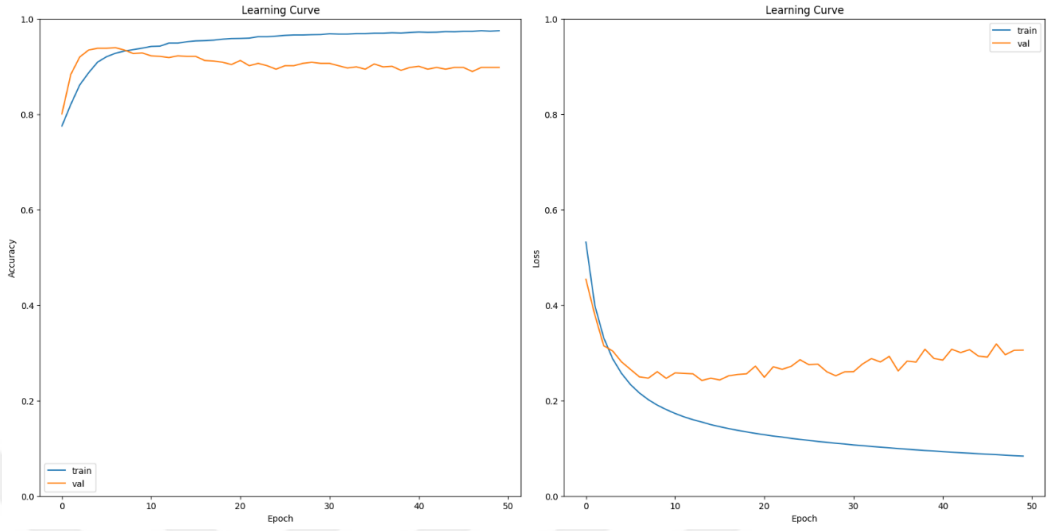

Figure 5.18 ResNet50 accuracy and loss graphs (thin stripe fabric)

When the confusion matrix performance evaluation of the thin stripe fabric image in Figure 5.19 is examined, in the ResNet50 architecture, 552 normal fabrics in the thin stripe fabric type were detected correctly and 64 normal fabrics were detected incorrectly. All 206 defective fabric types were detected correctly.

Thin Stripe		
NORMAL	552	64
DEFECT	0	206
	NORMAL	DEFECT

Figure 5.19 Confusion matrix for ResNet50 (thin stripe fabric)

5.4 DenseNet121 Model Test Results

The graph of accuracy and loss values at each epoch number for the white plain fabric data set of the DenseNet121 model is shown in Figure 5.20.

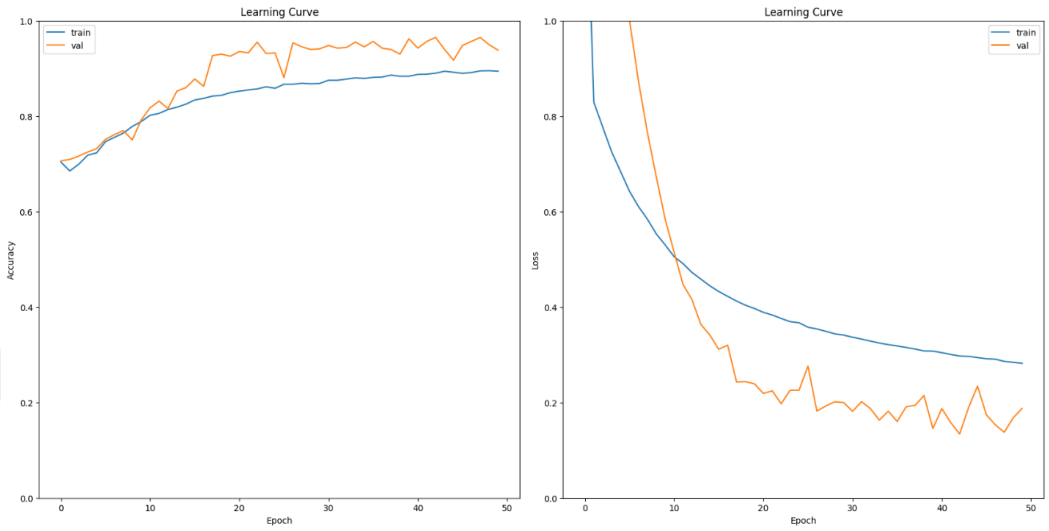

Figure 5.20 DenseNet121 accuracy and loss graphs (white plain fabric)

