

T.C.
University of Aegean
Department of Electrical and Electronic Engineering

Bachelor's Final Project Report

Deep Learning for Breast Cancer Detection from Thermal Images

Breast Cancer Detection from Thermal Images Using Deep Learning

Kaan EVRAN
05200000436

05200000436@ogre n times. Robe. End. Suddenly

Project Consultant: Prof.Dr. Dr. Mehmet Engin

Undergraduate project Information Form

E.Ü. Electrical-Electronics Engineering Department Undergraduate project Information Form	
1- Project Completion Period: 2023-2024	2- Report Date: 22/05/2024
3- Start and End dates of the project: 23/10/23 21/05/2024	
4-Name of the Project: Detecting Breast Cancer with Deep Learning with thermal images. Total Cost of the Project: 327810,12 TL	
5. Please specify your project team. (Select one.) <input checked="" type="checkbox"/> Interdisciplinary team work, <input checked="" type="checkbox"/> Multidisciplinary team work, <input checked="" type="checkbox"/> Individual work	
Students to carry out the project and contact information (address, email, tel.) 	
7--The organization where the project is carried out: Ege University Faculty of Engineering- Electrical and Electronics Engineering Department	
8--The name, address and amount of support of the supporting organization (s): No support was received from an organization.	
9--Self: A model that detects disease by using deep learning methods for breast cancer detection developed. The thermal images taken from five different angles were passed through the autocoder with evolution and inclusion was applied to the outputs and a healthy and patient classification was made. In addition, the front -time -free images for a patient were examined with a different model with a front -trained eviientnetb5 and LSTM layer. In Python software, Keras and Tensorflow libraries were used. Artificial neural network models using TPU on Google Colab Cloud System has been trained. Keywords: deep learning, breast cancer detection, autocoder, flexible neural network (CNN), long short-term memory (LSTM)	
12- Advisor's Professor Name/Surname and Opinion: Prof. Dr. Mehmet Engin	
13-Department of Department:	
14--SUCCESS STATUS OF THE PROJECT:	The project received by the project:

Preface

With the development of software and equipment, the usage areas of artificial intelligence are increasing day by day. The most commonly used sub -branch of artificial intelligence in practice is machine learning. Deep Learning method, which includes the use of artificial neural networks, can achieve high success even in very complex problems. For this reason, deep learning systems trained with Supervised Learning can be used in the field of medicine, especially in the field of diseases.

In this project, a study on the detection of breast cancer from thermal images by deep learning method was presented. Thus, it is aimed to contribute to the later studies in this field. The models offered are developed for the analysis of different medical images or different visual signals in general, although they have been developed for thermal image processing.

From the thermal images with deep learning Breast Cancer Detection

Kaan Evran

Ege University Department of Electrical
and Electronics Engineering
Izmir, Türkiye

05200000436@ogre n times. Robe. End. Suddenly

Essence - cancer -related deaths due to cancer due to cancer. In this study, a motor detection was developed by using deep learning methods for breast cancer detection. For each patient, the intertired images taken from five different angles were passed through evolution -centeredotocode, and a healthy and sick classification was made. Also for a patient, a front -time -intermittently taken images for a patient (Effeientnetb5) and a different model containing LSTM layer. Artificial neural network models are trained on Google Colab Cloud System.

Key Words - Learning, Breast Cancer Detection, Autocoter, Evolutionized Nervous Network (CNN), Long Short Short Survival (LSTM)

Abstract— Cancer-related deaths are mostly due to breast cancer. In this study, a disease detection model was developed using deep learning methods for breast cancer detection. Thermal images taken from five different angles for each patient were passed through the convolutional autoencoder, fusion was applied to the outputs, and healthy and patient classification was made. In addition, frontal time-lapsed images of a patient were examined with a different model containing a pre-trained convolutional neural network (EfficientNetB5) and LSTM layer. Keras and Tensorflow libraries were used in Python software. Artificial neural network models were trained on the Google Colab cloud system.

Keywords—deep learning, breast cancer detection, autoencoder, convolutional neural network (CNN), long short-term memory (LSTM)

Entrance and Literature Summary

In the literature, deep learning for detection of thermal images for the detection of breast cancer Tsietsos et al. [1]. Roslidar et al. [2] He examined the potential of thermographic imaging and deep learning models to effectively detect breast cancer.

Mambou et al. [3] He worked on breast cancer detection using infrared thermal images.

In the study, the feature vector was removed from the image using the preliminary trained Inception v3 model, which is a nerve with a single hidden layer.

Machine Learning Method Support Vector Machine (Support

Classification has been performed using Vector Machine - SVM. The classification results show confidence values ranging from 0.78 to 0.96.

Okal [4] has performed feature extraction by passing thermal images through different filtering operations, and then classified the obtained features using SVM. In this study, accuracy of 90%, recall of 86%, and specificity of 93% were obtained as output values.

Cabioğlu [5] has colored the front images on the Visual Lab image set using a jet color map and achieved 94.3 %accuracy by using the modified Alexnet front educated model in MATLAB environment.

Kirişken [6] The Resnet50 has used the preliminary educated model according to their own requirements.

With the replication process of Comsol Multiphysics program and Visual LAB data set, two different experimental sets were created and the highest test accuracy was 98.44 %. Ekici and Jawzal [7] have been used with an evolutionary neural network -CNN -CNN.

The results obtained have a 98.95% accuracy rate. Zuluaga-Gomez [8] has used a recurrent neural network, comparing the results obtained from different optimization algorithms such as Adam, RMSprop, and SGD, as well as different hyperparameters, to select the most ideal one. The best result obtained has 92% accuracy and 98% sensitivity.

Sánchez-Cauce et al. [9] have worked on a dataset that has three different thermal images taken from each patient. Classification has been made by adding the feature vectors obtained from the recurrent neural network of the three different images. The results obtained have 97% accuracy and 83% sensitivity (sensitivity) rates.

Yousefi et al. [10] Used an evolutionary neural network to remove low-size properties. In the thermal images used here, each example consists of 20 consecutive appearances of a patient in a dynamic state. Here, 20 images were applied to a 3-channel image by applying matrix factorization to 20 images and then the Random Forest (Random Forest) method was used. In addition, clinical data were included in the classification and the highest 78.2 %accuracy was obtained. Mammootill et al. [11] In their study, the thermal images taken from five different angles were classified by combining the properties obtained by passing through a different evolutionary nervous network for each angle. In this study, 85.4 %accuracy was obtained, then the clinical data was included in the model and this value was obtained as 93.8 %. Mohamed et al. [12] has automatically separated and isolated the chest area from thermal images using the U-Net network. Subsequently, feature extraction and classification were performed with the neural network. As a result, a sensitivity of 99.33%, specificity of 100%, and accuracy of 98.67% were achieved. Civilib et al. [13] has surrounded the potentially tumor-formation areas in thermal images with a bounding box to improve performance. Then, pre-trained neural networks ResNet-50 and ResNet-101 were trained using transfer learning for thermal images, and results were obtained. Tests resulted in accuracy rates of 97.1% and 96.2% for these networks, respectively. Alsaedi et al. [14] have used a hybrid method based on the difference in electromagnetic power transmission between healthy and sick tissue. They have obtained thermal images by heating with microwave radiation source, followed by classification using a rotating network (CNN). Healthy and sick chest differentiation training / test set ratios have achieved a accuracy ranging from 89% to 94%. The studies mentioned above have been conducted with images taken using the Static Infrared Thermography (SIT) [1] protocol. In this protocol, a thermal image is taken at a specific moment while the patient remains stationary. In addition to this, - DITE) [1] protocol images, patient's breast After the area is cooled, time is taken intermittently. Gonzalez-Hernandez et al. [18] He examined the studies used by images taken by DIT protocol. Ohashi and Uchida [19] examined the images taken by DIT protocol, while the images of the DIT images were applied. 82 %accuracy was obtained with the 54 %accuracy obtained for SIT.

