**DOKUZ EYLÜL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**Melanoma Skin Cancer Detection Using Deep Learning**

**by**

**S.Ayberk KILIÇASLAN**

**Hazar ÖZYAĞCI**

**Arif MERTASLAN**

**Advisor**

**Assoc. Prof. Dr. Zerrin IŞIK**

**November, 2021**

**İZMİR**

**Melanoma Skin Cancer Detection Using Deep Learning**

**A Thesis Submitted to the**

**Dokuz Eylül University, Department of Computer Engineering**

**In Partial Fulfillment of the Requirements for the Degree of  B.Sc.**

**by**

**S.Ayberk KILIÇASLAN**

**Hazar ÖZYAĞCI**

**Arif MERTASLAN**

**Advisor**

**Assoc. Prof. Dr. Zerrin IŞIK**

**June, 2022**

**İZMİR**

**SENIOR PROJECT EXAMINATION RESULT FORM**

We have read the thesis entitled **“Melanoma Skin Cancer Detection Using Deep Learning”** completed by **Hazar ÖZYAĞCI, S. Ayberk KILIÇASLAN** and **Arif MERTASLAN** under advisor of **Assoc. Prof. Dr. Zerrin IŞIK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of B.Sc.

Advisor

Committee Member           Committee Member

Prof. Dr. Yalçın ÇEBİ

Chair

Department of Computer Engineering

**ACKNOWLEDGEMENTS**

We would like to thank our consultant Assoc. Prof. Dr. Zerrin IŞIK for his continuous assistance at all stages of our project. We are deeply grateful to them for her guidance and support, her patience and motivation during difficult times. Her knowledgeable supervision and valuable comments on the thesis are of great importance to us. This project would not have been possible without her.

Hazar ÖZYAĞCI

S. Ayberk KILIÇASLAN

Arif MERTASLAN

**Melanoma Skin Cancer Detection Using Deep Learning**

**ABSTRACT**

Melanoma is a cancer caused by pigment-containing cells called melanocytes. It is one of the most dangerous cancers. Melanoma cancer causes nearly 55,000 deaths per year, accounting for 0.7 percent of all cancer deaths. On the other hand, recently, medical diagnostic systems with artificial intelligence support have become quite widespread. The aim of this project is to develop a mobile phone application that will help in the early diagnosis of melanoma skin cancer with the help of deep neural network models.

In the project, 4000 melanoma cancer and 4000 benign labeled skin lesion images selected from the dataset of the ISIC 2017 competition were used. After the images were passed through preprocessing, they were used in the training of deep neural networks. The results of different deep learning models have been compared with each other. Comparisons have been made both in terms of accuracy rates and their applicability to mobile phones. As a result of the comparisons made, the most suitable model has been determined and it has been brought to the format to be used in mobile applications.

**Derin Öğrenme Kullanarak Melanoma Cilt Kanseri Tespiti**

**özet**

Melanoma cilt kanseri, pigment içeren hücrelerin, melanositlerin neden olduğu en tehlikeli kanser türlerinden biridir. Her yıl melanoma cilt kanseri nedeniyle yaklaşık 55.000 ölüm meydana gelir ve bu da tüm kanser ölümlerinin %0.7 oranını kapsamaktadır. Diğer yandan, son zamanlarda yapay zeka destekli tıbbi teşhis sistemleri oldukça yaygınlaşmıştır. Bu projenin amacı, derin sinir ağı modelleri yardımıyla melanoma cilt kanserinin erken teşhisine yardımcı olacak bir cep telefonu uygulaması geliştirmektir.

Projede, ISIC 2017 yarışmasının veri kümesinden seçilen 4000 adet melanoma kanseri ve 4000 adet iyi huylu olarak etiketlenmiş cilt lezyon görüntüleri kullanılmıştır. Görüntüler ön işleme süreçlerinden geçirildikten sonra derin sinir ağlarının eğitiminde kullanılmıştır. Farklı derin öğrenme modellerinin sonuçları birbirleriyle kıyaslanmıştır. Hem doğruluk oranları hem de cep telefonlarına uygulanabilirlikleri açısından karşılaştırmalar yapılmıştır. Yapılan karşılaştırmalar sonucunda en uygun model tespit edilmiştir ve mobil uygulamalarda kullanılacak formata getirilmiştir.

**CONTENTS**

**Page**

**PROJECT EXAMINATION RESULT FORM** 3

**ACKNOWLEDGEMENTS** 4

**ABSTRACT**  5

**ÖZET** 6

**CHAPTER ONE**

**INTRODUCTION 13**

   1.1. Background Information 13

   1.2. Problem Definition 14

   1.3. Motivation / Related Works 14

   1.4. Goal / Contribution 15

   1.5. Project Scope 15

   1.6. Methodology / Tools / Libraries 16

**CHAPTER TWO**

**LITERATURE REVIEW 17**

   2.1. Related Works 17

2.1.1.Convolutional neural networks for classifying melanoma images 17

2.1.2. Melanoma skin cancer detection using deep learning 18

          and classical machine learning techniques: A hybrid approach

2.1.3. A Deep Learning-Based Transfer Learning Framework 19

           for the Early Detection and Classification of Dermoscopic

           Images of Melanoma

2.1.2. Vision-Based Classification of Skin Cancer 20

          Using Deep Learning

    2.2. Algorithms             22

2.2.1. Support Vector Machines             22

2.2.2. Logistic Regression             22

2.2.3. Convolutional Neural Networks             22

2.2.3.1. Generative Adversarial Networks           23

2.2.3.2. ResNet           24

2.2.3.3. VGG16           24

   2.3. Tools and Libraries                     25

2.3.1. TensorFlow         25

2.3.2. TensorBoard         25

2.3.3. Keras         25

2.3.4. Scikit-Learn         25

2.3.5. OpenCV         26

**CHAPTER THREE**

**REQUIREMENTS / REQUIREMENT ENGINEERING         27**

    3.1. Functional Requirements         27

3.1.1. Training of Models 27

3.1.2. Uploading Image or Taking Photo         27

3.1.3. Editing of Images         27

3.1.4. Lesion Classification Results         27

    3.2. Non-Functional Requirements         28

3.2.1. Performance         28

3.2.2. Scalability         28

3.2.3. Usability         28

3.2.4. Security         28

3.2.5. Reliability         28

3.2.6. Clarity         28

**CHAPTER FOUR**

**DESIGN         28**

   4.1. Architectural View         28

   4.2. UML Class Diagram 29

   4.3. UI Design         30

  4.4. Activity Diagram         31

   4.5. Use Cases         32

   4.6. Sequence Diagram                     32

   4.7. Deployment Diagram         33

**CHAPTER FIVE**

**IMPLEMENTATION         35**

   5.1. Data Preprocessing         35

5.1.1. Manuel Data Elimination         35

5.1.2. Defishing of Fisheye Images         36

5.1.3. Hair Removing         37

5.1.4. Preparation for Training         38

  5.2. Deep Convolutional Neural Networks         38

5.2.1. VGG16         38

5.2.2. MobileNet         39

5.2.3. Inception v3         40

   5.3. Mobile Application         41

5.3.1. Android Studio         41

5.3.2. Kotlin and XML (Extensible Markup Language)         42

5.3.3. Tensorflow Lite 42

5.3.4. Intent         42

5.3.5. Data Binding         42

5.3.4. Kotlin Coroutines         42

**CHAPTER SIX**

**TEST / EXPERIMENTS 43**

   6.1. Common Experiments Preparations 43

   6.2. Experiment 1 (Initial Training)         43

   6.3. Experiment 2 (Parameter Optimization)         45

6.3.1. Bayesian Search for MobileNet         45

6.2.3. Bayesian Search for Inception v3         46

6.4.Experiment 3 (Tuned Model Training)         47

6.4.1. MobileNet Accuracy - Loss Plots         47

6.4.2. Inception Accuracy - Loss Plots         50

6.5. Experiment 3 (Tuned Model Training)         53

6.5.1. Test Case 1         55

6.5.2. Test Case 2         56

6.5.3. Test Case 3         56

6.5.3. Test Case 4         57

6.5.3. Test Case 5         58

**CHAPTER SEVEN**

**CONCLUSION         59**

**REFERENCES         60**

**LIST OF FIGURES**

**Page**

Figure ‎4.1 Architectural view. 29

Figure ‎4.2 Class Diagram. 30

Figure ‎4.3 UI Design. 31

Figure ‎4.4 Activity Diagram. 32

Figure ‎4.5 Use Cases. 33

Figure ‎4.6 Sequence Diagram. 34

Figure ‎4.7 Deployment Diagram. 34

Figure ‎5.1 An Eliminated Image Example. 35

Figure ‎5.2 Defisheye Operations. 36

Figure ‎5.3 Hair removing Operations. 37

Figure ‎5.4 VGG16 Architecture (Samadhi P. K et al, 2020). 38

Figure 5.5MobileNet Architecture (Medium, 2020). 40

Figure 5.6 Inception v3 Architecture (Google Cloud, 2022). 41

Figure 6.1 MobileNet Fold 2 Initial Training Accuracy and Loss Plots. 44

Figure 6.2 Inception v3 Fold 2 Initial Training Accuracy And Loss Plots. 45

Figure 6.3 MobileNet Fold 1, Tuned Model, Training Line Charts. 47

Figure 6.4 MobileNet Fold 2, Tuned Model, Training Line Charts. 48

Figure 6.5 MobileNet Fold 3, Tuned Model, Training Line Charts. 48

Figure 6.6 MobileNet Fold 4, Tuned Model, Training Line Charts. 49

Figure 6.7 MobileNet Fold 5, Tuned Model, Training Line Charts. 49

Figure 6.8 Inception v3 Fold 1, Tuned Model, Training Line Charts. 50

Figure 6.9 Inception v3 Fold 2, Tuned Model, Training Line Charts 50

Figure 6.10 Inception v3 Fold 3, Tuned Model, Training Line Charts. 51

Figure 6.11 Inception v3 Fold 4, Tuned Model, Training Line Charts. 51

Figure 6.12 Inception v3 Fold 5, Tuned Model, Training Line Charts. 52

Figure ‎6.13  True Predicted MobileNet Results. 53

Figure ‎6.14  False Predicted MobileNet Results. 54

Figure ‎6.15  “*resizeImage*” Function Unit Test. 55

Figure ‎6.16 “*createImageByteArray*” Function Unit Test. 56

Figure ‎6.17 “*onCameraCardClicked*” Function Unit Test. 57

Figure ‎6.18 “*loadModelFile*” Function Unit Test. 57

Figure ‎6.19 “*loadLabelList*” Function Unit Test. 58

**LIST OF TABLES**

Table ‎6.1 Initial Training Results. 44

Table ‎6.2 Bayesian Search Results for MobileNet. 46

Table ‎6.3 Bayesian Search Results for Inception v3. 46

Table ‎6.4 Tuned Model Training Results. 47

**CHAPTER ONE**

**INTRODUCTION**

**1.1 Background Information**

Melanoma is one of the most dangerous cancer types that is caused by pigment-containing cells, melanocytes. Annually almost 55,000 deaths happen which is 0.7% of all cancer deaths because of melanoma cancer (Daghrir et al., 2021).

According to the reports of American Cancer Society, about 106,110 (62,260 men, 43,850 women) new melanomas will be diagnosed in 2021. In addition, 7,180 people are expected to die of melanoma. The development of melanoma is more common as people grow older. But it also develops in younger people, including those younger than 30 years old. It is one of the most common cancers diagnosed in young adults, particularly for women. In 2020, about 2,400 cases of melanoma were estimated to be diagnosed in people aged 15 to 29.

Tissue digital image is used for detection of different types of skin cancer such as melanoma, basal cell carcinoma, etc. One of the common techniques used for this objective is epiluminescence microscopy (ELM), also known as dermoscopy (Tsong-Long Hwang, 2007). ABCD (asymmetry, border structure, color, diameter) rule is the crucial method to diagnose melanoma skin cancer. In general, lesions which include melanoma structures asymmetrically and have uneven borders. These lesions also have multicolored structure and their diameters larger than one-quarter inch.