When the confusion matrix performance evaluation of the white plain fabric image in Figure 5.21 is examined, in the DenseNet121 architecture, 516 normal fabrics of the white plain fabric type were detected correctly and 18 normal fabrics were detected incorrectly. 171 defective fabrics were detected correctly, 7 defective fabrics were detected incorrectly.

White Plain		
NORMAL	516	18
DEFECT	7	171
	NORMAL	DEFECT

Figure 5.21 Confusion matrix for DenseNet121 (white plain fabric)

The graph of accuracy and loss values at each epoch number for the thick stripe fabric data set of the DenseNet121 model is shown in Figure 5.22.

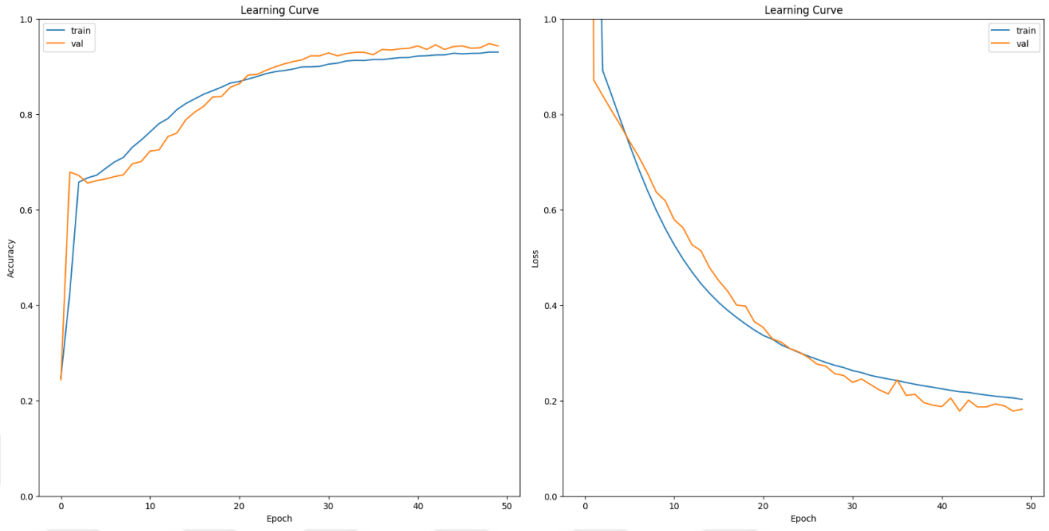

Figure 5.22 DenseNet121 accuracy and loss graphs (thick stripe fabric)

When the confusion matrix performance evaluation of the thick stripe fabric image in Figure 5.23 is examined, in the DenseNet121 architecture, 601 normal fabrics in the thick stripe fabric type were detected correctly and 15 normal fabrics were detected incorrectly. 176 defective fabrics were detected correctly, 30 defective fabrics were detected incorrectly.

Thick Stripe		
NORMAL	601	15
DEFECT	30	176
	NORMAL	DEFECT

Figure 5.23 Confusion matrix for DenseNet121 (thick stripe fabric)

The graph of accuracy and loss values at each epoch number for the thin stripe fabric data set of the DenseNet121 model is shown in Figure 5.24.