Abdel-Nasser et al. [20] Using the "Learning-to-Shrank" machine learning method and texture analysis on DIT images, it obtained 95.8 %accuracy and 94.6 %certainty. Silva et al. [21] For DIT images, the main data set has achieved 100 %accuracy using time series methods on 68 images separated from within. Network -cnn) is used. Mammootill et al. [11]

View Thermal Images Using Convolutional Neural Networks ". The following differences are between the current project and this study:

- a) In this article, clinical data were also included in the classification. In the current project, it is more general by trying to make classification based on thermal images without using clinical data. and to obtain a applicable model is working. Clinical data usage is not preferred because the model to be created will be restrictive, because the record of clinical data for the person to be scanned in the field getirir.
- b) In this study, rectangular images are converted into squares and used, whereas in the current study, the images are used in their original rectangular form. Besides, an attempt has been made to prevent data loss that may occur in the images.
- c) Standard recurrent neural networks are used to obtain the feature vectors to be combined in this study. In the current study, feature vectors are obtained using autoencoders. To successfully train recurrent neural networks, it is necessary to have as large a dataset as possible. Since the size of the dataset available for training is limited, the use of autoencoders is preferred. Autoencoders are more successful in this situation. Especially for dimensionality reduction, autoencoders are more successful than the pca method. is ongoing.
- d) Due to the use of clinical data in this study, examples with conflicting information about the patient's age have been extracted. However, in the current study, since this data was not used, these examples could also be included in the study.
- e) In this study, the data series was divided into 80% training and 20% testing. In the current study, the data was divided into 70% training and 30% testing, with a broader test set. This way, the performance of the resulting model can be more accurately tested.
- f) In this study, the images were combined from only 3 different perspectives. In the current study, combinations were made for both 3 and 5 images. For the analysis of DIT protocol images in the current project, the movement in videos

With a similar method, a patient's sequential DIT images were first converted into a video and then the disease was detected with evolution nervous network and LSTM layer. In the studies of DIT on breast cancer, such an approach has not been tried before.

Introduction and Working

Principles of the System

Silva et al. [31] a data set prepared by images with images taken with SIT and DIT protocols were used. Thermal images are recorded as matrices in txt files.

SIT protocol and DIT protocol images were converted into JPG files using "viridis" and "prism" color maps respectively and separate deep learning models were created for them. (Fig. 1 and 2)

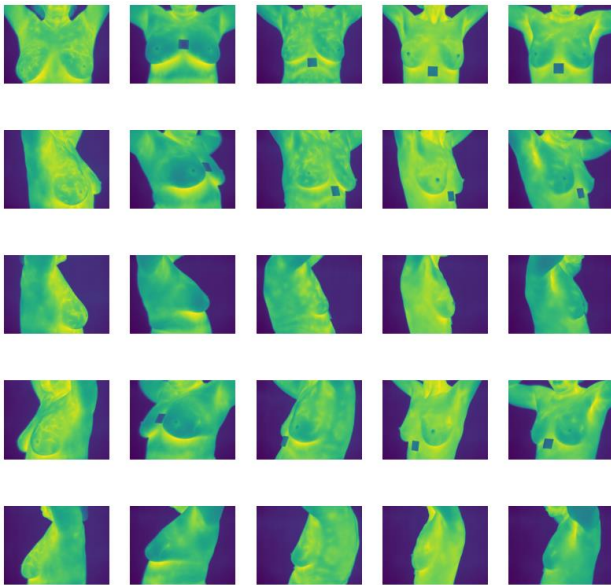


Fig. Images of 1.5 different patients from 5 different angles (SIT)

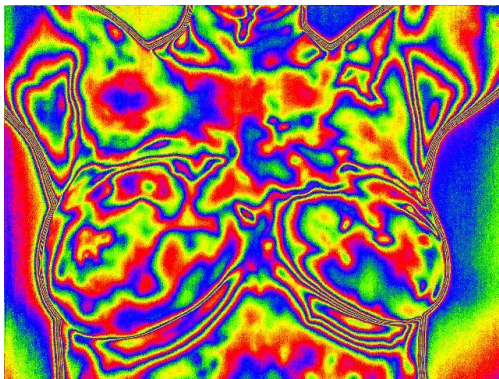


Fig. 2.DIT protocol image

70% of the data is used for training, and 30% for test data during the training of deep learning models. SIT protocol images are processed through separaterotational autocoder structures, converted into featurevectors, and then passed through an artificial neural networklayer to add output vectors at the end, applying imagecombining process. The combined vector is then passed through the final neural network layer to classify patients. The layer structure of the used autocoder is given in Fig. 3. The entire system is shown in Fig. 4. DIT protocol images consist of 20 images takenat timed intervals for a specific patient. An approach applied in video classification for inspection has been adopted [32].

In the Evriyli LSTM model, the Timedistributed layer takes an input of an evolving neural network. The input was used as an input. Each example consists of 20 image frames, but 10 frames are selected equally intermittently, they are passed through the evolution nervous network, the output straightening process is converted to vector and given to the LSTM layer. (Fig. 5) In addition, Dropout was added to the dense layers of the LSTM layer output to prevent the memorization problem in the system.

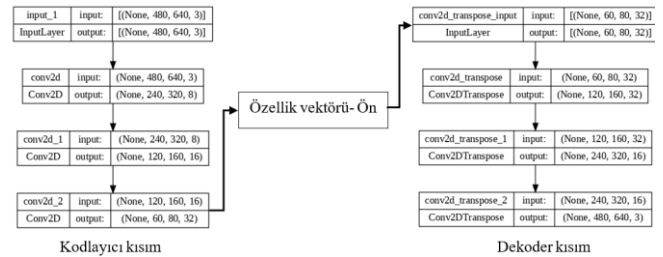


Fig. 3.An image autocode layer structure

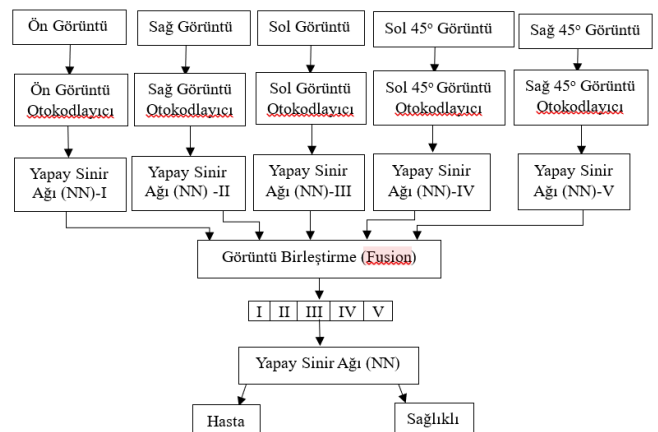


Fig. 4.The merge model scheme

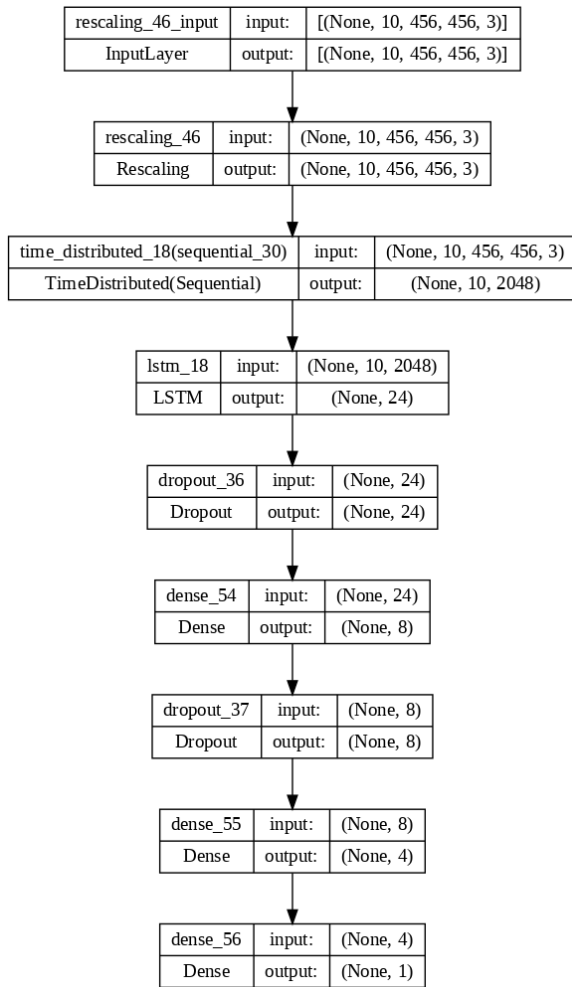


Fig. 5.Evolutionary LSTM Model Layer Structure

System Analysis Through Software-Based Simulations

Deep learning models have been trained on TPU in the Google Colab environment. (Fig. 6) Then, the classification results of the models on the test data have been examined with the trained systems. The results have been given in the form of a confusion matrix. (Fig. 7)

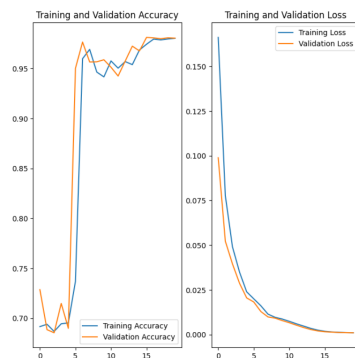


Fig. 6.Otokodlayıcı training success graph (Preview)

TN	FP
FN	TP

Fig. 7.Confusion matrix

The SIT protocol image fusion results with the single preview image and the fused results of 3 and 5 images are given in Figs. 8 and 9, respectively. In the case of image fusion, the number of false positives (FP) has decreased and the number of false negatives (FN) has increased compared to the single preview image. Since the system uses multiple images instead of a single image to determine whether the sample input is a patient, the number of FP decreases, but the system becomes more selective in deciding whether the person is sick, so the number of FN increases. As expected, the 5-image fusion provides better results than the 3-image fusion in terms of FP, and this number is lower. After the SIT protocol images, the images of the DIT protocol were examined. (Fig. 10) The results obtained are less successful than image junction.