Due to late diagnosis of melanoma cases, patient would not get proper cancer treatment in time. Therefore, easy and effective diagnosis tools, which are applicable in home environment, might help to expand the lifetime of patients.

**1.2 Problem Definition**

Usually, melanoma is a type of skin cancer that is detected after reaching the critical stage. It is a disease that is difficult to treat because it has a structure that spreads throughout the body. That is why the detection of melanoma should occur as early as possible. The melanoma in the dermoscopic images may not be noticed by the dermatologist. In this case, developing computer vision systems has become inevitable to use disease detection in this field. Day by day, new studies are being conducted for new automated cancer detection systems. In this project, image processing operations and deep learning techniques will be proposed to contribute to more effective diagnosis of skin cancer.

**1.3 Motivation / Related Works**

Main goal of the project is to implement a dermoscopic image classification approach by applying deep learning models and image processing methods. As a result of the research conducted, it was found that there are other studies conducted on this topic. Proposed method is aimed to develop a system that gives better results by taking advantage of these studies.

Russell S. Berman et al. proposed a system that heavily relies on the processing unit for removing image occlusions and the data generation unit. In the stage of populating scarce lesion classes, or equivalently creating virtual patients with predefined types of lesions, generative adversarial networks are based on.

Adekanmi A. Adegun et al. contribute a system that includes a supervised learning method which provides an end to end and pixel by pixel classification approach using deep convolutional networks combined with softmax classifier and dice loss function.

Jinen Daghirir et al. contribute an automatic detection of melanoma lesions on skin images by using the concept of deep learning. The results indicate that deep learning methods using convolutional neural networkCNN are able to detect the melanoma lesion efficiently.

**1.4 Goal / Contribution**

Recently, artificial intelligence-assisted disease detection systems have become quite widespread. The aim of this project is to help the early detection of skin cancer by performing melanoma detection using deep neural network models.

In the studies to be carried out in this direction, ISIC Challenges (2016, 2017, 2018, 2020) and PH2 data that have been passed through the necessary preprocessing methods will be used during the training of deep learning models that we will use to detect melanoma.

If successful models are used in the health sector, the prevention of possible errors that may be made in cancer diagnosis will be largely provided. The diagnosis tools applicable at home would expand the lifetime of patients by taking proper cancer treatments on time.

**1.5 Project Scope**

Although disease detection with deep learning models is a method that has started to be applied frequently and has yielded successful results, it is not possible to access a model with one hundred percent accuracy and error-free. Diagnosis of melanoma, which has an important place in the treatment of skin cancers at the beginning, cannot be done with a hundred percent accuracy. Automated detection system that works with the highest accuracy will be created by considering the risk level.

The data that will be used for the detection of diseases will go through the stages respectively collecting, pre-processing and the cleaning from the noise, to achieve higher accuracy. The classification process required for a disease detection can be completed by different methods; making a classification based on segmented images or estimating the disease class based on the original image.

According to the proposed method, an automated melanoma detection system will be implemented. Thanks to this automation, it is expected that a patient will be classified whether the lesion is melanoma or benign according to the original lesion image of the patient.

**1.6 Methodology / Tools / Libraries**

In this project, it will be mentioned about the research conducted on the accuracy of deep learning models, which have become very widely used in disease detection, in the detection of melanoma cancer. Finally, a new solution will be experimented by using the methods already presented with the system obtained.

In order to develop deep learning models in a computer environment, the necessary programming languages and libraries were investigated. As a result of the research conducted, the Python programming language meets the requirements with the tools and libraries contained in it. PyTorch and Tensorflow effectively use libraries such as Keras, NumPy, Pandas, Matplotlib, OpenCV while developing a system. These tools will be used when implementing the proposed system.

In the training phase of models to be developed, a sufficient number of labeled images are needed. Different databases have been used to collect these labeled samples: ISIC Challenges, PH2. The necessary pre-processing operations will be performed on the data obtained from these databases. As initial operations, image pre-processing methods that will be used to reduce noise on images. In addition, data augmentation will be applied to balance the number of diseased (melanoma) dermoscopic images with disease-free images.

**CHAPTER TWO**

**LITERATURE REVIEW**

* 1. **Related Works**

As in this project, some studies using different machine learning methods for skin lesion classification were described.

***2.1.1. Convolutional neural networks for classifying melanoma images***

The study was conducted to reduce cancer deaths by facilitating the early detection of skin cancer (Sagar and Jacob 2020). In order to achieve the cancer binary classification process such as benign and malignant, the transfer learning method was applied. Various ready-made deep convolutional neural networks have been used and tested throughout the transfer learning. ISIC data was used throughout the training and testing processes, it covers 3000 images for training and 600 images for validation operations. An equal number of images with a resolution of 224x224 was used in both classes, they were also distributed in the right proportion to the training and validation sets. The images used in the training process and validation processes have been passed through augmentation operations: shearing, zooming, flipping, brightness changing. Inception v3, Inception ResNet v2 and ResNet50, MobileNet and DenseNet169 models, which are one of the commonly used deep learning models, were trained from the beginning using the training kit. The top layers have been replaced with average pooling layers and 2 new fully connected layers have been added to the end of the models. The first of these layers contains 64 dense, while the last layer contains 2 dense to do binary classification, and softmax was used as an activation function. In the models, overfitting was prevented by using drop-out and batch normalization methods by 50%. At the training stage, binary cross-entropy was used as a loss function.

When all the parameters at the training stage were compared in themselves, it was seen that the Adam optimizer contributed more to the use of SGD, and the choice of 0.0001 as the learning rate gave better results than 0.00001. At the same time, ResNet50, which gave the best results, had an accuracy of 0.93% and a ROC-AUC of 0.86%. Two other models that performed relatively better, DenseNet169 and Inception ResNet v2, achieved accuracy and ROC-AUC values of 0.90, 0.85, and 0.91, 0.84, respectively.

***2.1.2. Melanoma skin cancer detection using deep learning and classical machine learning techniques: A hybrid approach***

The study is based on 3 different methods (Daghrir et al., 2021). These methods are one CNN and two classical machine learning models. A training stage was conducted according to the characteristics of the visual data such as border, texture, and color. After separate training operations, these three different models were combined with the majority voting method to get better results.

Before the training, several pre-processing stages were applied to the data. The first of these has been the visual erasure of parts such as hair on the skin. Detecting hairlines from the dermoscopic images is done based on the 2-D derivatives of Gaussian(DOG) of the blue component of the images. The DOG function very successfully detects parts of the hair on the visual from 4 different angles. Immediately after the DOG function, the hair and hair parts were visually erased using the Otsu method, which is a threshold method, and mask images were created depending on them. At the stage of segmentation of the lesion, morphological snakes were used, which are faster and more stable. There are two morphological snakes methods: Morphological Geodesic Active Contours (MorphGAC), Morphological Active Contours without Edges (MorphACWE). In addition, to classify lesions as malignant or benign, the CNN model was used, as well as the k-nearest neighbor (KNN) classifier and support vector machine (SVM), which are classic machine learning models.

Recently, the use of CNN models instead of classical machine learning has become more widespread and has begun to give more successful results. For this reason, the CNN model was first trained with 512 images, but the CNN models need larger data sets. In accordance with this need, augmentation has been applied to the data. After the necessary augmentation was performed, 124x124 size data was provided as input to the CNN model. In order to solve the problem of overfitting that may arise during training. Other methods, such as dropout and regularization were used since the possibility of increasing the number of data is not a straightforward process. The CNN architecture is composed of 9 layers. Three convolutional layers were used (filter size is 3x3) with ReLU activation function, processed by three spatial max-pooling layers which down-samples the feature maps.

ISIC archives, which are public dataset, were used during the experiments. There are more than 20000 melanoma images in this archive. However, only 640 images were used in this study, of which 512 are in the training and the rest are in the test section. The CNN model used during the training was trained as 10 epochs. As a result of the trainings, the CNN model performs much better in addition to the classical machine learning models. KNN, SVM, CNN, Majority voting achieved accuracy values of 0.57, 0.72, 0.85, 0.88 respectively. So, integration of decision of individual models improved the classification accuracy compared to the single-model ones.

* + 1. ***A Deep Learning-Based Transfer Learning Framework for the Early Detection and Classification of Dermoscopic Images of Melanoma***

In this study, an effective and highly accurate model, called CAD-MD (automated computer-aided diagnosis system for melanoma detection), which detects benign and melanoma stages of a lesion is presented (Singh et al., 2021). As part of the project, the size of images is reduced to decrease complexity, a system called digital hair removal (DHR) is used to reduce noisy data, and segmentation is applied with the Watershed algorithm to better analyze the images.

The original versions PH2 and ISIC 2016 datasets and their augmented versions are used. Machine learning and deep learning models with transfer learning are being used. The PH2 image sample has dimensions of 765x572, the ISIC-16 sample has dimensions of 1022x767. The dimensions have been reduced to 120x160 for both the train and test sets. The original images were first converted into grayscale. Then, a morphological blackhat filter was used to remove hair images. Finally, the inpainting method was performed with the cv.INPAINTTELEA function of the OpenCV library to obtain a clean image. Data augmentation was performed with the application of hair removal procedures. For augmentation, the rotation, vertical-horizontal flips, horizontal-vertical shear methods were used. The total number of images of the PH2 is increased from 200 to 8640, and the number of images ISIC-16 is achieved to 8018 from 1271 after data augmentation. Using the Watershed algorithm, images were segmented by finding pixel boundaries with gradient magnitude of the input image and using morphological methods such as dilation, erosion, respectively.

Machine learning models such as Random Forest, SVM, KNN, Logistic Regression, and deep learning models such as LeNet-5, VGG16, VGG19, Inception V3, Xception were used. In the training conducted with augmented versions of the PH2 data, the highest accuracy value was reached to 99.1 in the VGG16 model. In the training conducted with augmented ISIC-16 data, the highest accuracy value was reached to 82.5 in the LeNet model.

* + 1. ***Vision-Based Classification of Skin Cancer Using Deep Learning***

This study, which is one of the projects conducted for the early detection of skin cancer, allows the classification of skin lesion photos taken with a standard camera amd applies both deep learning and machine learning algorithms. This study aims to develop a low-budget tool that allows doctors and patients to quickly identify benign and malignant lesions by simply photographing them. The visual classification was made based on the ABCD methods used by dermatologists to determine whether the lesions are cancerous, which allows them to be classified according to the anomaly in the image of the lesion.

The ISIC data set was used as training and test data. The data set includes 249 malignant melanomas and 1031 benign images. The data has been pre-processed due to the following problems that they detected in the original images and would create noise.

* A vignetting effect that causes the brightness to decrease towards the edges of the frame in some images.
* The brightness is caused by the presence of a band-aid in some images.
* The effect of hairs of different densities.
* The presence of photos of different sizes.
* Color differences are caused by the environment in which the photo was taken and the skin color of the owner of the photo.
* Differences in aspect ratio and resolution are caused by shooting images with different devices.
* Differences between the standard camera and dermoscopic images.

In order to prevent these problems that will reduce the success rate of the models, data was pre-processed on all images by operations such as cropping, contrasting, filtering, defining boundaries, and resizing according to the size determined for the model. First, the Gaussian kernel blur filter has been applied to each image to soften the edges of the image and reduce the effect of the hairs in the image. Then, with an algorithm that detects whether there is a vignetting effect, images with a vignetting effect are determined, and the edges of these determined images are cleaned from the image by a circular cropping process. As next stage, the contrast filter, was applied to all images that did not have a vignetting effect, as well as to cropped images with a vignetting effect. Then, the contrast filter, was applied to the cropped state of images with a vignetting effect and to all other images. After the contrast of the images was increased using the functions in the OpenCV library, the images were converted to gray-scale. Again, using the Canny Edge Detector function of the OpenCV library, the circumference of the lesion was determined and the lesion was clipped. Then, using the average pixel value determined by an algorithm as a threshold, a binary mask was obtained. By comparing the mask obtained as a result of preprocessing with the original masks that came with the data, the error rate was measured to be 0.30. After preprocessing and segmentation, all images were resized to a size of 256x256x3 with the resizing function of OpenCV. In order to prevent the bias caused by the number imbalance in the 4:1 ratio of benign and melanoma images, undersampling and augmentation were performed and the total number of samples were determined as 346 for both classes.