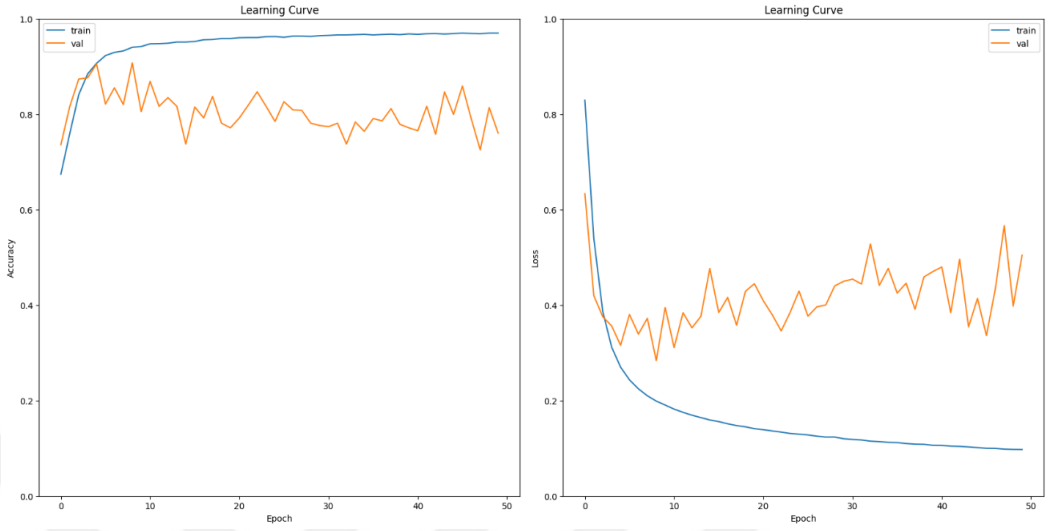

Figure 5.24 DenseNet121 accuracy and loss graphs (thin stripe fabric)

When the confusion matrix performance evaluation of the thin stripe fabric image in Figure 5.25 is examined, in the DenseNet121 architecture, 540 normal fabrics in the thin stripe fabric type were detected correctly and 76 normal fabrics were detected incorrectly. All 206 defective fabric types were detected correctly.

Thin Stripe		
NORMAL	540	76
DEFECT	0	206
	NORMAL	DEFECT

Figure 5.25 Confusion matrix for DenseNet121 (thin stripe fabric)

5.5 MobileNet Model Test Results

The graph of accuracy and loss values at each epoch number for the white plain fabric data set of the MobileNet model is shown in Figure 5.26.

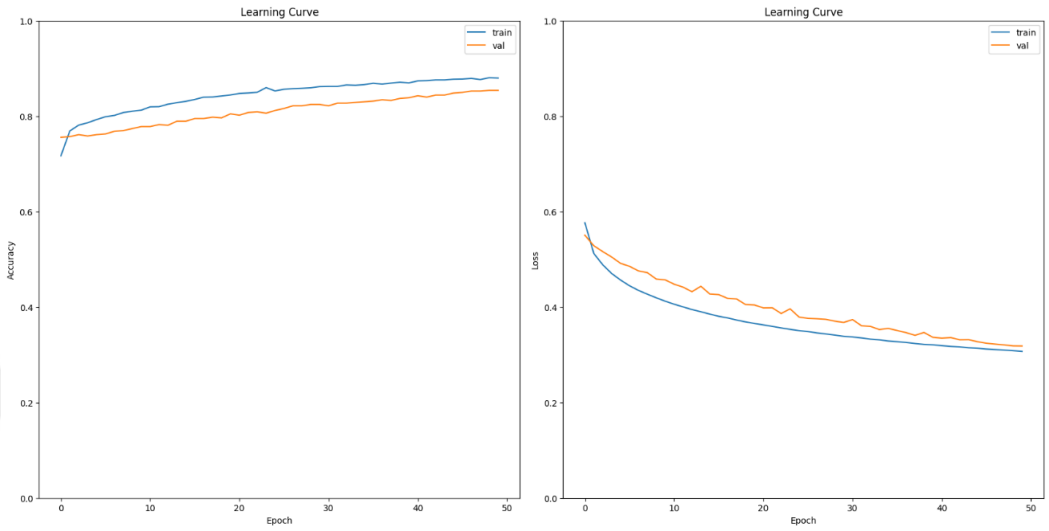

Figure 5.26 MobileNet accuracy and loss graphs (white plain fabric)

When the confusion matrix performance evaluation of the white plain fabric image in Figure 5.27 is examined, in the MobileNet architecture, 519 normal fabrics of the white plain fabric type were detected correctly and 15 normal fabrics were detected incorrectly. 89 defective fabrics were detected correctly, 89 defective fabrics were detected incorrectly.