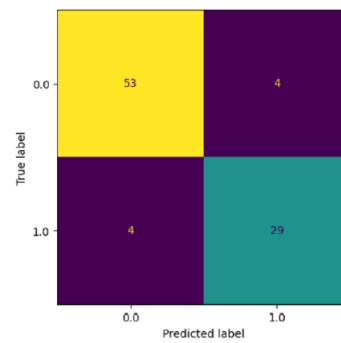


Fig. 8. Classification with only the pre -image

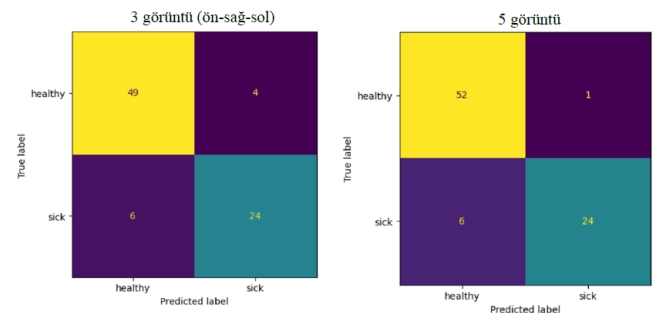


Fig. 9. Classification with Correction Meeting

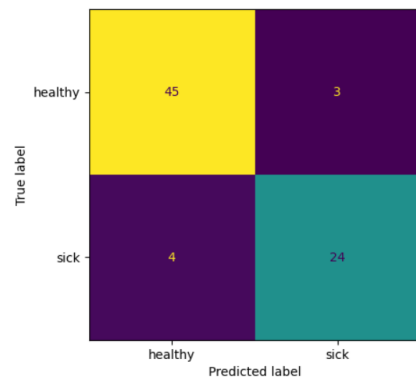


Fig. 10. Classification of DIT images with an evolving LSTM

Performance, according to the obtained values, has been examined in Table 1 based on the standard metrics used for measurement.

TABLE I. ANALYSIS RESULTS (%)

Metrics	Adjustable LSTM	Single image	image merging	
			3 images	5 image
Precision (Precision) (%)	89	88	86	96
Sensitivity (Recall) (%)	86	88	80	80
Accuracy (Accuracy) (%)	91	91	88	92
F1 Score (%)	87	88	83	87

Results and Discussion

As a result of the study, a deep learning model was developed to detect breast cancer from thermal images. The system has achieved high success in terms of certainty, but is open to development in sensitivity. In general, thermal images are examined in the study, but the models presented are also adapted for the analysis of different medical imaging or signals and can contribute to these areas. For example, brain waves or speech sound signals are converted into spectrograms that can be shown as 2 -dimensional color images, and evolutionic nervous network models presented in the study can be adapted to them.

Source

- [1] World Health Organization, "Breast Cancer," who.int, Mar. 13, 2024. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/breast-cancer>. [Accessed Apr. 11, 2024]
- [2] R. Lawson, "Implications of surface temperatures in the diagnosis of breast cancer," Canadian Medical Association Journal vol. 75, pp. 309-310, Aug. 1956.
- [3] B. Rakhi, s. Gotar, s. G Chaudhari, m. B. Rakhi, s. Gotarkar, and S. G. Choudhari, "Thermography as a Breast Cancer Screening Technique: A Review Article," Cureus Journal of Medical Science, vol. 14, no. 11, Nov. 2022, doi: <https://doi.org/10.7759/cureus.31251>.
- [4] D. Tsietso, A. Yahya, and R. Samikannu, "A Review on Thermal Imaging-Based Breast Cancer Detection Using Deep Learning," Mobile Information Systems, vol. 2022, pp. 1-19, 2022, doi: <https://doi.org/10.1155/2022/8952849>.
- [5] R. Roslidar, A. Rahman, R. Muharar et al., "A review on recent progress in thermal imaging and deep learning approaches for breast cancer detection," IEEE Access, vol. 8, pp. 116176-116194, 2020.
- [6] S. J. Mambou, P. Maresova, O. Krejcar, A. Selamat, and K. Kuca, "Breast cancer detection using infrared thermal imaging and a deep learning model," Sensors, vol. 18, no. 9, p. 2799, 2018.
- [7] G. Okal, "Identification of breast cancer risk using thermal imaging and machine learning," Master's thesis, Faculty of Science, Pamukkale University, Denizli, Turkey, 2019.
- [8] Ç. Cabioglu, "Breast cancer diagnosis from thermal images," Master's thesis, Faculty of Science, Başkent University, Ankara, Turkey, 2020.
- [9] T. Kirişken, "Deep Learning Classification of Thermal Images for Early-Detection of Breast Cancer," Master's Thesis, Science Institute, Ege University, İzmir, Turkey, 2022.
- [10] S. Ekici and H. Jawzal, "Breast cancer diagnosis using thermography and convolutional neural networks," Medical Hypotheses, vol. 137, p. 109542, 2020.
- [11] J. Zuluaga-Gomez, Z. Al Masry, K. Benagounne, S. Meraghni, and N. Zerhouni, "A CNN-based methodology for breast cancer diagnosis using thermal images," Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, vol. 9, no. 2, pp. 131-145, 2020.
- [12] R. Sánchez-Cauce, J. Pérez-Martin, and M. Luque, "Multi-input convolutional neural network for breast cancer detection using thermal images and clinical data," Computer Methods and Programs in Biomedicine, vol. 204, p. 106045, 2021.
- [13] B. Yousefi, H. Akbari, M. Hershman, S. Kawakita, H.C. Fernandes, C. Ibarra-Castaneda, S. Ahadian, X. P. V. Maldague, "SPAER: Sparse Deep Convolutional Autoencoder Model to Extract Low Dimensional Imaging Biomarkers for Early Detection of Breast Cancer Using Dynamic Thermography," Applied Sciences, vol. 11, no. 7, p. 3248, 2021, doi: <https://doi.org/10.3390/app11073248>.
- [14] M. Mammoottil, I. J. Kulangara, A. S. Cherian, P. Mohandas, K. Hasikin, and M. Mahmud, "Detection of breast cancer from five-view thermal images using convolutional neural networks," Journal of Healthcare Engineering, vol. 2022, pp. 115, 2022.
- [15] E. A. Mohamed, E. A. Rashed, T. Gaber, and O. Karam, "Deep learning model for Fully Automated Breast Cancer Detection System from thermograms," PLOS ONE, vol. 17, no. 1, 2022, doi: [10.1371/journal.pone.0262349](https://doi.org/10.1371/journal.pone.0262349)
- [16] S. Civilibal, K. K. Cevik, A. Bozkurt, "A deep learning approach for automatic detection, segmentation and classification of breast lesions from thermal images," Expert Systems with Applications, vol. 212, p. 118774, 2023.
- [17] D. Alsaedi, A. Melnikov, K. Muzaffar, A. Mandelis and O. M. Ramahi, "A Microwave-Thermography Hybrid Technique for Breast Cancer Detection," IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology, vol. 6, no. 1, pp. 153-163, 2022, doi: [10.1109/JERM.2021.3072451](https://doi.org/10.1109/JERM.2021.3072451).
- [18] J. L. Gonzalez-Hernandez, A. N. Recinella, S. G. Kandlikar, D. Dabydeen, L. Medeiros, and P. Phatak, "Technology, application and potential of dynamic breast thermography for the detection of breast cancer," International Journal of Heat and Mass Transfer, vol. 131, pp. 558-573, 2019.
- [19] Y. Ohashi and I. Uchida, "Applying dynamic thermography in the diagnosis of breast cancer," IEEE Engineering in Medicine and Biology Magazine, vol. 19, no. 3, pp. 42-51, May-June 2000, doi: [10.1109/51.844379](https://doi.org/10.1109/51.844379).
- [20] M. Abdel-Nasser, A. Moreno, and D. Puig, "Breast Cancer Detection in Thermal Infrared Images Using Representation Learning and Texture Analysis Methods," Electronics, vol. 8, no. 1, p. 100, Jan. 2019, doi: <https://doi.org/10.3390/electronics8010100>.
- [21] T. A. E. da Silva, L. F. da Silva, D. C. Muchaluat-Saade, and A. Conci, "A Computational Method to Assist the Diagnosis of Breast Disease