The logistic regression model was constructed using the softmax loss function and the stochastic gradient descent optimizer with parameters of 30 epochs and alpha = 0.01. A neural network was created with different number of layers and neurons except for the last layer. The last layer is made up of 2 neurons, since a binary classification is used. The linear activation function was used in the last layer, while the ReLU activation function was used in the other layers. The stochastic gradient descent optimizer was also used as an optimization function. The L1 regularization method was also used and the model was trained with 30 epochs. The transfer learning was applied on the VGG16 model with the original ImageNet weights. All layers of the VGG16 model up to the last layer have been closed for training. The last layer was opened for training and fine-tuning was applied with the fully connected layer, which was added later and used the SGD optimizer. The Transfer Learning model was also trained with 30 epochs. In the model created by the logistic regression method, 10% training and 50% test error were calculated. 1% training and 48% test error were obtained in the neural network model. In the model trained with transfer learning approach, 8% training and 25% test error were calculated. Therefore, the best-performing classification model became the VGG16 model that was trained with transfer learning.

**2.2 Algorithms**

***2.2.1 Support Vector Machines***

Support vector machine is one of the most useful supervised learning methods generally used in classification problems. Since SVM is one of the supervised learning methods, labeled data must be used for training. Otherwise, it is not possible for the training to give a successful result.

In the SVM algorithm, each data is plotted as a point in the n-dimensional space along with the value of each property, which is the value of a particular coordinate. Then the classification is carried out by finding the hyperplane, which distinguishes quite well from the two classes.

***2.2.2 Logistic Regression***

Logistic regression is a machine learning algorithm used to classify categorical data. Using the sigmoid cost function, it gives the probability values of the inputs as output. The outputs it decodes are between 0 and 1. The incoming data is assigned to a class according to the threshold value set between 0 and 1. The working logic is that the determined cost function is minimized with the Gradient Descent optimization function and the lowest erroneous model is created. Logistic regression is often used in cases such as determining whether emails are spam or not, and determining whether a transaction is fraud in banking etc.

***2.2.3 Convolutional Neural Network***

Neural networks are a subset of machine learning, and they are the most important part of deep learning algorithms. The neural network is formed as a result of connecting multiple nodes with weights determined by each other. When moving from one node to another node, the result that occurs after processing with this decoy weight is transferred to other nodes as additional data.

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are: convolutional layer, pooling layer, fully connected (FC) layer (IBM, 2021)

***2.2.3.1 Generative Adversarial Networks***

GANs (stands for Generative Adversarial Networks) can learn the images and create new images that are almost no different from the real ones ([Goodfellow](https://arxiv.org/search/stat?searchtype=author&query=Goodfellow%2C+I+J) et al., 2014). There are two main structures that make up GANS: generator, discriminator.

The generator generates an image, and the discriminator determines whether the image generated by the generator is real or fake. There is no pre-trained structure, both are trained together from scratch, and during training, these two are connected with each other. The discriminator has real images, the generator sends the images it generates to the discriminator so that the discriminator compares the image generated by the generator with the image it has and returns an answer to the generator. According to this answer, the generator updates itself. As long as the cycle continues in this way, the generator now produces images that are more similar to the discriminator's images.

The most commonly used examples of GAN are for producing images: the human face, animals, nature photos, maps, vehicle designs, clothes, pokemon, etc. In a previously completed study on the classification of skin cancer which is the subject of this project, the GAN model was used to perform data augmentation (Berman et al., 2019). New images have been obtained that are very similar to the dermoscopic images that will be used for training. A new method called coupled DCGAN is proposed in this study.

***2.2.3.2 ResNet***

The winner of the ILSVRC ImageNet competition held in 2015 was Microsoft ResNet, which had an error rate of 3.6%. While people classify the image with an average error rate of 5%-10%, ResNet with an error rate of 3.6% has shown that it performs better visual recognition than humans. ResNet contains more convolutional layers than previous models, naturally, a deeper model appears than others (Russakovsky, 2015).

When models start to contain more hidden layers, that is, when they start to deepen too much, computing becomes much more difficult and complex. There are different versions of ResNets, the most commonly used is ResNet50. ResNet50 architecture has a deep structure with residual blocks, a technique called skip connection is used in this network. Skip connections bypass several layers and connect directly to the output. In this way, the problem of exploding/vanishing gradient is prevented. Exploding gradient, on the other hand, is the opposite of vanishing and is an overgrowth of the gradient. Given the formula H(x) = F(x) + x, going deeper into the network, it is very difficult to learn H(x) since there are a lot of layers so “skipping connection” is used here. F(x) as the final output is directly in the form of input X, therefore, F(x) is called “residual”.

***2.2.3.3 VGG16***

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford. The model won the ILSVRC 2014 challenge with a 92.5% success rate. The model, trained with ImageNet data, in which there are more than 14 million images, has managed to classify 1000 classes with fairly high accuracy. The training time of the VGG16 model is higher than that of other models. The model with an input size of 224x224 contains 13 convolutional layers with 3x3 filters and 3 fully connected layers that follow the convolutional layers. The last layer consists of 1000 channels for classification with 1000 images and uses the softmax activation function, while all other layers are created with the ReLU activation function. When using transfer learning, the number of channels and the activation function of the last layer are optimized according to the categories in the visual classification to be performed.

**2.3 Tools and Libraries**

***2.3.1 TensorFlow***

Tensorflow is an open-source library that contains the basics of mathematical calculations and deep learning models. It offers GPU/CPU assisted training during the creation of deep learning models and training of these models. When using the transfer-learning method, it is also a nice advantage that it hosts pre-trained models instead of manually implementing them.

***2.3.2 TensorBoard***

Tensorboard is a visualization library that allows you to study and visualize the statistics obtained as a result of training machine learning models. It is useful to analyze which parameters the trained model is trained with, how there is a relationship between train-loss values, and many other parameters such as these.

***2.3.3. Keras***

Keras is a library for creating deep learning models. It is a neural network-based API that can work with multiple libraries, such as Tensorflow.

***2.3.4. Scikit-learn***

Scikit-learn is an open source library that contains machine learning functions and provides convenience in the field of machine learning. Data processing and training of machine learning models are the main areas of use.

Some of the most important features of Scikit-Learn:

* Datasets
* Data Split
* Machine Learning models
* Cross Validation
* Feature Selection
* Confusion Matrix
* Classification Report
* Data Pre-processing

***2.3.5. OpenCV***

OpenCV is an open-source computer vision and machine learning software library. There are a lot of algorithms for image processing and machine learning in the OpenCV library. With these algorithms, operations such as face recognition, distinguishing objects, detecting human movements, object classification can be easily performed.

**CHAPTER THREE**

**REQUIREMENTS / REQUIREMENT ENGINEERING**

**3.1. Functional Requirements**

***3.1.1. Training of Models***

The resulting skin images and their labels indicating whether they have cancer or not will be transferred to the model.

The deep learning model with the highest accuracy rate among the models trained will be used in the application.

***3.1.2. Uploading Image or Taking Photo***

From the user, an image found on his device or an image taken by the camera located on his device is transferred to the application.

***3.1.3. Editing of Images***

The user is required to resize the image to be transferred to the model. The image will be submitted to the necessary pre-processing procedure after the user has made their arrangements.

***3.1.4. Lesion Classification Result***

The image inputted into the test process is displayed with probability values, indicating which lesion class (Nevus or Melanoma) it belongs to.

**3.2. Non-Functional Requirements**

***3.2.1. Performance***

Since the trained model will operate on a mobile device, it must be compatible with the hardware of that device so that the working time is not excessive and the device does not crash.

***3.2.2. Scalability***

The application works only on devices with the operating system Android 4 and above.

***3.2.3. Usability***

It will be a system that can respond rapidly as a result of a simple image upload, thanks to an easy to use and simple user interface.

***3.2.4. Security***

Despite the fact that the software is available on personal mobile devices, no personal information will be stored or shared in the application.

***3.2.5. Reliability***

It is critical that the model deployed in the application classifies lesions with high accuracy. Except for minor variances, the image to be taken by the user from different angles should have no effect on the model prediction.

***3.2.6. Clarity***

It is critical that the model's output is easily comprehended by the user and does not have a confused meaning.

**CHAPTER FOUR**

**DESIGN**

**4.1 Architectural View**

By obtaining the ISIC 2017 Challenge dataset through the web, melanoma and nevus lesion images were utilized in classification algorithms (Figure 4.1). The images are pre-processed with the OpenCV module and made ready for training. Data augmentation techniques are used to avoid overfitting and underfitting issues in the model, with assistance from third-party technologies. The models were given processed and ready-to-use images, and TensorFlow and Keras tools were used to build and train models.

For melanoma/nevus classification, the deep learning model with the greatest accuracy rate will be chosen. The model is integrated into the application for mobile devices, and a user interface is designed. The mobile application will be able to be installed on the server, which will allow users to easily access and use it.

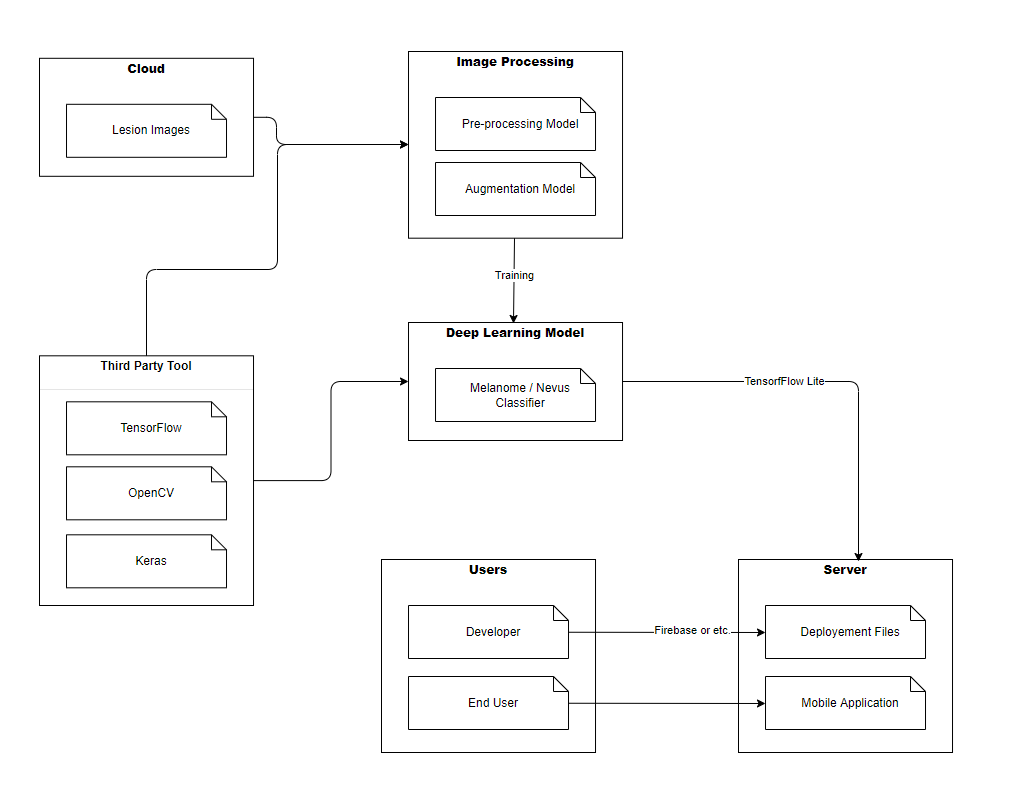


Figure ‎4.1 Architectural view.