White Plain		
NORMAL	519	15
DEFECT	89	89
	NORMAL	DEFECT

Figure 5.27 Confusion matrix for MobileNet (white plain fabric)

The graph of accuracy and loss values at each epoch number for the thick stripe fabric data set of the MobileNet model is shown in Figure 5.28.

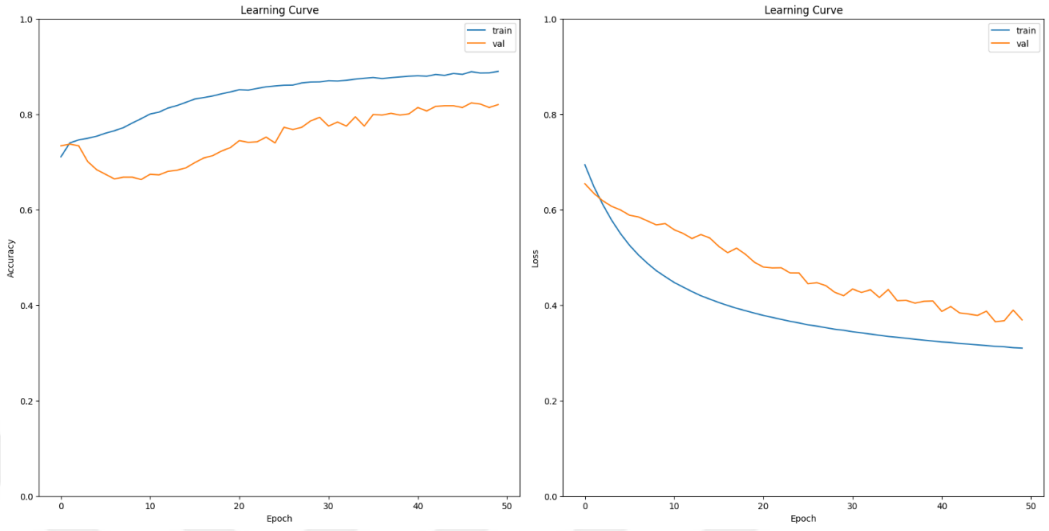

Figure 5.28 MobileNet accuracy and loss graphs (thick stripe fabric)

When the confusion matrix performance evaluation of the thick stripe fabric image in Figure 5.29 is examined, in the MobileNet architecture, 502 normal fabrics in the thick stripe fabric type were detected correctly and 114 normal fabrics were detected incorrectly. 175 defective fabrics were detected correctly, 31 defective fabrics were detected incorrectly.

Thick Stripe		
NORMAL	502	114
DEFECT	31	175
	NORMAL	DEFECT

Figure 5.29 Confusion matrix for MobileNet (thick stripe fabric)

The graph of accuracy and loss values at each epoch number for the thin stripe fabric data set of the MobileNet model is shown in Figure 5.30.