- Using Dynamic Thermography.” *Sensors*, vol. 20, no. 14, p. 3866, Jul. 2020, doi: <https://doi.org/10.3390/s20143866>.
- [22] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2021.
- [23] R. Pramoditha, “The Concept of Artificial Neurons (Perceptrons) in Neural Networks,” [towardsdatascience.com](https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc), Dec. 29, 2021. [Online]. Available: <https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc>. [Accessed Apr. 19, 2024]
- [24] IBM, “What Are Neural Networks?,” [www.ibm.com](https://www.ibm.com/topics/neural-networks), 2023. [Online]. Available: <https://www.ibm.com/topics/neural-networks>. [Accessed Apr. 19, 2024]
- [25] E. Grossi and M. Buscema, “Introduction to artificial neural networks,” *European Journal of Gastroenterology & Hepatology*, vol. 19, no. 12, pp. 1046–1054, Dec. 2007, doi: <https://doi.org/10.1097/meg.0b013e3282f198a0>.
- [26] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional Neural networks: an Overview and Application in Radiology,” *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, June 2018, doi: <https://doi.org/10.1007/s13244-018-0639-9>.
- [27] J. Jain, “What Are Tensors?,” [medium.com](https://medium.com/@jayeshjain_246/what-are-tensors-495cf37c18e6), January 13, 2023. [Online]. Available: https://medium.com/@jayeshjain_246/what-are-tensors-495cf37c18e6. [Accessed Apr. 20, 2024]
- [28] S. Bishayee, “The Basic Concept of Autoencoder — The Self-supervised Deep Learning,” [medium.com](https://medium.com/@soumallya160/the-basic-concept-of-autoencoder-the-self-supervised-deep-learning-454e75d93a04), Apr. 13, 2023. [Online]. Available: <https://medium.com/@soumallya160/the-basic-concept-of-autoencoder-the-self-supervised-deep-learning-454e75d93a04>. [Accessed Apr. 20, 2024]
- [29] M. Seeland and P. Mäder, “Multi-view classification with Convolutional Neural Networks,” *PLOS ONE*, vol. 16, no. 1, 2021, doi: [10.1371/journal.pone.0245230](https://doi.org/10.1371/journal.pone.0245230).
- [30] Z. Khademi, F. Ebrahimi, and H. M. Kordy, “A transfer learning-based CNN and LSTM hybrid deep learning model to classify motor imagery EEG signals,” *Computers in Biology and Medicine*, vol. 143, p. 105288, Apr. 2022, doi: [10.1016/j.compbiomed.2022.105288](https://doi.org/10.1016/j.compbiomed.2022.105288).
- [31] L. F. Silva et al., “A New Database for Breast Research with Infrared Image,” *Journal of Medical Imaging and Health Informatics*, vol. 4, no. 1, Pages 92–100, March. 2014, doi: <https://doi.org/10.1166/jmihi.2014.1226>.
- [32] P. Ferlet, “Training neural network with image sequence, an example with video as input,” [medium.com](https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f), Dec. 05, 2019. [Online]. Available: <https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f>. [Accessed Apr. 20, 2024]

Table of Contents:

Figure List:	10
Table List	12
Owner	13
Abstract	14
1) Introduction	15
2) Development	22
3) Result	47
4) References	50
5) License Completion Project Self-Assessment Form	54
6) Attachments	56
EC-1: Data Download Code	57
EC-2 JPG Conversion Code	63
EC-3: Getting Tensorflow Data Set Code	65
Image Filtering Code	67
Automatic Coder Code (v1)	70
Auto Encoder Code (v2)	74
Auto Encoder Training Set Acquisition Code	78
Image Fusion Code (SVM)	80
Image Merging Code (NN)	85
Rotatable LSTM Network Code	93
AVI Video Conversion Code EK-11	98

Figure List:

Figure 1. Method flowchart	21
Figure 2. Artificial Intelligence, Machine Learning, and Deep Learning Hierarchy ...	22
Figure 3. Training the Machine Learning System	23
Figure 4. Structure and connections of the artificial neuron	23
Figure 5. Artificial neural network and layers	24
Figure 6. Loss Function and Learning Process	25
Figure 7. Shifting neural network	25
Figure 8. Convolution operation	26
Figure 9. Tensor dimensions	26
Figure 10. Convolutional Autoencoder structure	27
Figure 11. Image Fusion (Fusion) Process	28
Figure 12. LSTM Cell	28
Figure 13. Rotational LSTM	29
Figure 14. Data set user interface	30
Figure 15. The hierarchical folder structure of the data set in the Google Drive environment ...	30
Figure 16. Views from 5 different angles of 5 different patients	31
Figure 17. Differences in the number of examples between image modes due to dataset incompleteness ...	32
Figure 18. Autoencoder used for the front view (v1)	33
Figure 19. Autoencoder used for the front view (v2)	33
Figure 20. Image Fusion Model Flow Diagram (SVM)	34
Figure 21. Deep Learning Model Flow Diagram (Artificial Neural Network)	35
Figure 22. DIT protocol image (prism color map)	36
Figure 23. DIT protocol evolving LSTM model	37
Figure 24. Operation of the Time Distributed Layer	37
Figure 25. Layers Added on EfficientNetB5	38
Figure 26. Confusion Matrix	38
Figure 27. Otokodlayıcı Verification Graph (Preview Image)	39
Figure 28. Classification with only the preview image (Otokodlayıcı v1)	39
Figure 29. Classification with image fusion (SVM)	40
Figure 30.	40
Figure 31.	41
Figure 32. Classification with image fusion (NN- Autoencoder v2)	42
Figure 33. Classification with image fusion (SVM- Autoencoder v2)	42

Figure 34. Classification of DIT images using an Evolvable LSTM (12-cell LSTM) ... 43

Figure 35. Image classification with a variable-length LSTM (24-cell LSTM) 43

Table List

Table 1. Cost Analysis	20
Table 2. Performance Metrics	21
Table 3. Performance metrics for patient classification using only the preview ...	44
Table 4. Performance metrics of image fusion using SVM (autoencoder v1) ...	45
Table 5. Performance metrics of image fusion using SVM (autoencoder v2) ...	45
Table 6. Image Marketing Performance Metriks with NN (Autocoter V1)	45
Table 7. Image Marketing Performance Metriks with NN (Autocoter V2)	46
Table 8.	46

Owner

Breast cancer is the most common cause of cancer -related deaths in women. Although mammography is still the most widely used method for breast cancer detection, there are studies for the use of thermal images in disease. In this study, a model that uses deep learning methods for detection of thermal images assistance was developed. For each patient, thermal images taken from five different angles were passed through evolving autocoder and a healthy and patient classification was made using the support vector machine (SVM) and artificial neural network. Two different types of autocoders were examined and they were compared in terms of success. In addition, for a patient, the front -time -intermittently taken images were examined with a different model containing an edujicientnetb5 and LSTM layer.

Python software using the Keras and Tensorflow libraries has been used for analysis. Deep learning models have been trained on the Google Colab cloud system. According to the results obtained, image fusion with artificial neural network has given more successful results than the Support Vector Machine. Additionally, it has been observed that the autoencoder works more successfully when size reduction is done by changing the step parameter (stride) of the convolution layer instead of the Pooling layer.

Keywords: deep learning, breast cancer detection, autoencoder, convolutional neural network (CNN), long short-term memory (LSTM)

Abstract

The most common cause of cancer-related deaths among women is breast cancer. Although mammography is still the most commonly used method for breast cancer detection, there are studies on the use of thermal images in disease detection. In this study, a model using deep learning methods was developed for breast cancer detection with the help of thermal images. For each patient, thermal images taken from five different angles were passed through a convolutional autoencoder and fusion was applied to the resulting vectors. Healthy and sick classification was made using Support Vector Machine (SVM) and artificial neural network. Two different types of autoencoders were examined and compared in terms of success. In addition, frontal time-lapsed images of a patient were examined with a different model containing a pre-trained convolutional neural network (EfficientNetB5) and LSTM layer. Keras and Tensorflow libraries of Python software were used for analysis. Deep learning models are trained on Google Colab cloud system. According to the results obtained, image fusion with artificial neural network gave more successful results than Support Vector machine. Additionally, it has been observed that the autoencoder works more successfully when size reduction is done by changing the step parameter (stride) of the convolution layer instead of the Pooling layer.

Keywords: deep learning, breast cancer detection, autoencoder, convolutional neural network (CNN), long short-term memory (LSTM)

1) Introduction:

Today, the use of artificial intelligence, which has become widespread in every field, is increasing in the field of medicine. The most commonly used arm of artificial intelligence in practice is deep learning. There are many studies in different fields of detection of deep learning from medical signals or images. Among these, it is of great importance to detect breast cancer, a type of cancer encountered in women.

It is important to detect breast cancer because breast cancer is the most common of cancer types in women. Breast cancer is a disease that develops as a result of non -controlled growth of abnormal breast cells and forming tumors. In 2022, 2.3 million women were diagnosed in breast cancer and 670000 deaths due to the disease took place in a global diameter [1]. In general, the success rate in the treatment can be increased with the determination of early cancer types. Therefore, early detection of this type of cancer with a reliable, fast and effective method is of great importance. In addition to biopsy and clinical examination for breast cancer detection, different medical imaging techniques such as mammography and MRI stand out. The most common scanning method used today is mammography. However, the application of this method is laborious and costly, but also exposes the patient to radiation. In addition, the results of the mammography should be interpreted and diagnosed by a specialist for the precise detection of the disease.