**4.2 UML Class Diagram**

During the project's Android application development phase, the MVVM (Model-View-ViewModel) design pattern will be used. The View represents the interface of the screens, the Model represents the data class that will be displayed on the screens, and the ViewModel represents the business layer where interface elements and data are interacted with, according to the MVVM design pattern.

The application to be developed will have three screen designs. The model and viewmodel classes were not required for the first screen, as shown in the Figure 4.2, because this is the screen where the user will choose whether to upload images or take shots instantly. An interface has been established that has functions to conduct the listener event for the click operation, which is the user's only interaction on this screen. Because only the uploaded image would be modified for the second screen, the model and viewmodel classes were not required. It is proposed to develop an interface class with listeners for events to save the edit included in this page or return to the main page, as shown in the Figure 4.2. On the last screen, the uploaded image will be evaluated by a deep learning model and the results will be displayed. As shown in Figure 4.2, the ResultPageViewModel class will be used to conduct activities like taking an image and a deep learning model, as well as completing a prediction operation. The obtained results are saved in objects belonging to the ResultPageModel class when the prediction processes are performed, and they will be displayed on the screen using this class.

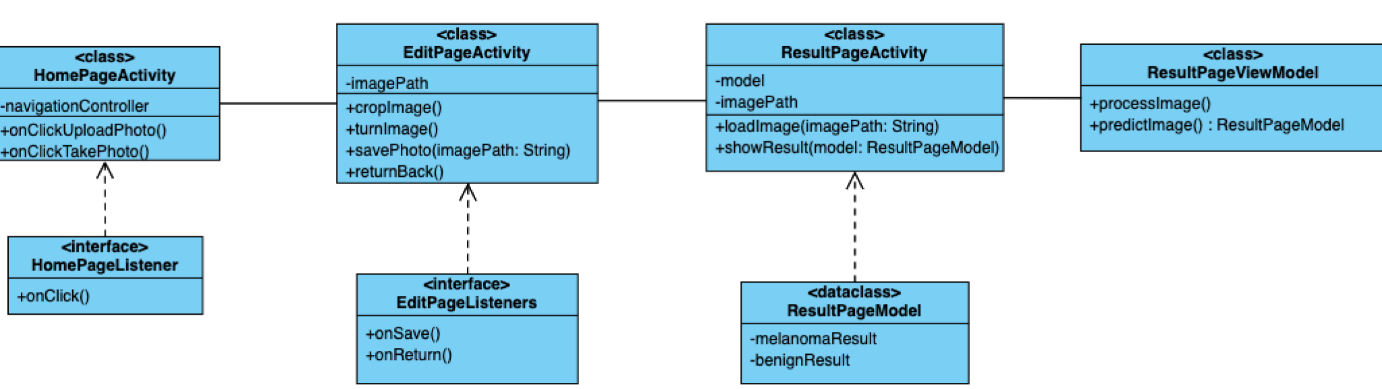
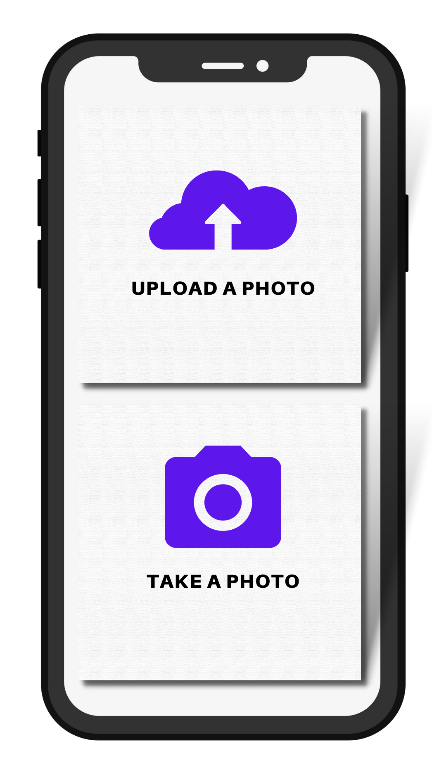


Figure ‎4.2 Class Diagram.

**4.3 UI Design**

The end-users of this application are not only doctors but also people who can access the application from their mobile device and use it in their daily lives. The application warns users by making a conclusion at a promised accuracy rate and helps them to diagnose early by providing an idea before the doctor's appointment, in any case, it is necessary to consult a specialist doctor.

Within the mobile application, user interfaces have been created to be simple to comprehend and utilize. In the application, which consists of three main screens, users are first greeted by a screen that allows them to upload images or take new photos using the device's camera (Figure 4.3). To make it suitable for the model, the image, which can be obtained in two ways, must be processed, such as cropping and rotating. There is a screen where users can edit the image for this purpose. The image is given to the model after it has been edited using the button on the same screen. The model's output is displayed on a new screen in the form of diagnosis probabilities alongside the uploaded image. To utilize a new image, the user must return to the application's homepage.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure ‎4.3 UI Design.

**4.4 Activity Diagram**

The application is simple and comprises a single-function flow. As shown in the Figure 4.4, when a user joins in to the program, he can begin the activity by uploading an image from his device or by directly connecting to the camera, taking a new photo, and uploading it. The user gets redirected to the editing page, along with the image he has submitted, after ensuring that the image is installed correctly. This page directs the user to the page where the prediction procedure will be carried out after the user has completed the cropping and rotating operations as desired. The deep learning model utilized here is being put to the test by displaying the image that the user has submitted. The results of the classification procedures are displayed to the user when the evaluation process has been verified as correct, and the activity is done.

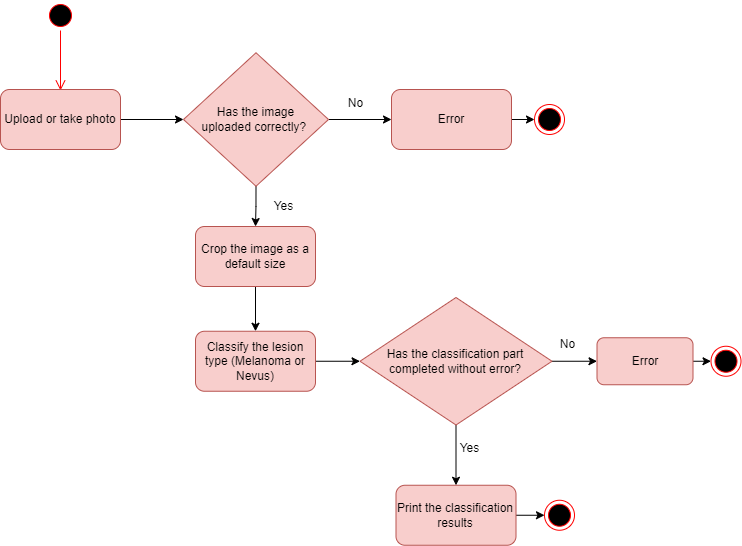


Figure ‎4.4 Activity Diagram.

**4.5 Use Cases**

The main actors in the existing system are the application developer and the end-user (Figure 4.5). The model, that the developer has implemented using skin lesion images, should be integrated into the mobile application for users to use as a diagnostic system. The developed application must be uploaded on platforms accessible to users. In order to get a result from the model in the application, users must first upload and edit the image as desired. The image that completes all the operations is given to the model by the developer to return a prediction result from the model embedded in the application. The result generated by the model is again transmitted to the user with the help of the application interface.

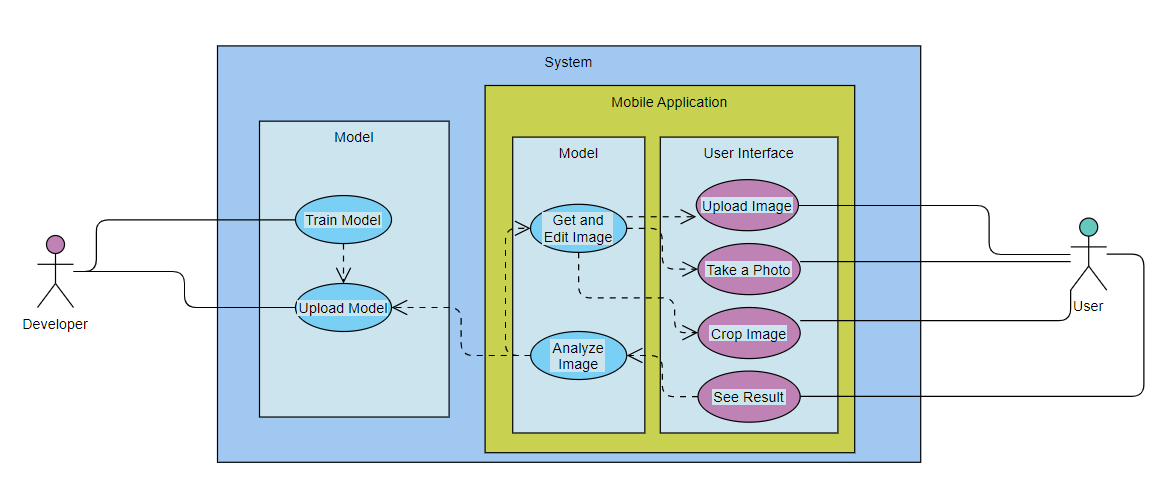


Figure ‎4.5 Use Cases.

**4.6 Sequence Diagram**

Three components of the project are shown in Figure ‎4.6. To begin, the user must upload an image to the application or take a photo with their mobile device. At the next level, the image obtained after the necessary preprocessing stages is introduced into the deep learning model. The classification result from the model is returned to the user at this point.

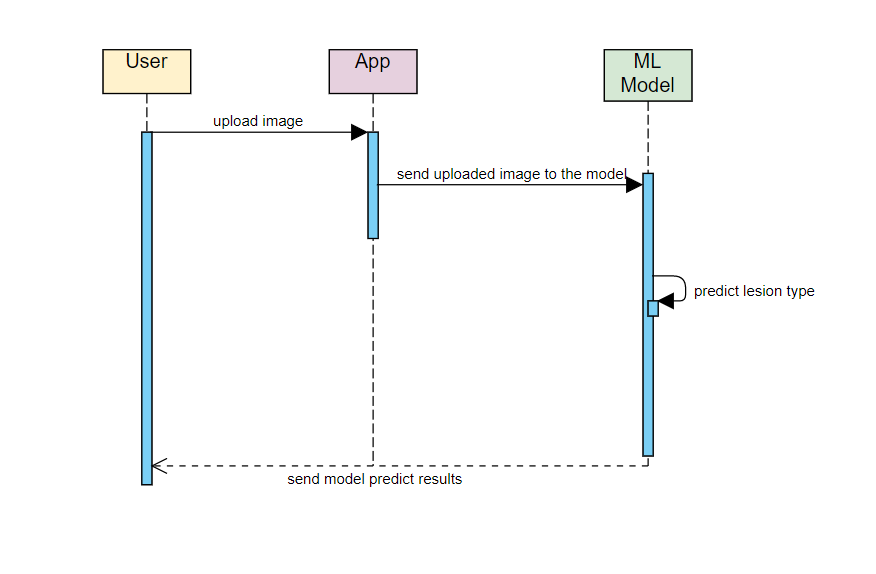


Figure ‎4.6 Sequence Diagram.

**4.7 Deployment Diagram**

The apk file for the application that will be produced for compatibility with devices above Android 4.0 will be uploaded to the firebase console as seen in Figure 4.7. It will be accessible for download and use via the Play Store following the deployment process.