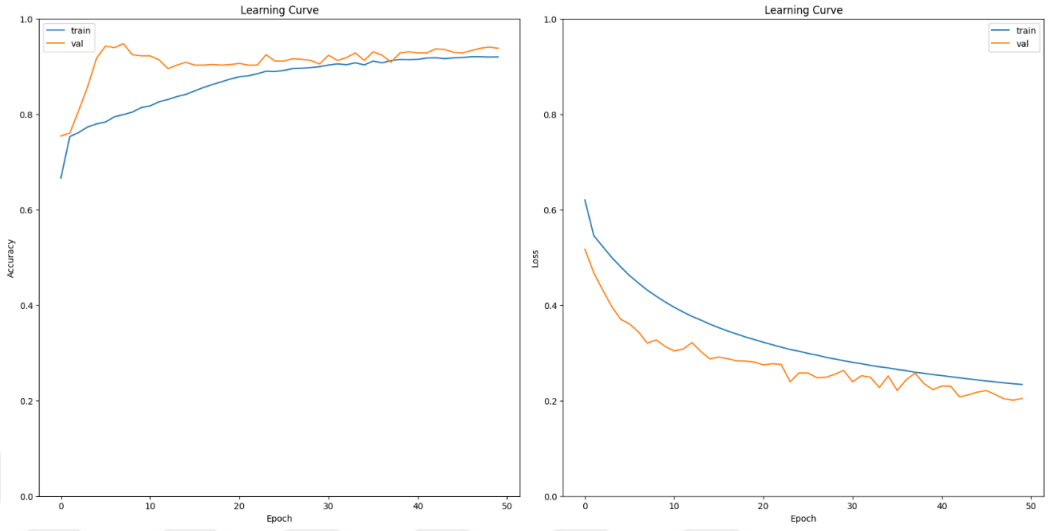

Figure 5.30 MobileNet accuracy and loss graphs (thin stripe fabric)

When the confusion matrix performance evaluation of the thin stripe fabric image in Figure 5.31 is examined, in the MobileNet architecture, 567 normal fabrics in the thin stripe fabric type were detected correctly and 49 normal fabrics were detected incorrectly. All 206 defective fabric types were detected correctly.

Thin Stripe		
NORMAL	567	49
DEFECT	0	206
	NORMAL	DEFECT

Figure 5.31 Confusion matrix for MobileNet (thin stripe fabric)

Precision, recall, F1-Score, and accuracy results of all models used within the scope of the thesis are shown in Table 5.1.

Table 5.1 General performance values of the models

		Precision	**Recall**	**F1-score**	**Accuracy**
VGG16	**White Plain**	98,6	96,73	97,61	98,17
	Thick Stripe	97,17	97,32	97,24	97,93
	Thin Stripe	94,57	92,11	93,24	94,77
EfficientNet	**White Plain**	97,19	98,29	97,73	98,31
	Thick Stripe	99,92	99,76	99,84	99,88
	Thin Stripe	99,03	99,03	99,03	99,27
ResNet50	**White Plain**	91,48	97	93,85	95,65
	Thick Stripe	96,84	96,40	96,62	97,45
	Thin Stripe	94,81	88,15	90,54	92,21
DenseNet121	**White Plain**	96,35	94,57	95,41	96,49
	Thick Stripe	91,5	93,7	92,53	94,53
	Thin Stripe	93,83	86,52	88,93	90,75
MobileNet	**White Plain**	73,6	85,47	77,01	85,39
	Thick Stripe	83,22	77,37	79,04	82,36
	Thin Stripe	96,02	90,39	92,61	94,04

5.6 Discussion

As far as the results presented in Table 5.1, the best precision, recall, F1-score, and accuracy values of the VGG16 model were in the white plain fabric type with 98.6%, 96.73%, 97.61%, 98.17%, respectively. The best precision, recall, F1-score, and accuracy values of the EfficientNetB0 model were in the thick stripe fabric type with 99.92%, 99.76%, 99.84%, 99.88%, respectively. The best precision, recall, F1-score, and accuracy values of the ResNet50 model were in the thick stripe fabric type with 96.84%, 96.40%, 96.62%, 97.45%, respectively. The best precision, recall, F1-score, and accuracy values of the DenseNet121 model were in the white plain fabric type with 96.35%, 94.57%, 95.41%, 96.49%, respectively. The best precision, recall, F1-score, and accuracy values of the MobileNet model were in the thin stripe fabric type with 96.02%, 90.39%, 92.61%, 94.04%, respectively.

Additionally, when Table 5.1 is examined, it is seen that different models are successful in 3 different fabric types.