Another method that can be used for the detection of breast cancer is thermal imaging. The idea that the body surface temperature can be used in the detection of breast cancer was first proposed by Lawson [2]. This method is highly sensitive and low-cost. The working principle of this method is as follows: Cancerous cells require a lot of oxygen-rich blood during their development. As a result, blood flow to this area increases and the temperature around the tumor rises. [3] As a result, there are differences between the thermal images of cancerous and healthy patients. The operation of systems using thermal imaging is based on this basic principle.

In the project implemented, the aim is to detect individual patients by training a deep learning system with the differences formed in thermal images. Firstly, similar studies that have been done in this area have been reviewed. As a result, an understanding has been gained about the goals to be determined in the project and the performance metrics to be measured. While there are bachelor theses on deep learning, no research has been found on the detection of breast cancer from thermal images using deep learning. The studies using deep learning from thermal images for the detection of breast cancer have been reviewed by Tsietso et al. [4]. Roslidar et al. [5] have investigated the potential for effectively detecting breast cancer using thermographic imaging and deep learning models.

Mambou et al. [6] He worked on breast cancer detection using infrared thermal images. In the study, the feature vector was removed from the image using the preliminary trained Inception V3 model.

Learning) method was classified with the Support Vector Machine-SVM). Confidence is between 0.78 and 0.96 in classification results.

Okal [7] made the feature extraction by passing the thermal images through different filtering processes and then classified the features obtained with SVM. In this study, the output was obtained as accuracy, 90 %of the accuracy, 86 %of sensitivity and 93 %of specificity.

Cabioğlu [8] has colored the front images on the Visual Lab image set using a jet color map and achieved 94.3 %accuracy by using the modified Alexnet front educated model in MATLAB.

Kirişken [9] Resnet50 has used the preliminary educated model according to their own requirements. With the replication process of Comsol Multiphysics program and Visual LAB data set, two different experimental sets were created and the highest test accuracy was 98.44 %.

Ekici and Jawzal [10] used an optimized evolutionary neural network (Convolutional Neural Network - CNN) with the Bayesian algorithm. The obtained results have a 98.95% accuracy rate.

Zuluaga-Gomez [11] used an evolutionary neural network and compared the results obtained with different optimization algorithms such as Adam, RMSprop, and SGD, as well as different hyperparameters, to select the most ideal one. The best result achieved was 92% accuracy and 98% sensitivity.

Sánchez-Cauce et al. [12] worked on a dataset that contains thermal images taken from three different angles for each patient. Classification was performed by adding the feature vectors obtained from three different images with an evolutionary neural network. The obtained results have a 97% accuracy and 83% sensitivity (sensitivity) rate.

Yousefi et al. [13] Used an evolutionary neural network to remove low -size properties. In the thermal images used here, each example consists of 20 consecutive appearances of a patient in a dynamic state. Here, 20 images were applied to a 3 -channel image by applying matrix factorization to 20 images and then the Random Forest (Random Forest) method was used. In addition, clinical data were included in the classification and the highest 78.2 %accuracy was obtained.

Mammootill et al. [14] In their study, the thermal images taken from five different angles were classified by combining the properties obtained by passing through a different evolutionary nervous network for each angle. In this study, 85.4 %accuracy was obtained, then the clinical data was included in the model and this value was obtained as 93.8 %.

Mohamed vd. [15] has automatically segmented and isolated the chest area from thermal images using the U-Net network. Subsequently, feature extraction and classification were performed using the neural network. As a result, %99.33 sensitivity, %100 specificity, and %98.67 accuracy were achieved.

Civilibal vd. [16] has surrounded the potentially tumor-forming areas in thermal images with a bounding box to improve performance. Then, he trained the pre-trained neural networks ResNet-50 and ResNet-101 for thermal images using transfer learning. The results of the tests yielded an accuracy of 97.1% and 96.2% for these networks, respectively.

Alsaedi vd. [17] has used a hybrid method based on the electromagnetic power transmission difference between healthy and diseased tissue to take thermal images by heating with a microwave radiation source, and then classified them using a convolutional neural network (CNN). A varying accuracy of 89% to 94% has been achieved in distinguishing healthy and diseased chests, depending on the training set / test set ratio.

The above -mentioned studies were performed with images taken with the protocol of Static Infrared Thermography (SIT) [4]. In this protocol, the thermal image is taken at a certain moment while the patient stands constant. In addition, Dynamic Infrared Thermography (Dynamic Infrared Thermography - DIT) [4] Protocol images are taken intermittently after the patient's chest area is cooled.

Gonzalez-Hernandez et al. [18] He examined the studies used by images taken by DIT protocol.

Ohashi and Uchida [19] examined the images taken by DIT protocol, while the images of the DIT images were applied. 82 %accuracy was obtained with the 54 %accuracy obtained for SIT.

Abdel-Nasser et al. [20] Using the "Learning-to-Shrank" machine learning method and texture analysis on DIT images, it obtained 95.8 %accuracy and 94.6 %certainty.

Silva vd. [21] have achieved 100% accuracy on the 68 images they extracted from the main dataset for DIT images using time series methods.

The work done in this project is closest to the protocol of Mammoottil vd. [14]'s published paper titled "Detection of Breast Cancer from Five-View Thermal Images Using Convolutional Neural Networks". The differences between the current project and this work are as follows:

a) In the work described in this paper, clinical data was also included in the classification. In the current project, however, clinical data is not used, and an attempt is being made to classify only based on thermal images to obtain a more general and applicable model. The use of clinical data is to create a model that

It has not been preferred due to the restrictive nature, as recording clinical data for the person to be scanned in the field also brings additional cost and time loss.

b) In this study, rectangular images were converted into squares for use, while in the current study, the images were used in their original rectangular form. In this way, the potential data loss that could occur in the images is attempted to be prevented.

c) Neural networks with standard rotation were used to obtain the feature vectors to be combined in this study. In the current study, feature vectors are obtained using autoencoders. In order for rotation neural networks to be trained successfully, it is necessary to have as large a dataset as possible. Since the size of the dataset to be used for training is limited, the use of autoencoders is preferred. Autoencoders are more successful in this situation. Especially for dimensionality reduction, autoencoders work more successfully than the pca method.

d) Due to the use of clinical data in this study, examples with conflicting information about the patient's age were extracted. However, since this data was not used in the current study, these examples could also be included in the study.

and In this study, data series is divided into 80 %training and 20 %test. In the current study, the data was separated as 70 %training and 30 %test and the test set was kept wider. Thus, the operation of the last emerged model can be tested more accurately.

f) In this study, only 3 different aspects of images were combined. In the current study, mergers were made for both 3 and 5 images.

In order to analyze the DIT protocol images in the current project, a patient's sequential DIT images were first converted into a video with a method similar to the classification approach in the videos, and then the disease was detected with evolution nervous network and LSTM layer. In the studies of DIT on breast cancer, such an approach has not been tried before.

After the literature research, engineering constraints were determined in the project. These are economy, cost analysis, health and productivity.

1.1. Economy: Device methods such as MRI and tomography used for breast cancer detection are costly. The recommended detection system stands out economically with its low cost. Since the investment cost of a system to be established will be low, the depreciation share will be low. In addition, the system will not use a consumable material (eg the film in mammography) will consume only low amounts of electrical energy.

Sysytem's environmental recycling rate will be higher due to the absence of radioactive sources like mammography equipment and complex hardware like MRI equipment. The use of consumables also positively impacts environmental recycling.

The most significant cost for the project in terms of cost and feasibility is the hardware used for training the neural network system. To reduce costs, the use of cloud computing options is being considered instead of purchasing hardware.

Since the operation of a system in use does not require a high level of expertise for personnel, personnel costs will therefore also decrease. Moreover, since the system is software-oriented, an update will be sufficient for renewal. (For example, a system trained with a new dataset)

Model development has used TPUs instead of GPU hardware for training the model, thereby shortening the time spent on development and reducing costs. Due to the reduced power consumption of the cloud computing system, a positive environmental impact has also been achieved.

2. Health: Since the project will be developed in a software environment, there is no concern about containing a component that could affect human health. The method of breast cancer detection to be developed in the project does not have a negative impact on human health in general application. For example, the mammography method exposes the patient to radiation. In the biopsy method, the patient requires surgical intervention. Since only a thermal camera was used during the preparation of the dataset, the patients were not exposed to radiation. The same situation would apply if a different dataset were to be created using the same method later. Similarly, an trained system would use a thermal camera for detection, so there would be no negative impact on human health.

Besides the system to be implemented, it will also contribute positively to human health by increasing the success rate of treatment due to early diagnosis.