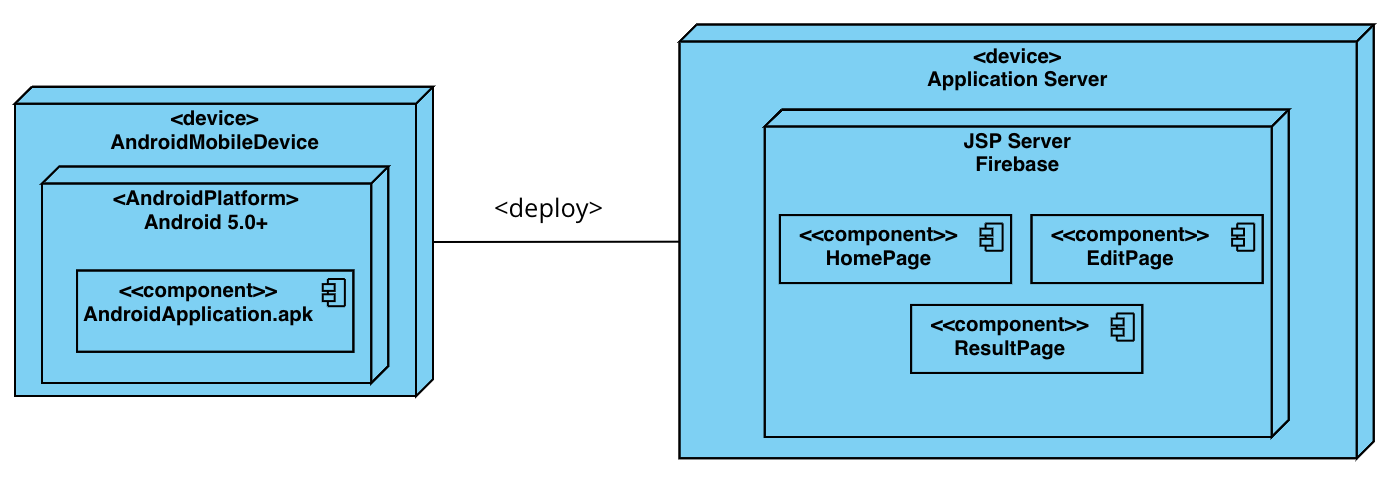


Figure ‎4.7 Deployment Diagram.

**CHAPTER FIVE**

**IMPLEMENTATION**

**5.1. Data Preprocessing**

***5.1.1. Manuel Data Elimination***

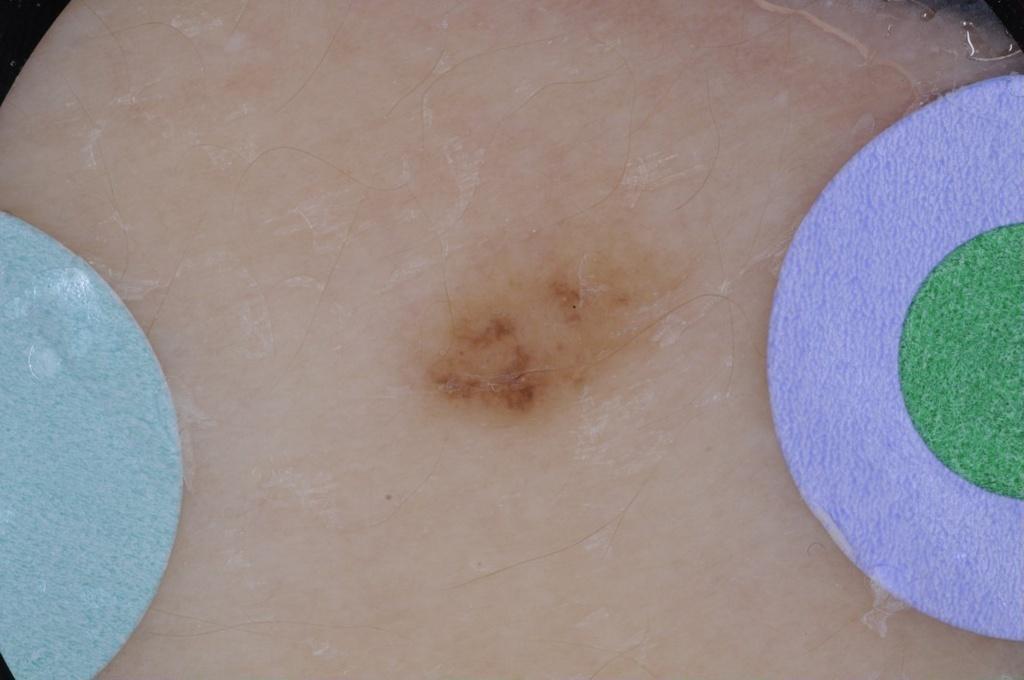
There are images from different viewpoints and with various concepts within the data set. Some of the images were not obtained from the proper or appropriate angles. Lines, fonts, and other embellishments were created with tools such as a pen in some of the images (Figure 5.1). These external influences have a noise impact on the model during training. These images were eliminated from the data set for the success results. No technological tools were used in the elimination process from the data set; a human expert manually eliminated them.

Figure ‎5.1 An eliminated image example

***5.1.2. Defishing of Fisheye Images***

One of the visual types found in the data set is the fisheye image type. The fisheye effect makes the image more rounded, adds black layers around it, and makes the image like obtained by the fisheye. The main reason for the presence of such images is that the shots taken on the patient are made by using some microscopic tools. The black pixels located around these images can create negative consequences at the stage of training the model.

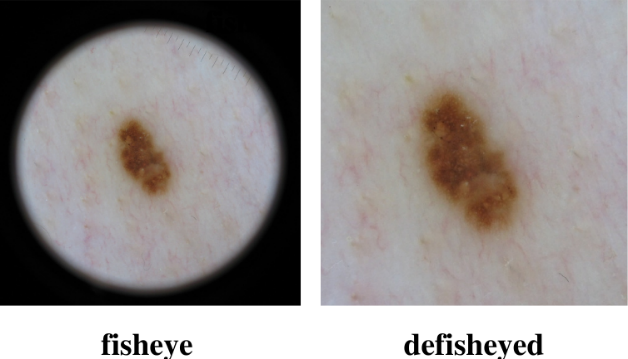
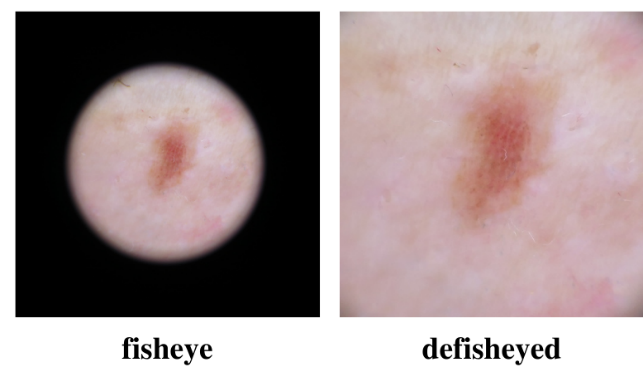
 In order to increase the success of the model, it is necessary to remove these black pixels. For this reason, the Python library, ***Defisheye***, has been used. The basic logic of the function is to expand the image horizontally and vertically from different angles and spread the image over the black pixels (Figure 5.2). Images in the form of fisheyes can be of different sizes. For this reason, images that are in the form of fisheyes are divided into three separate classes: small, medium, large fisheyes; thanks to an automation that counts the number of black pixels. In the following process, the *pov* and *fpov* values, which are the parameters of the Defisheye function, were given as 150, 110, 95 and 65, 65, 65, respectively for three different sizes of fisheye images.

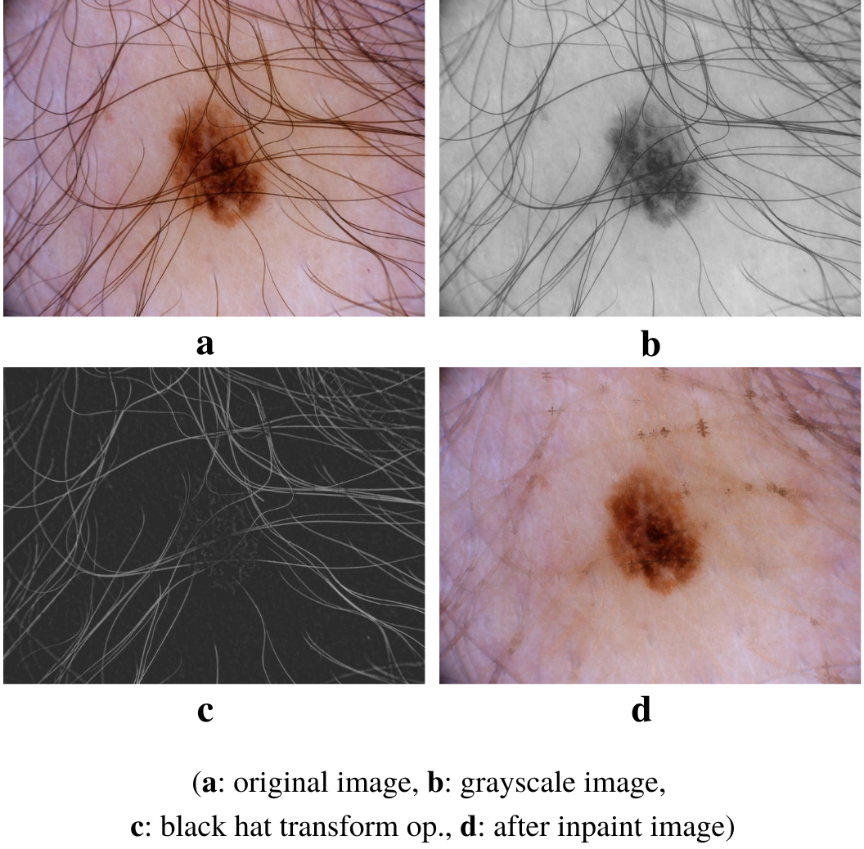
Figure ‎5.2 Defisheye operations.

***5.1.3. Hair Removing***

After manual cleaning and defishing operations performed on the raw data, the stage of cleaning the hairs from the images was completed. Since the hairs in the image can usually be the same colors as the lesion, they are very likely to be noticed as a lesion by the model. For this reason, some filtering operations have been applied using the ***OpenCV*** library for black hairs in images.

Firstly, the images contained in RGB colors are converted into black and white color images. The goal here is to be able to focus on the black hairs in the filters that will be applied next. A morphological filter, the black-hat filter, has been applied to black-and-white images with the help of a kernel with dimensions of 17x17. The difference between the closing and input images is known as the “***black-hat transform****”*. These transforms are used for a variety of image processing applications, including feature extraction, background equalization, and image enhancement (Adil H. Khan et al, 2021).

In the second stage, the enhanced black pixels were edited with the help of threshold and thus only the parts with hairs were obtained. The hair-free image was obtained by removing the threshold-applied images from the original data. In this section, the *“****inpain****t”* function included in OpenCV is used (Figure 5.3).

Figure ‎5.3 Hair removing operations

***5.1.4. Preparation for Training***

After the defisheye and hair removal operations were completed, the images were prepared for training of deep learning models. The images were resized, because the input layers of the VGG16 and MobileNet models accept a 224x224 image. As a result of the model, two different disease classes (melanoma, benign) are estimated, an equal number of samples is taken from both classes. Thus, the model was prevented to being biased towards a single class. A total of 4120 images were selected for each class and the model was trained. While training the model, a cross-validation method was used, 5 different folds with an equal number of images were created, and during each training, training was performed with 4-folds, while testing was completed with 1-fold. The training with the cross validation shows the real potential of the model and ensures the most reliable completion of the test result.

**5.2 Deep Convolutional Neural Networks**

***5.2.1* *VGG16***

At the beginning of the studies, the VGG16 deep learning model was used for the first time to determine capability of the dataset. ImageNet, a massive visual database project utilized in visual object identification software research, uses VGG16, a basic and widely used Convolutional Neural Network (CNN) Architecture. Many deep learning image classification approaches utilize VGG16, which is popular due to its ease of use. Along with its advantages, VGG16 is often used in deep learning applications.