In general, when the performances of the models on 3 different fabric types were examined, it was observed that the most successful model was EfficientNetB0. The performance of the EfficientNetB0 model was almost 99% on all fabric types.

CHAPTER SIX
CONCLUSION AND FUTURE WORK

In this thesis study, fabric classification was made on normal and defective fabric types using deep learning methods. The studies carried out within the scope of the thesis are aimed at minimizing fabric defects caused by human or machine errors in fabric defect detection.

A study was conducted on white plain, thick stripe, and thin stripe fabric types in the ZJU-Leaper data set. In the study conducted on normal and defective fabric types in the data set, convolutional neural networks (VGG16, EfficientNetB0, ResNet50, DenseNet121, and MobileNet) were used to classify the images. The same hyperparameters were applied to each model. As a result, the performance of the data set utilised in the study was measured in the way of accuracy, precision, recall and F1-score values. In addition, the models' accuracy and loss graphs are shown in graphical form.

In this thesis study, it has been observed that convolutional neural network models are generally successful in fabric defect classification. It has been observed that the models show different performances depending on the fabric type. Additionally, it has been observed that the hyperparameters of the models are directly proportional to the performance of the models.

In future studies, different model-specific hyperparameters can be selected to obtain higher performance results on fabric types. Choosing a model-specific hyperparameter applied to the fabric type will increase performance results. In addition, by increasing the number of 3 different fabric types used in this study, the performance of models can be measured on a total of 19 different fabric types in the ZJU-Leaper data set. Apart from the models used within the scope of this thesis, we can also examine different models to see performance differences.

REFERENCES

Kırat, S. S., & Aydın, İ. (2023). Açıklanabilir Yapay Zekâ Tabanlı Denetimsiz Öğrenme ile Ray Kusur Tespiti. Demiryolu Mühendisliği, (18), 1-13.

Ekici, B. B., & Ustaoğlu, S. T. (2023). Deep Learning for Physical Damage Detection in Buildings: A Comparison of Transfer Learning Methods. Turkish Journal of Science and Technology, 18(2), 291-299.

Şahin, N., Alparslan, N., İlçin, M., & Hanbay, D. (2023). Evrişimsel Sinir Ağı Mimarileri ve Öğrenim Aktarma ile Bitki Zararlısı Çekirge Türlerinin Sınıflandırması. Fırat Üniversitesi Mühendislik Bilimleri Dergisi, 35(1), 321-331.

Yıldırım, Ö., & Özbay, F. A. (2022). Fundus Görüntülerinden Derin Öğrenme Teknikleri ile Glokom Hastalığının Tespiti. Avrupa Bilim ve Teknoloji Dergisi, (44), 1-6.

Demirci, D., Saraçbaşı, E., Emrah, E., Uzun, İ., Genç, Y., & Özkan, K. (2022). Domates Hastalığı Tahmini İçin Gerçek Zamanlı Uygulama. Eskişehir Osmangazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi, 30(1), 90-95.

Örenç, S., Acar, E., & Özerdem, M. S. (2022). Utilizing the ensemble of deep learning approaches to identify monkeypox disease. Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi, 13(4), 685-691.

Eliaçık, B., Dal, A., & Isık, A. H. (2023). Beyin Tümörü MR Görüntülerinin Evrişimsel Sinir Ağları ile Sınıflandırılması. Uluborlu Mesleki Bilimler Dergisi, 6(1), 71-91.

Eryılmaz, F., & Karacan, H. (2021). Akciğer X-Ray Görüntülerinden COVID-19 Tespitinde Hafif ve Geleneksel Evrişimsel Sinir Ağ Mimarilerinin Karşılaştırılması. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 9(6), 26-39.

Bozkurt, F. (2021). Classification of blood cells from blood cell images using dense convolutional network. Journal of Science, Technology and Engineering Research, 2(2), 81-88.