3. Feasibility: The detection method under consideration has also been taken into account in terms of feasibility. A standard thermal camera set is sufficient to obtain images. This camera can also be easily carried in the field application as a mobile unit. The software and hardware needed for the training and field operation of the system (such as laptops, mice, and thermal cameras) are standard products available on the market and used in various fields. A laptop with internet infrastructure that can connect to the cloud computing system has been found suitable for training. This way, the trained system can also be mobile while providing services in the field. The use of cloud computing will reduce the capacity requirements of the equipment to be used during both training and the application phase, making it both more economical and more portable. In addition, in the event that multiple mobile systems are created for the field, the use of cloud computing will allow multiple systems to use the same cloud computing system, making the system more feasible.

4.4. Cost Analysis:

The cost analysis of the project is included in Table 1.

Table 1. Cost Analysis

Requirements	Features	Cost
Computer (Laptop)	RAM Memory: 16 GB Operating System: Windows 10 CPU: Intel Core i5 Graphic processor: shared	23499 TL
Mouse		2599 TL
Google Colab	1 Pro+ License * 9 months * (826.80 TL)	7441.20 TL
Engineer Salaries	1 Engineer * 9 months * (32500 TL)	292500 TL
Deep Learning with Python (Book)	Pressed	1770,92 TL
Total Cost		327,810.12 TL

The goal of the project is to emerge as a successful and reliable method that can be compared to classical methods in cancer detection through deep learning. In this way, the detection method with thermal imaging will be an alternative that can replace other methods. This is one of the most important goals of the project. For this, specific performance parameters should be provided on the outputs. Additionally, to create a reference source for future work in image processing with deep learning, which has developed in recent years.

Some standard performance metrics used in the evaluation of deep learning systems are shown in Table 2. Positive results correspond to patient, negative to healthy status. To explain these metrics, some concrete measurable concepts must be defined first. True Positive (TP) indicates that the actual result is positive and the system has classified it as positive, True Negative (TN) indicates that the actual result is negative and the system has classified it as negative, False Positive (FP) indicates that the actual result is negative and the system has classified it as positive, and False Negative (FN) indicates that the actual result is positive and the system has classified it as negative, showing the number of examples in these cases.

Table 2 . Performance Metrics

Metrics	Formula	Target Value
Precision (Precision)	$\frac{TP}{TP+FP}$	%93 \pm %2
Sensitivity (Recall)	$\frac{TP}{TP+FN}$	85 \pm 3
Accuracy (Accuracy)	$\frac{TP+TN}{TP+TN+FP+FN}$	95 \pm 3
F1 Score	$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$	0.92 \pm 0.04

The project method is given as a flowchart in Figure 1. The theoretical foundations of the techniques used in the following sections are provided and their implementation is explained, and finally, the results are presented.

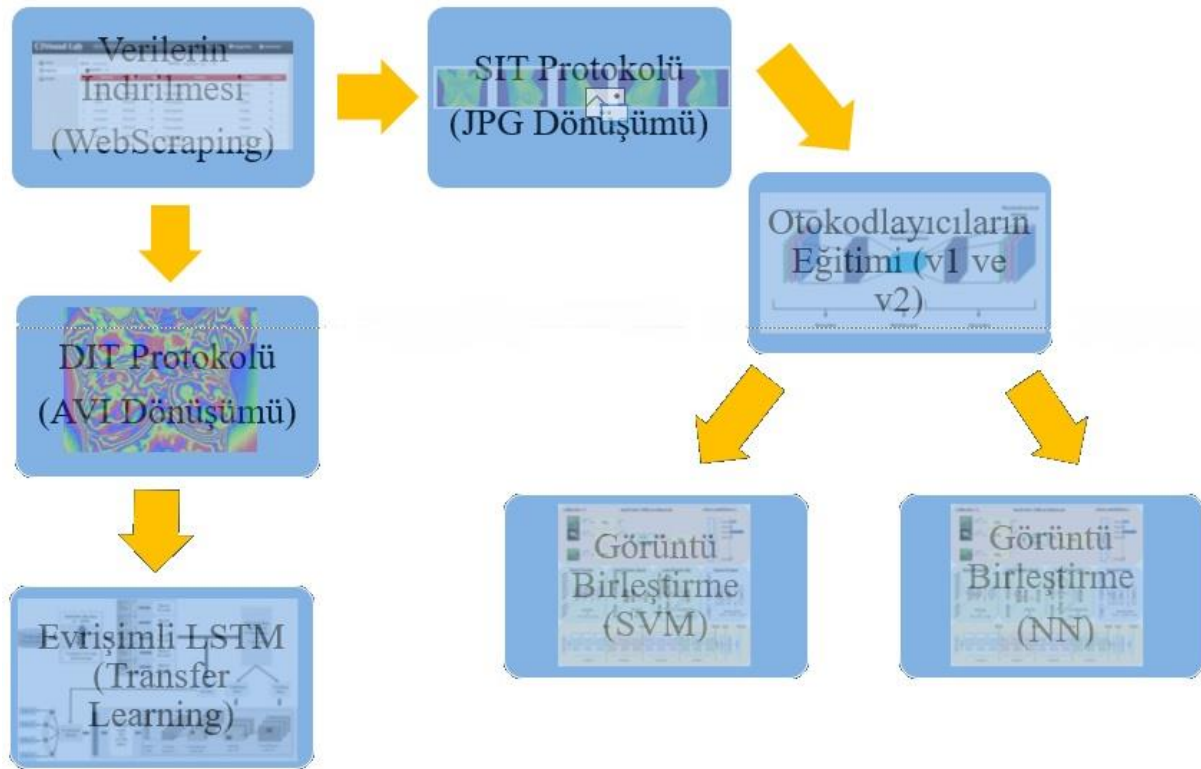


Figure 1. Method flowchart

2) Development:

Theory:

Project theoretical foundation relies on artificial intelligence. Artificial intelligence, briefly defined, is the attempt to automate intellectual tasks normally performed by humans [22]. The concept of artificial intelligence emerged in the 1950s, but it has only been widely used in recent times. Today, the most widely used sub-field of artificial intelligence is machine learning. Deep learning, on the other hand, is within the scope of machine learning but specifically involves the use of artificial neural networks. (Figure 1)

Normally, a computer can perform a specific task only if programmed with an algorithm or software by a human. Machine learning systems, however, are trained rather than explicitly programmed [22]. The most important requirement for the training of these systems is data.

Machine learning training processes are divided into supervised and unsupervised categories. If labeled data is available, the system is trained using supervised learning (supervised learning). A label of an input is defined as the expected output from the system for that input. For example, if the image of a patient is labeled as 1, the image of a healthy individual is labeled as 0. The system used in the current project has been trained using supervised learning.

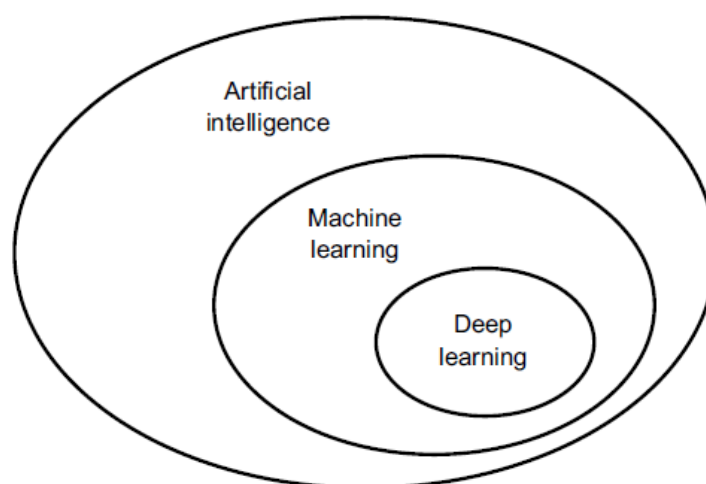


Figure 2. Artificial intelligence, machine learning, and deep learning hierarchy [22]

The basic working principle of a system that operates with machine learning is shown in Figure 2. During the training phase, the system divides the space according to labeled data to identify different regions. After this phase,

when the trained system needs to make a decision, it determines the label value based on the coordinates of the sample point according to which region it is in.

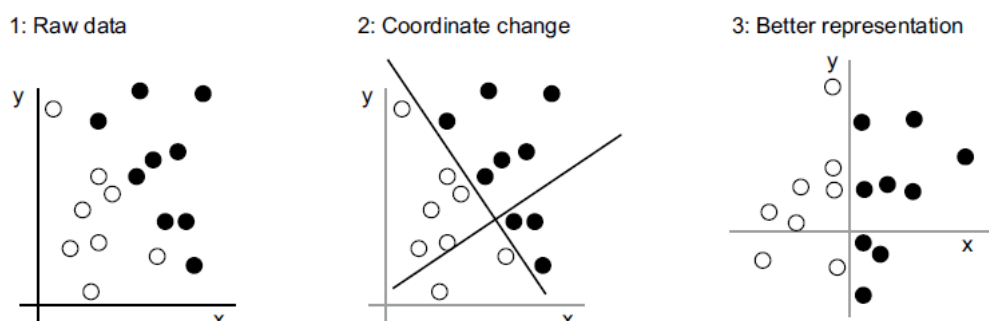


Figure 3. Training the machine learning system [22]

Deep learning is a subfield of machine learning but has a different processing and is related to artificial neural networks. Artificial neural networks consist of interconnected layers of artificial neurons. The schema of an artificial neuron is shown in Figure 3.