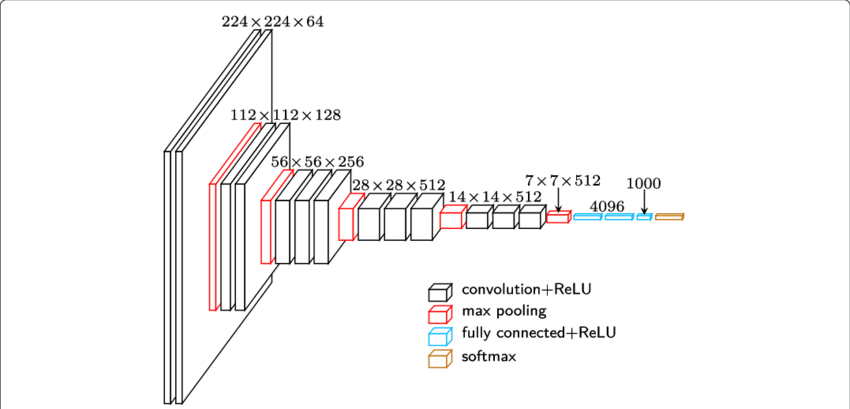


Fig. 5.4 VGG16 Architecture (Samadhi P. K et al, 2020)

The convolutions are given a fixed-size 224 by 224 RGB image during training (Figure 5.4). The only pre-processing done here is subtracting the mean RGB value derived on the training set from each pixel. In the original VGG16 architecture, following a stack of convolutional layers, there are three Fully Connected (FC) layers, the first two of which have 4096 channels each, and the third of which conducts 1000-way ILSVRC classification and so has 1000 channels, and the soft-max layer is the last layer (Medium, 2021).

In this study, since the binary classification model was developed, the FC layers in the original VGG16 model have been changed to be 128, 32, 1, respectively. After each fully connected layer, a dropout layer is found by 0.2. The ***Sigmoid function*** was preferred as the activation function. During the training process, the model containing ImageNet weights was initiated for training with every layer on these weights.

***5.2.2* *MobileNet***

After training of VGG16, a new training process was started with MobileNet which is TensorFlow's first mobile computer vision model designed to use in mobile applications. The main reason for this is that the deep learning model to be used is aimed at lower time complexity on mobile devices.

Except for the first layer, which is a full convolution, the MobileNet structure is constructed on depth wise separable convolutions (Figure 5.5). When compared to a deep neural network with normal convolutions of the same depth, it considerably reduces the number of parameters. With the exception of the last FC layer, which has no nonlinearity and feeds into a softmax layer for classification, all layers are followed by a ***batch normalization*** and ***ReLU*** for activation function. In the depth-wise convolutions as well as the first layer, down sampling is handled via strided convolution. Before the completely linked layer, a last average pooling decreases the spatial resolution to 1. MobileNet contains 28 layers when depth-wise and point-wise convolutions are counted separately ([Andrew G. Howard](https://arxiv.org/search/cs?searchtype=author&query=Howard%2C+A+G) et al, 2017).

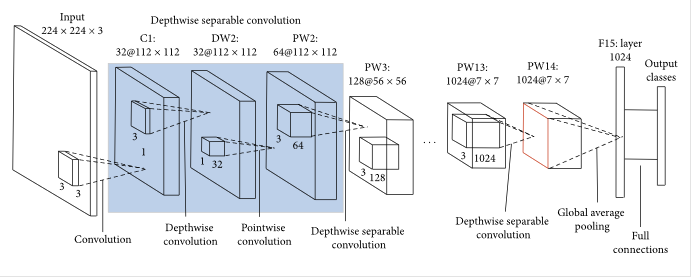


Fig. 5.5 MobileNet Architecture (Medium, 2020)

In this study, since the binary classification model was developed, the fully connected layers in the original MobileNet model have been changed to be 128, 32, 1, respectively. After each fully connected layer, the dropout layer is found by 0.2. The ***Sigmoid function*** was preferred as the activation function. During the training process, the model containing ImageNet weights was initiated for training with every layer on these weights.

***5.2.3* *Inception v3***

TensorFlow Lite publishes models that work stably in mobile applications and can be easily converted to the lite version on its website. In this context, considering both performance and speed, it was directed to the Inception v3 model, and the model began to be trained. On the ImageNet dataset, Inception v3 is an image recognition model that has been demonstrated to achieve higher than 78.1% accuracy. The model represents the result of several concepts explored over time by several scholars.

Convolutions, ***average pooling***, ***max pooling***, ***concatenations***, ***dropouts***, and fully connected layers are among the symmetric and asymmetric building components in the model (Figure 5.6). Batch normalization is done to activation inputs and is utilized extensively throughout the model. ***Softmax*** is used to calculate loss value (Christian Szegedy et al, 2015).

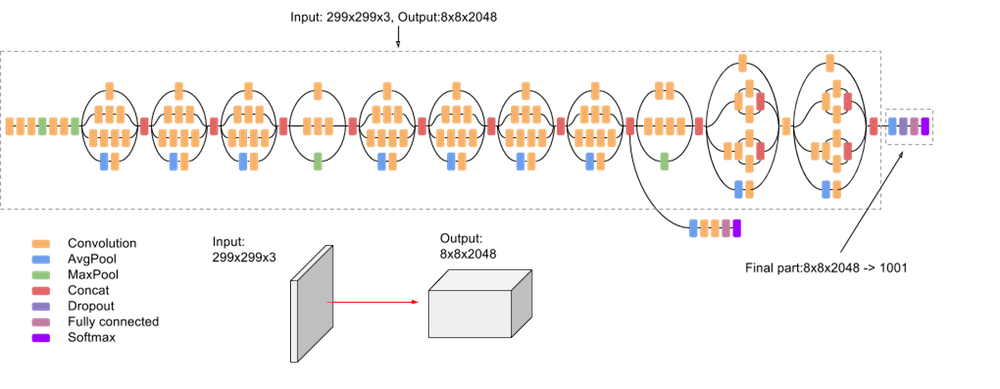


Fig. 5.6 Inception v3 Architecture (Google Cloud, 2022)

In this study, since a binary classification model was developed, the fully connected layers in the original Inception v3 model have been changed to be 128, 32, 1, respectively. After each fully connected layer, the dropout layer is found by 0.2. The ***sigmoid function*** was preferred as the activation function. During the training process, the model containing ImageNet weights was initiated for training with every layer on these weights.

**5.3 Mobile Application**

***5.3.1* *Android Studio***

Android Studio is used to create the mobile application for this project. Although Android Studio can be used for a variety of reasons, it is an official integrated development environment built by the Google and JetBrains teams for developing applications for Android-based devices. The main reasons for us to choose Android Studio are it is an official environment, and it has a strong community, so it is an environment that is constantly being improved, and that it provides faster code writing, interface control, and testing facilities.

***5.3.2* *Kotlin and XML (Extensible Markup Language)***

The Kotlin programming language was used to develop the mobile application. Kotlin is a programming language developed by a team at Jetbrains since 2010 with the aim of creating a more modern and useful language based on old languages. Despite the fact that Kotlin is a JVM-compilable language that converts written code to Java bytecode, it was chosen over Java because the developer writes more readable and short codes and the Java libraries are all available through Kotlin.

XML is the markup language used to specify the properties such as location, color, and size etc. of interface elements that appear on the screen in Android applications.

***5.3.3* *Tensorflow Lite***

Tensorflow Lite is a simpler framework for using Tensorflow, Google's open-source artificial intelligence library, on Android. The goal of utilizing Tensorflow Lite is to save models trained in Python and bring them to the mobile application so that users can make predictions based on their own data using the trained (saved) model.

***5.3.4* *Intent***

In Android applications, Intent is a library that allows you to switch from one page to another. On the current page, an intent object is created, and a page switch is done using the library's ***startActivity*** method. We'll be able to switch between pages in our application using this library, as well as offer access to the camera and gallery that the application requires.

***5.3.5* *DataBinding***

The user interface elements of Android applications are located on the XML side, while their functions are performed on the Kotlin side. Using the ***DataBinding*** library to access elements such as a button, text, etc. on the XML side, the object of each element is created while the application is running, thus providing access on the Kotlin side.

***5.3.6* *Kotlin Coroutines***

Kotlin coroutines allow developers to specify the thread in which the Android application's code should be executed, preventing interface and background jobs from executing on separate threads and obstructing each other.

**CHAPTER SIX**

**TEST / EXPERIMENTS**

**6.1. Common Experiment Preparations**

During the training of models, two different deep learning models were used: MobileNet and Inception v3. As mentioned in the previous sections, pre-trained ImageNet weights were used while training the deep learning models used. All layers of the models have been made trainable, and the last layer, the fully connected layer, has been turned into a layer with 1 node due to having a binary classifier. Also, different layers have been added between the convolutional layers and the last layer. The models were trained along the same epoch number with the same hyperparameters and the same folds set earlier. In order to find the optimal values of these parameters, the Bayesian Search optimization was performed. As a result of the optimization operations performed on the best fold of both models, the most ideal models were obtained with the optimized parameters. It has been trained from the beginning for all folds on both models using the optimized hyperparameters.

**6.2 Experiment 1 (Initial Training)**

During the initial training of models, the goal is to find the fold that gives the best result. For this purpose, both models were trained using the same hyperparameters and using the same number of layers, and nodes in the fully connected layers section. Parameters and values used during training are:

* Batch-size: 32
* Epoch: 50
* Optimizer: Adam
* Learning rate: 0.001
* Number of FC layer: 3 (in order 128, 32, 1)
* Batch Normalization: 2
* Activation: ReLu for first two layers and Sigmoid for last layer.

Table 6.1 shows five-fold cross validation results of two deep models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-Score |
| MobileNet | Fold 1: 0.828  **Fold 2: 0.850**  Fold 3: 0.815  Fold 4: 0.821  Fold 5: 0.835 | Fold 1: 0.825  Fold 2: 0.866  Fold 3: 0.835  **Fold 4: 0.881**  Fold 5: 0.848 | **Fold 1: 0.834**  Fold 2: 0.825  Fold 3: 0.780  Fold 4: 0.755  Fold 5: 0.807 | Fold 1: 0.829  **Fold 2: 0.845**  Fold 3: 0.807  Fold 4: 0.813  Fold 5: 0.827 |
| InceptionV3 | Fold 1: 0.757  Fold 2: 0.760  Fold 3: 0.736  **Fold 4: 0.777**  Fold 5: 0.759 | Fold 1: 0.813  Fold 2: 0.770  Fold 3: 0.794  Fold 4: 0.769  **Fold 5: 0.836** | Fold 1: 0.665  Fold 2: 0.748  Fold 3: 0.647  **Fold 4: 0.768**  Fold 5: 0.656 | Fold 1: 0.732  Fold 2: 0.758  Fold 3: 0.713  **Fold 4: 0.768**  Fold 5: 0.735 |

Table. 6.1 Inital Training Results

When the loss and accuracy graphs are examined in fold 2 (Figure 6.1, Figure 6.2), where both models give the best joint results, it seems that none of the model increases stably, since they do not flatten in the latest epochs. Similar graphs have been obtained as a result of other folds.