Mpinyuri, P. T., & Tarambiwa, E. (2022, November). Vehicle Damage model classification for Zimbabwe Insurance Sector using MobileNetV2 and DenseNet121. In 2022 1st Zimbabwe Conference of Information and Communication Technologies (ZCICT) (pp. 1-5). IEEE.

Gill, K. S., Anand, V., & Gupta, R. (2023, July). Arrhythmia Classification Using ECG Image Dataset Using Machine Learning Approach on DenseNet121 Model. In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-4). IEEE.

Şafak, E., & Barışçı, N. (2022). Hafif Evrişimsel Sinir Ağları Kullanılarak Sahte Yüz Görüntülerinin Tespiti. El-Cezeri, 9(4), 1282-1289.

Goutham, V., Sameerunnisa, A., Babu, S., & Prakash, T. B. (2022, April). Brain Tumor Classification using EfficientNet-B0 Model. In 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 2503-2509). IEEE.

Arun, Y., & Viknesh, G. S. (2022, January). Leaf Classification for Plant Recognition Using EfficientNet Architecture. In 2022 IEEE Fourth International Conference on Advances in Electronics, Computers and Communications (ICAECC) (pp. 1-5). IEEE.

Bilgin, M. M., Özdem, K., & Akaçyol, M. A. (2022). Derin Öğrenme ile Kuş Türü Sınıflandırma: Karşılaştırmalı Bir Çalışma. Journal of Polytechnic, 25(3).

Şafak, E., & Barışçı, N. (2023). Derin öğrenme kullanılarak mobil cihazlar için gerçek zamanlı yangın ve duman tespiti. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 38(4), 2179-2190.

İşlek, B., & Erol, H. (2023). Yüz İfadelerini Sınıflandırmada CNN Modellerinde Kullanılan Optimizasyon Yöntemlerinin Karşılaştırılması. Bilgisayar Bilimleri ve Teknolojileri Dergisi, 4(1), 1-7.

Utomo, S., Affandi, M. N. W., & Shidik, G. F. (2023, September). Asphalt Road Quality Classification Based on CNN Architecture VGG16. In 2023 International Seminar on Application for Technology of Information and Communication (iSemantic) (pp. 101-106). IEEE.

Zhang, J. W., Lee, K. C., & Reddy, G. A. K. (2022, October). Rubber Gasket Defect Classification by VGG16 model. In 2022 IEEE 4th Eurasia Conference on IOT, Communication and Engineering (ECICE) (pp. 494-499). IEEE.

Wang, H. (2020, April). Garbage recognition and classification system based on convolutional neural network VGG16. In 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE) (pp. 252-255). IEEE.

Şengöz, N. Utilizing Deep Learning and Data Augmentation for Early Detection of Eye Diseases In Pets. International Journal of Engineering and Innovative Research, 5(2), 112-122.

Cengiz, E., Yilmaz, C., & Kahraman, H. (2021). Classification of human and vehicles with the deep learning based on transfer learning method. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 9(3), 215-225.

Giray, A. O., & Doğan, H. (2023). Lane detection for autonomous driving in a video game environment. Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi, 6(2), 209-222.

Varjovi, M. H., Talu, M. F., & Hanbay, K. Fabric Defect Detection Using Customized Deep Convolutional Neural Network for Circular Knitting Fabrics. Türk Doğa ve Fen Dergisi, 11(3), 160-165.

Şeker, A., & Yüksek, A. G. (2017). Stacked autoencoder method for fabric defect detection. Cumhuriyet Üniversitesi Fen Edebiyat Fakültesi Fen Bilimleri Dergisi, 38(2), 342-354.

Peng, Z., Gong, X., Lu, Z., Xu, X., Wei, B., & Prasad, M. (2021, November). A novel fabric defect detection network based on attention mechanism and multi-task fusion. In 2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC) (pp. 484-488). IEEE.

Durmuşoğlu, A., & Kahraman, Y. (2021, October). Detection of fabric defects using convolutional networks. In 2021 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-5). IEEE.

Şeker, A. (2018, September). Evaluation of fabric defect detection based on transfer learning with pre-trained AlexNet. In 2018 International Conference on Artificial Intelligence and Data Processing (IDAP) (pp. 1-4). IEEE.