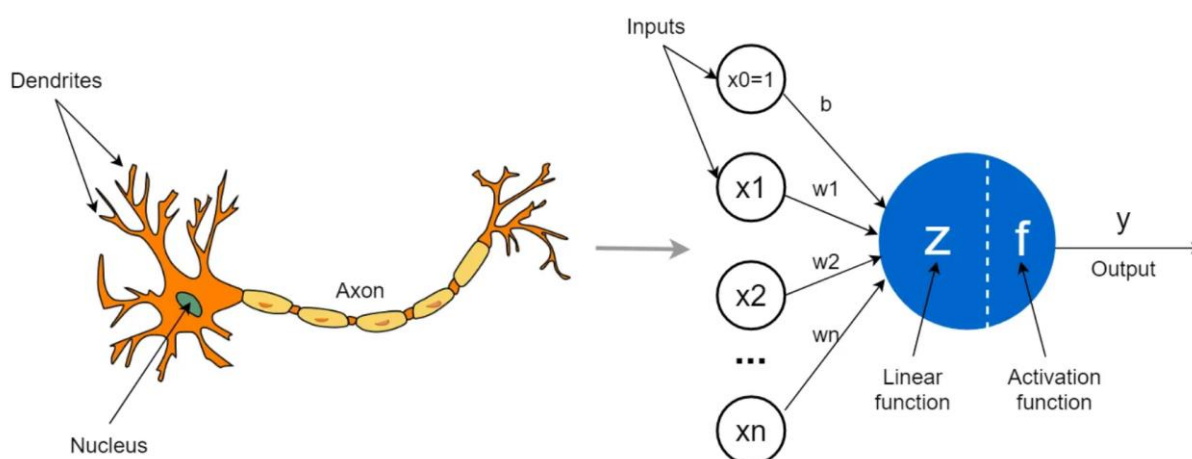


Figure 4. Structure and connections of an artificial neuron [23]

Each neuron in an artificial neural network can be considered as a linear regression model within itself, consisting of input data, weights (weight), bias, and an output [24]. Considering the input and output shown in Figure 3, the equation of this model is

$$z(x) = \sum_{k=0}^{\infty} \frac{1}{k!} \frac{d^k f(x)}{dx^k} + \frac{1}{2} \frac{d^2 f(x)}{dx^2} + \dots \quad (3.1)$$

It is assumed. The input layer, hidden layer, and output layer of the artificial neural network are shown in Figure 4. The hidden layer may have multiple numbers depending on the complexity of the neural network. Since there is a depth in the formed sequential layers, the concept of deep learning (deep learning) comes from here. A dense (dense) neuron in a layer is connected to all artificial neurons in the adjacent layers.

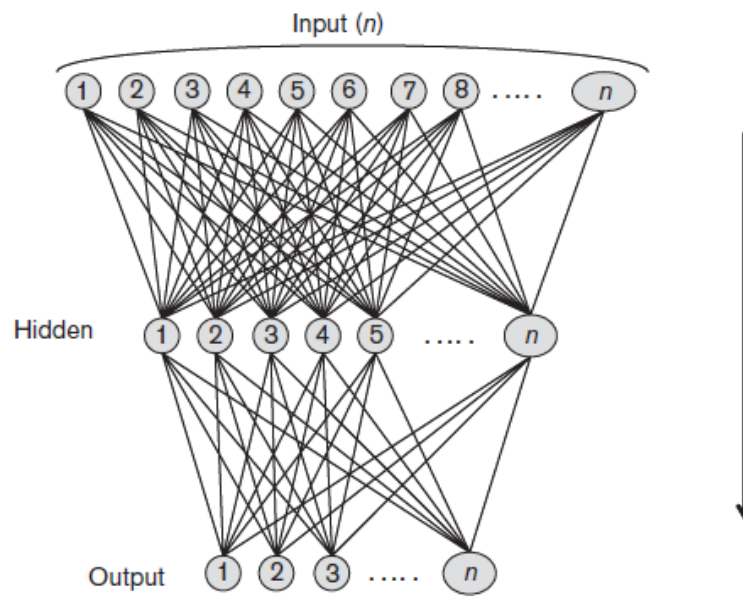


Figure 5. Artificial neural network and layers [25]

In the initial state, the weight values of all neurons in the system are given arbitrarily. The output is calculated by giving the result of the activation function calculated according to Equation 3.1 to the input and bias value of a neuron, and this output is given as input to the neuron in the next layer. This process is called propagation (propagation). A loss function dependent on the weights is calculated by comparing the output value obtained in the last output layer with the expected output value (label). The goal in the training phase is to minimize this loss function. Each weight value is updated in the direction of minimizing the loss function by using the chain rule and partial derivative properties. This process is called backpropagation (backpropagation). (Figure 5)

The data to be processed is used when there is an image instead of a vector. Convolutional Neural Network is used. Figure 6 shows the structure of a evolution nervous network.

Convolution is shown in Figure 7. In this process, the classic image process called "kernel" is used as filters. In deep learning, tensors are used as data structure, especially in evolution systems. (Figure 8) The concept of tensor is a more general concept that includes matrix and vectors. For example, a colorful image in the RGB format is a 3 -dimensional tensor structure. In this case, the kernel is a 3 -dimensional receipt. A set consisting of RGB images is a 4 -dimensional receipt.

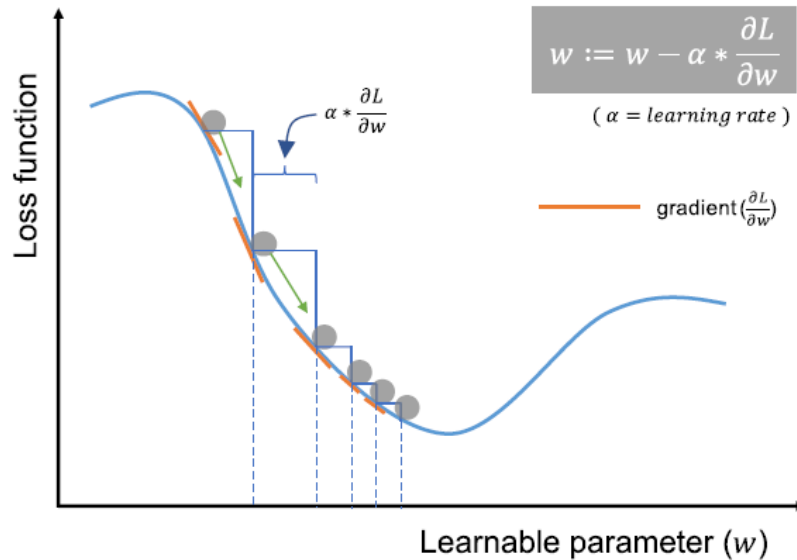


Figure 6. Lost function and learning process [26]

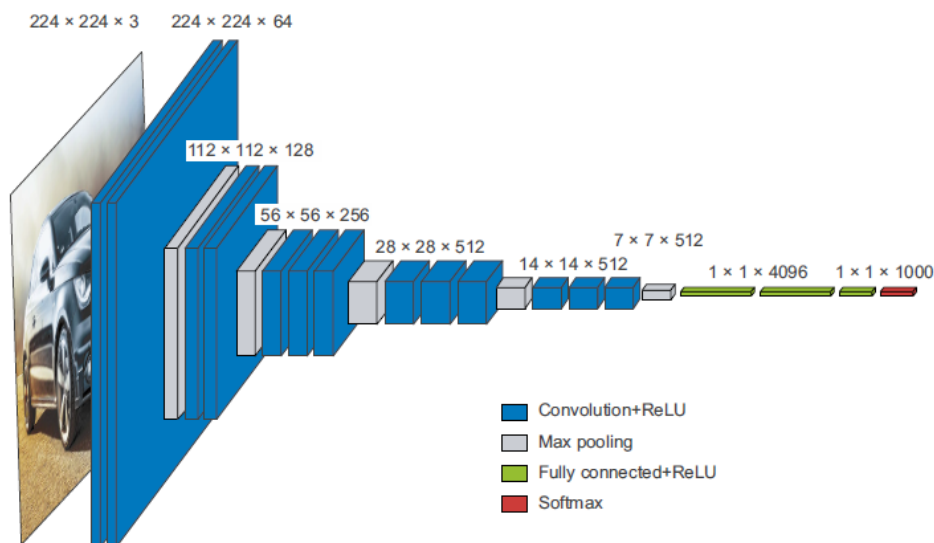


Figure 7.