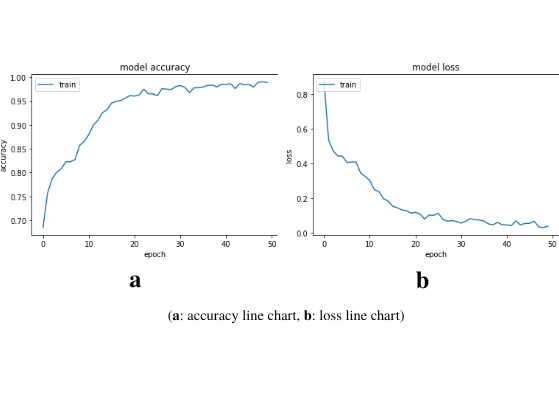


Fig. 6.1 MobileNet Fold 2 Initial Training accuracy and loss plots

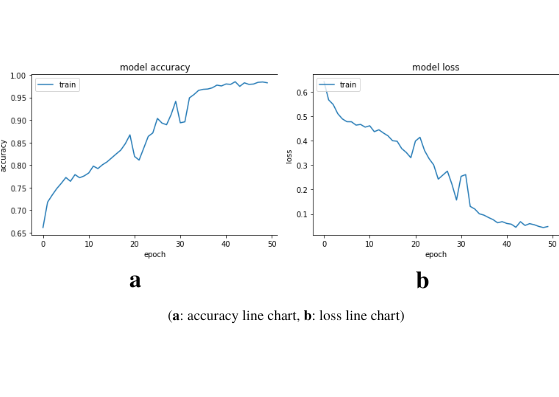


Fig. 6.2 Inception v3 Fold 2 Initial Training accuracy and loss plots

**6.3. Experiment 2 (Parameter Optimization)**

To achieve more consistent and higher performing models, hyperparameter tuning processes are applied to both models. In this process, the Bayesian Search technique was used to reduce the training time and to limit the search operations among the desired possibilities. In this process, it was tried to choose the common best fold for both models. In doing so, it was selected as the fold 2 test series, which gave the best result for MobileNet and the second-best result for Inception v3. While applying Bayesian Search, hyperparameter search was limited for learning rate, activation function, number of fully connected layers and number of nodes for layers. The intervals used for these parameters are listed below:

* Learning Rate: [1e-6 , 1e-1]
* Number of Layers: [1 , 10]
* Number of Dense for a Layer: [5 , 512]
* Activation Function: Sigmoid, ReLU

***6.3.1 Bayesian Search for MobileNet***

Table 6.2 shows the top-three results obtained when the Bayesian search was performed for MobileNet. When the table is examined in terms of learning rate, it seems that three best results were obtained in the e-4 and e-5 bands. In the same way, a smaller-layer structure leads to higher accuracy values. It is noteworthy that the best three results are obtained with the sigmoid activation function. On the other hand, the activation function can be changed to ReLU while the other parameters remain the same. In the subsequent process, this table can be compared with the newly obtained results that obtained with ReLU.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Learning Rate** | **Number of Layers** | **Number of Dense for a Layer** | **Activation Function** | **Accuracy** |
| 1.3e-4 | 1 | 91 | Sigmoid | 0.850 |
| 4.7e-5 | 2 | 96 | Sigmoid | 0.849 |
| 6.8e-5 | 1 | 512 | Sigmoid | 0.847 |

Table. 6.2 Bayesian Search Results for MobileNet

***6.3.2 Bayesian Search for Inception v3***

Table 6.3 shows the top-three results obtained when the search was performed for Inception v3. When the table is examined in terms of the learning rate, it seems that the three best results were obtained with the e-5 band. On the other hand, as in the MobileNet part, the best result was achieved with 1-layer and a sigmoid function. In contrast, Inception can give the highest results with higher-layer as well as smaller-layer structures. At the same time, all three results were obtained with the number of 512 dense, and the third-best result was obtained with the ReLU activation function. The next optimization process may be the creation of a new Bayesian Search using these parameter values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Learning Rate** | **Number of Layers** | **Number of Dense for a Layer** | **Activation Function** | **Accuracy** |
| 8.1e-5 | 1 | 512 | Sigmoid | 0.864 |
| 6.0e-5 | 4 | 512 | Sigmoid | 0.859 |
| 6.2e-5 | 3 | 512 | ReLu | 0.847 |

Table. 6.3 Bayesian Search Results for Inception v3

**6.3. Experiment 3 (Tuned Model Training)**

MobileNet and Inception v3 were trained again with 5-fold with parameters that allow them to give the best results obtained separately after the Bayesian Search process. The results obtained are as follows in Table 6.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-Score |
| MobileNet | Fold 1: 0.778  Fold 2: 0.842  Fold 3: 0.833  Fold 4: 0.842  **Fold 5: 0.864** | **Fold 1: 0.947**  Fold 2: 0.893  Fold 3: 0.773  Fold 4: 0.910  Fold 5: 0.905 | Fold 1: 0.592  Fold 2: 0.772  **Fold 3: 0.936**  Fold 4: 0.772  Fold 5: 0.806 | Fold 1: 0.728  Fold 2: 0.828  Fold 3: 0.846  Fold 4: 0.835  **Fold 5: 0.852** |
| InceptionV3 | Fold 1: 0.843  **Fold 2: 0.875**  Fold 3: 0.855  Fold 4: 0.848  Fold 5: 0.873 | Fold 1: 0.855  **Fold 2: 0.893**  Fold 3: 0.838  Fold 4: 0.812  Fold 5: 0.878 | Fold 1: 0.885  Fold 2: 0.854  Fold 3: 0.884  **Fold 4: 0.893**  Fold 5: 0.873 | Fold 1: 0.869  Fold 2: 0.873  Fold 3: 0.860  Fold 4: 0.850  **Fold 5: 0.875** |