Çelik, H. İ., Dülger, L. C., Öztaş, B., Kertmen, M., & Gültekin, E. (2022). A Novel Industrial Application of CNN Approach: Real Time Fabric Inspection and Defect Classification on Circular Knitting Machine. Textile and Apparel, 32(4), 344-352.

Hamdi, A. A., Sayed, M. S., Fouad, M. M., & Hadhoud, M. M. (2018, February). Unsupervised patterned fabric defect detection using texture filtering and K-means clustering. In 2018 International Conference on Innovative Trends in Computer Engineering (ITCE) (pp. 130-144). IEEE.

Liu, K. H., Chen, S. J., & Liu, T. J. (2022, July). Unsupervised UNet for Fabric Defect Detection. In 2022 IEEE International Conference on Consumer Electronics-Taiwan (pp. 205-206). IEEE.

Zhang, H. W., Zhang, L. J., Li, P. F., & Gu, D. (2018, May). Yarn-dyed fabric defect detection with YOLOV2 based on deep convolution neural networks. In 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS) (pp. 170-174). IEEE.

Yin, Y. Y., & Li, L. Q. (2022, November). A lightweight algorithm for woven fabric defect detection. In 2022 China Automation Congress (CAC) (pp. 1025-1029). IEEE.

Guan, M., Zhong, Z., Rui, Y., Zheng, H., & Wu, X. (2019, September). Defect detection and classification for plain woven fabric based on deep learning. In 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD) (pp. 297-302). IEEE.

Erdoğan, Y. (2022). Kumaş Hatalarının Derin Öğrenme İle Tespiti. [Master's Thesis]. Süleyman Demirel University.

Kahraman, Y. (2022). Deep Learning Based Fabric Defect Detection. [Doctoral Dissertation]. Gaziantep University

Çıklaçandır, G. F. (2022). Detection and Classifying Fabric Defects With Computer-Vision Algorithms. [Doctoral Dissertation]. Dokuz Eylül University.

Softwaretestinghelp. (2024, January 22). Data Mining Vs Machine Learning Vs Artificial Intelligence Vs Deep Learning. https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/

Mathworks. (2023, Jan 09). What Is a Convolutional Neural Network? https://www.mathworks.com/discovery/convolutional-neural-network.html

Geeksforgeeks. (2023, Jan 09). CNN | Introduction to Pooling Layer. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Jin, J., Dundar, A., & Culurciello, E. (2014). Flattened convolutional neural networks for feedforward acceleration. arXiv preprint arXiv:1412.5474.

Unzueta, D. (2023, December 3). Fully Connected Layer vs. Convolutional Layer: Explained. Builtin.https://builtin.com/machine-learning/fully-connected-layer

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of Machine Learning Research, 15(1), 1929-1958.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510-4520).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

Sharma, T. (2023, December 2). Detailed Explanation of Resnet CNN Model. Medium.https://medium.com/@sharma.tanish096/detailed-explanation-of-residual-network-resnet50-cnn-model-106e0ab9fa9e

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 4700-4708).

Radwan, N. (2019). Leveraging sparse and dense features for reliable state estimation in urban environments (Doctoral Dissertation, University of Freiburg, Freiburg im Breisgau, Germany).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378.

Seyyarer, E., Ayata, F., Uçkan, T., & Karci, A. (2020). Derin öğrenmede kullanılan optimizasyon algoritmalarının uygulanması ve kıyaslanması. Computer Science, 5(2), 90-98.

Zhang, C., Feng, S., Wang, X., & Wang, Y. (2020). ZJU-Leaper: A benchmark dataset for fabric defect detection and a comparative study. IEEE Transactions on Artificial Intelligence, 1(3), 219-232.

Erikçi, E. (2023). Derin Öğrenme İle Kolorektal Kanserde Mikrosatellite İnstabilite'nin Tahmini. Zonguldak Bülent Ecevit University.