Table 6.4 Tuned Model Training Results

***6.3.1 MobileNet Accuracy - Loss Plots***

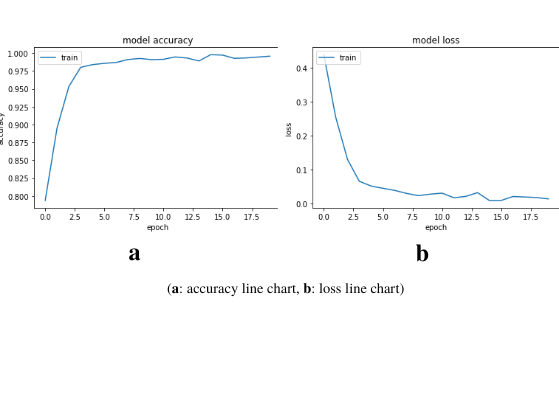


Fig. 6.3 MobileNet Fold 1, Tuned Model, Training Line Charts

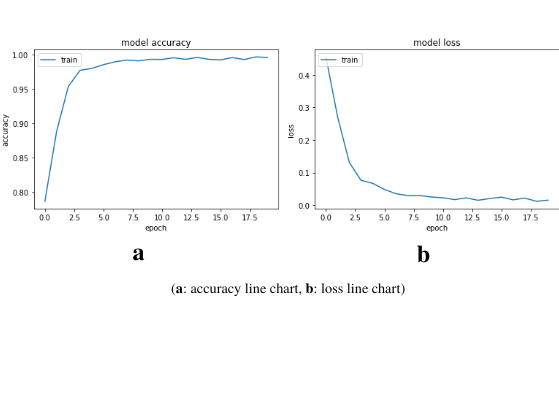
******

Fig. 6.4 MobileNet Fold 2, Tuned Model, Training Line Charts

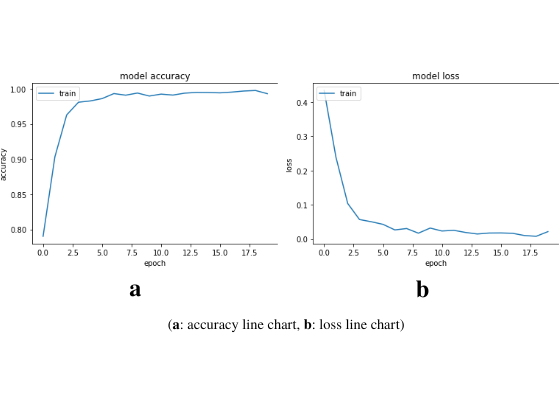
******

Fig. 6.5 MobileNet Fold 3, Tuned Model, Training Line Charts

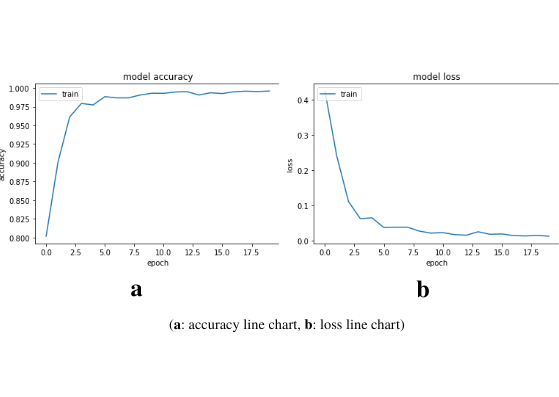
******

Fig. 6.6 MobileNet Fold 4, Tuned Model,Training Line Charts

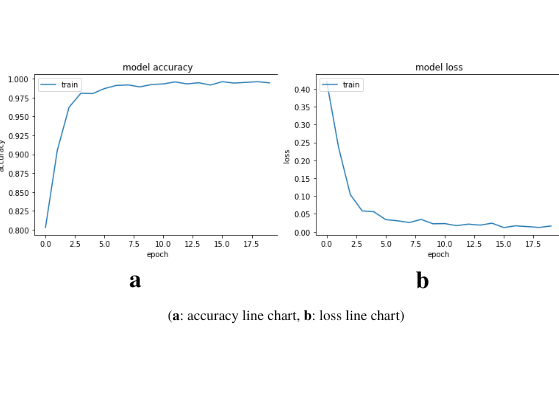


Fig. 6.7 MobileNet Fold 5, Tuned Model, Training Line Charts

***6.3.2 Inception Accuracy - Loss Plots***

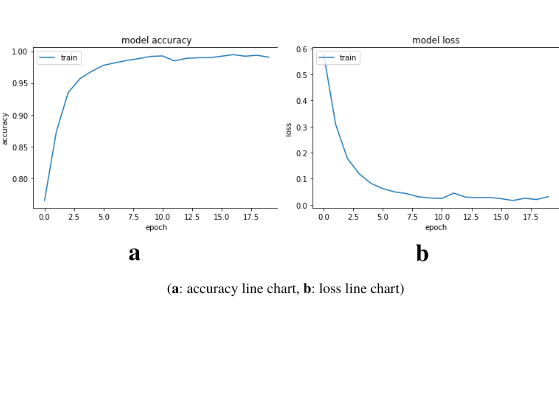


Fig. 6.8 Inception v3 Fold 1, Tuned Model, Training Line Charts

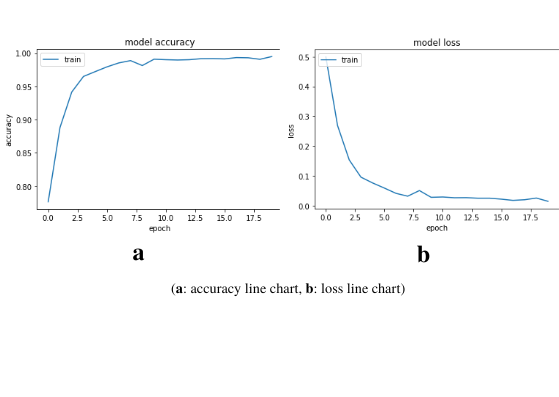
******

Fig. 6.9 Inception v3 Fold 2, Tuned Model, Training Line Charts

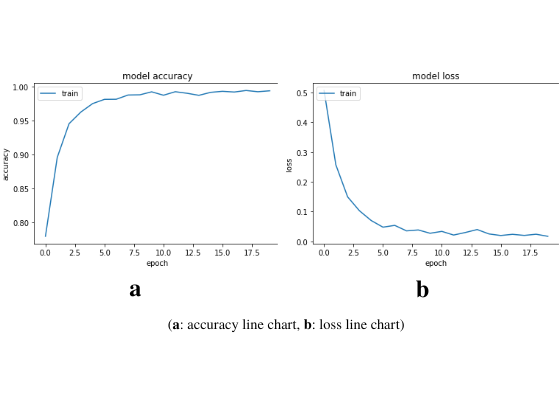
******

Fig. 6.10 Inception v3 Fold 3, Tuned Model, Training Line Charts

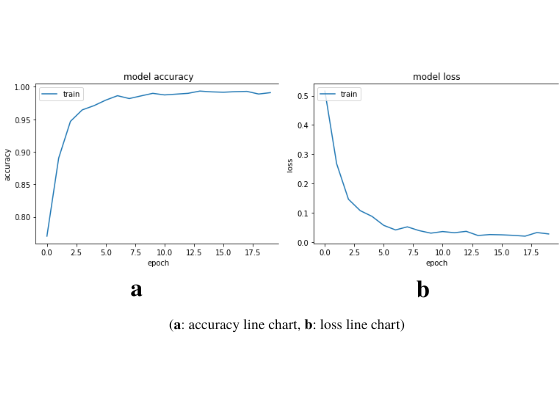
******

Fig. 6.11 Inception v3 Fold 4, Tuned Model, Training Line Charts

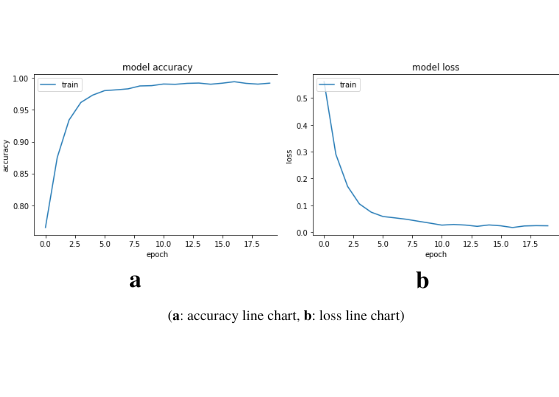
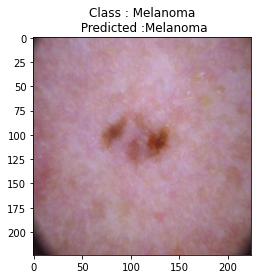


Fig. 6.12 Inception v3 Fold 5, Tuned Model, Training Line Charts

***6.3.3. Final Model Test Examples***

After testing both models in the mobile application, it was decided that the MobileNet deep learning model works faster. At the same time, there was a very small loss of accuracy rate compared to the Inception v3 model. The following images contain examples of true and false test results for the selected MobileNet model.

True Predicted:



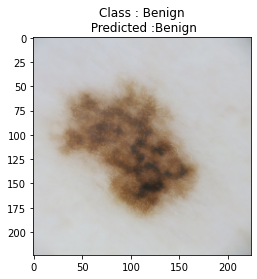
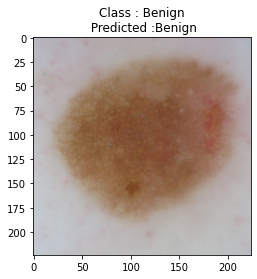
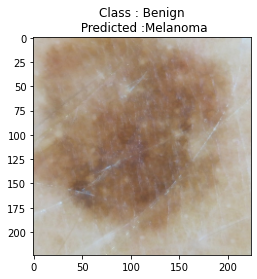
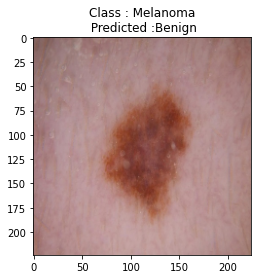


Fig. 6.13 True Predicted MobileNet Results

False Predicted:

metin, kare içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 6.14 False Predicted MobileNet Results

**6.4 Mobile Software Tests**

***6.4.1. Test Case 1***

Test Title: Image Resize

Test Scenario: Resizing the images taken from the user to the size required for evaluating in a deep learning model.

Test Explanation: The corresponding activity has a “*resizeImage”* function that resizes the images received from the user to the size required for deep learning model testing. This function accepts a bitmap image as a parameter and the needed size as integer type and returns a resized bitmap type image. As can be seen from Figure 6.13, the test image and the desired size were transmitted to the “*resizeImage”* function, and it was checked by the “*assertEquals”* function whether the dimensions of the new image returned could match the given size.

Test Situation: Passed

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 6.15 “*resizeImage*” Function Unit Test

***6.4.2. Test Case 2***

Test Title: Create Image Byte Array

Test Scenario: Transforming an image obtained from users via a *MainActivity* into byte array type for transmission to *ResultActivity* for testing on deep learning models.

Test Explanation: Bitmap images are not transmitted between the two classes, the relevant images are decoded to a byte array using the “*createImageByteArray”* method before the image transfer process occurs, despite the fact that images are used as bitmaps in the application. As can be seen from Figure 6.14, the “*createImageByteArray”* method accepts a bitmap image and an integer type as a parameter before calling the “*resizeImage”* function. It returns the resized image as a byte array after the translation process is completed.Test Situation: Passed

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 6.16 “*createImageByteArray*” Function Unit Test

***6.4.3. Test Case 3***

Test Title: Create Intent to Access Camera

Test Scenario: When the option to upload images with the camera on the main page is selected, the device camera will turn on.

Test Explanation: The Intent library, previously described in Chapter five, was used to access the device camera. Clicking on the camera card on the main page triggers the active listener “*onCameraCardClicked”* function, and thus an intent object is created that provides access to the camera, as in Figure 6.15. The same intent object was created here, and the return value of the function was checked using the “*assertEqual*” method.

Test Situation: Passed

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Fig. 6.17 “*onCameraCardClicked*” Function Unit Test

***6.4.4. Test Case 4***

Test Title: Load Model File

Test Scenario: After the user uploads the photo, the result page opens and the model where the user's image will be tested is uploaded.

Test Explanation: The deep learning model that has been trained and is ready for use is located in the assets folder. *loadModelFile* accesses this model file and makes the model available as a bytebuffer. The model load process was also performed here and it was tested whether it was installed with the isLoaded boolean variable.

Test Situation: Passed

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 6.18 “*loadModelFile*” Function Unit Test

***6.4.5. Test Case 5***

Test Title: Load Label List

Test Scenario: After the user uploads the photo, the result page opens and the label list with the model where the user's image will be classified is uploaded.

Test Explanation: The label list that has been created with the label names as a text file to use is located in the assets folder. *loadLabelList* accesses this text file and assigns these labels to list. The label list load process was also performed here and it was tested whether it was installed with the size variable which should be “2” as there are two labels.

Test Situation: Passed

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Fig. 6.19 “*loadLabelList*” Function Unit Test

**CHAPTER SEVEN**

**CONCLUSION**

Melanoma is a cancer caused by pigment-containing cells called melanocytes. It is one of the most dangerous cancers. Melanoma cancer causes nearly 55,000 deaths per year, accounting for 0.7 percent of all cancer deaths. On the other hand, recently, medical diagnostic systems with artificial intelligence support have become quite widespread. The aim of this project is to develop a mobile phone application that will help in the early diagnosis of melanoma skin cancer with the help of deep neural network models.

In the project, 4000 melanoma cancer and 4000 benign labeled skin lesion images selected from the dataset of the ISIC 2017 competition were used. In the preprocessing of the images, the normalization of the fish-eye images was performed first. Then, the factors that could create noise such as hairs on the images were removed from the image. The images have been resized to the appropriate dimensions for use in deep learning models to be used in training. When choosing deep learning models, algorithms that can work both in mobile applications and give high results were used. Inception v3 and MobileNet are pretrained models and they retrained with training data. The results obtained were compared with each other by using Accuracy, Precision, Recall, F1 Score metrics and model training graphs, and the most appropriate deep learning model was selected. The hyperparameters that will give the best results for both models in the training processes have been determined with the help of Bayesian Search algorithm. In this way, optimization of models is achieved. After testing both models in the mobile application, it was decided that the MobileNet deep learning model works faster. At the same time, there was a very small loss of accuracy rate compared to the Inception v3 model.

A mobile application has been developed in Android Studio by using the Kotlin software language. The selected model has been made available to work on mobile devices with Tensorflow Lite. With the help of the necessary libraries, access to the camera and gallery of the device is provided and an image is taken from the user, and the corresponding lesion part of this image is cropped. After all these operations, the final image obtained is tested by the deep learning model and the result is obtained.

**REFERENCES**

Jinen Daghrir, Lotfi Tlig, Moez Bouchouicha, Mounir Sayadi (2021). Melanoma skin cancer detection using deep learning and classical machine learning techniques: A hybrid approach. *HAL (hal-03172718)*.

Tsong-Long Hwang, Woan-Ruoh Lee, Shu-Chiou Hua, and Jia-You Fang (2007). Cisplatin encapsulated in phosphatidylethanolamine liposomes enhances the in vitro cytotoxicity and in vivo intratumor drug accumulation against melanomas. *Journal of dermatological science, 46(1):11–20*.

Russell S. Berman, Jennifer A. Stein, David Polsky, Ronald O. Perelman (2019). Towards Automated Melanoma Detection with Deep Learning: Data Purification and Augmentation.

Adekanmi A. Adegun, Serestina Viriri (2019). Towards Automated Melanoma Detection with Deep Learning: Data Purification and Augmentation.

Lokesh Singh, Rekh Ram Janghel and Satya Prakash Sahu (2021). A Deep Learning-Based Transfer Learning Framework for the Early Detection and Classification of Dermoscopic Images of Melanoma. *(Biomedical & Pharmacology Journal, September 2021)*.

Abhinav Sagar, Dheeba Jacob (2020). Convolutional Neural Networks for Classifying Melanoma Images.

Simon Kalouche (2020). Vision-Based Classification of Skin Cancer using Deep Learning. *(Stanford University)*

[Kaiming He](https://arxiv.org/search/cs?searchtype=author&query=He%2C+K), [Xiangyu Zhang](https://arxiv.org/search/cs?searchtype=author&query=Zhang%2C+X), [Shaoqing Ren](https://arxiv.org/search/cs?searchtype=author&query=Ren%2C+S), [Jian Sun](https://arxiv.org/search/cs?searchtype=author&query=Sun%2C+J) (2015). Deep Residual Learning for Image Recognition

[Ian J. Goodfellow](https://arxiv.org/search/stat?searchtype=author&query=Goodfellow%2C+I+J), [Jean Pouget-Abadie](https://arxiv.org/search/stat?searchtype=author&query=Pouget-Abadie%2C+J), [Mehdi Mirza](https://arxiv.org/search/stat?searchtype=author&query=Mirza%2C+M) (2014). Generative Adversarial Networks. *(University of Montreal)*

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision*, 115(3), 211-252.*

Adil H. Khan,vD.N.F. Awang Iskandar, Jawad F. Al-Asad and Samir El-Nakla (2021). Classification of Skin Lesion with Hair and Artifacts Removal using Black-hat Morphology and Total Variation*, International Journal of Computing and Digital Systems.*

Samadhi P. K., Wickrama Arachchilage and Ebroul Izquierdo (2020). Deep-learned faces: a survey.

Medium (2021).  *What is VGG16? — Introduction to VGG16.* Retrieved 30 March 2022,from https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615

Medium (2020). *Image Classification with MobileNet.* Retrieved 30 March 2022,from https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470

[Andrew G. Howard](https://arxiv.org/search/cs?searchtype=author&query=Howard%2C+A+G), [Menglong Zhu](https://arxiv.org/search/cs?searchtype=author&query=Zhu%2C+M), [Bo Chen](https://arxiv.org/search/cs?searchtype=author&query=Chen%2C+B), ... & [Hartwig Adam](https://arxiv.org/search/cs?searchtype=author&query=Adam%2C+H) (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

Christian Szegedy, Vincent Vanhoucke, … & Jonathon Shlens (2015). Rethinking the Inception Architecture for Computer Vision.

Google Cloud (2022) *Advanced Guide to Inception v3.* Retrieved 23 April 2022, from https://cloud.google.com/tpu/docs/inception-v3-advanced