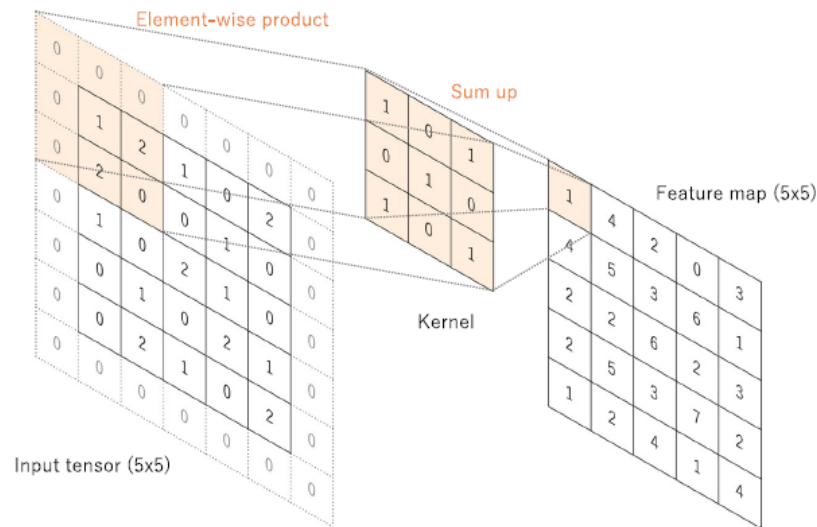


Figure 8. Convolution operation [26]

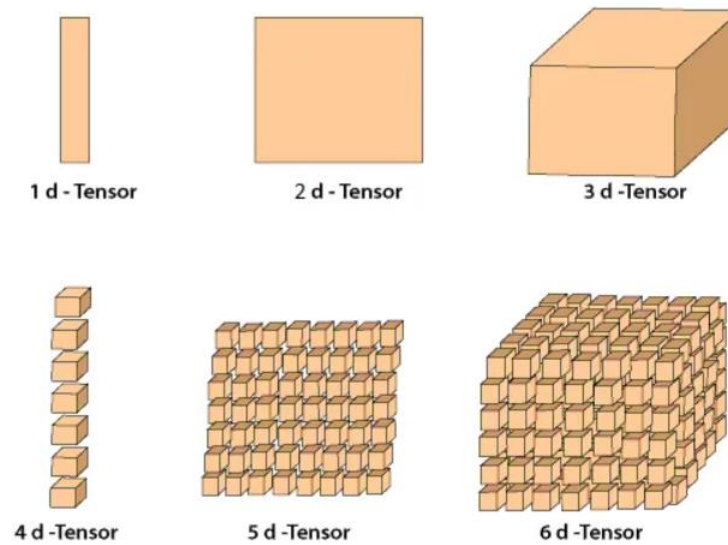


Figure 9. Tensor dimensions [27]

The value of each pixel in the kernel corresponds to a weight and is updated during training. The center pixel of the kernel is overlaid so that it aligns with a specific pixel on the image and element-wise matrix multiplication is performed, the resulting value being the value of the corresponding pixel in the output image for the next layer. The kernel is shifted by the step (stride) and the operation is repeated. In this way, a new image is obtained from the original image.

In the SIT section of the project, autokoders have been preferred, as they can be successfully trained even with small-sized training sets. (Figure 9) In autokoders, the input and output are the same and they are unsupervised.

They are trained with learning. They consist of coding and decoding parts. Feature vectors are obtained from the output (bottleneck) of the coding part after the system is trained.

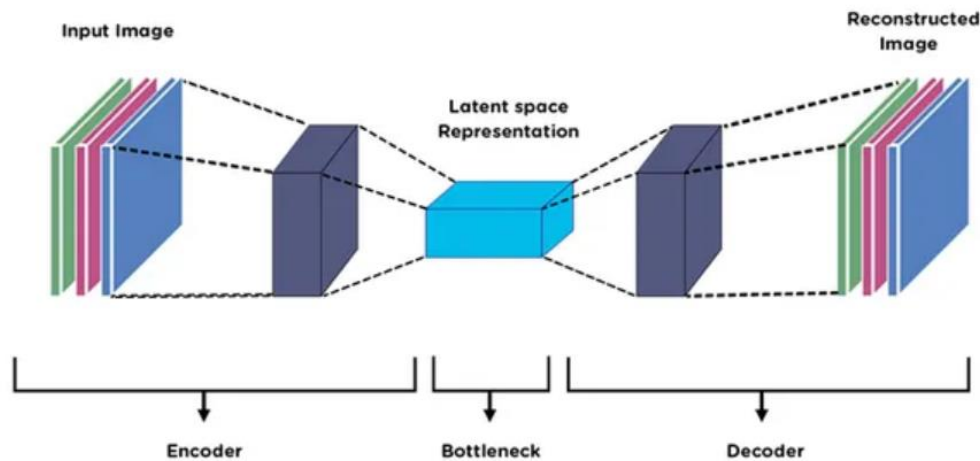


Figure 10. Structure of the Convolutional Autoencoder [28]

In the SIT protocol section of the project, the end-to-end concatenation method, as followed in the work of Mammoottil et al. [14], has been used to fuse the feature vectors of different images. A diagram explaining the fusion process of the transformed outputs is given in Figure 10.

LSTM layer was used to examine the images of DIT protocol. Recurrent Neural Network is used for time series data such as DIT images. In ordinary neural networks, while flowing in one direction in one direction, the output of a nervous network in the structure of RNN can affect the input with a feedback mechanism. LSTM is a more advanced version of the rnn. Specifically, LSTM with evolution was used as the image data was discussed. A LSTM cell figure Shown in 11. Figure 12 gives an example of a system that works with LSTM with evolution.

The deep learning system used in the project was created with Python software code and tensorflow library and trained using TPU processing units in Google Colab cloud computing environment. The codes written are given in the annexes. TPU provides a faster processing advantage than GPU, but there is a need to make some special plug -ins and arrangements in the standard code.

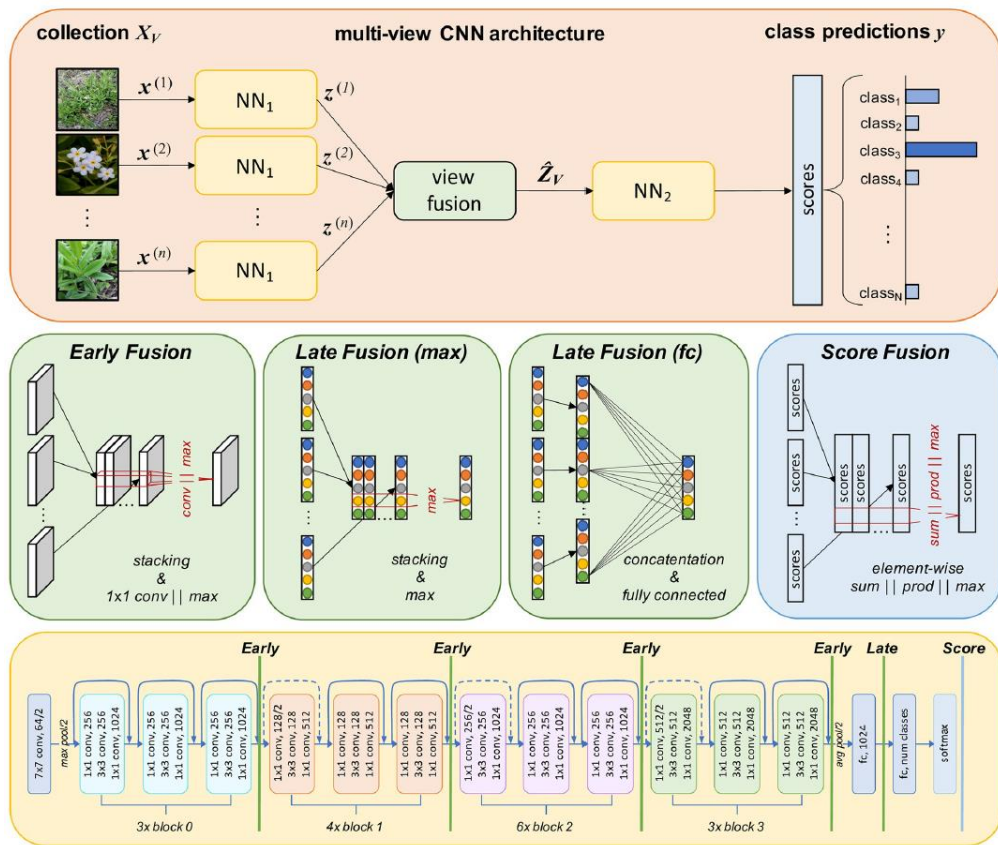


Figure 11. Fusion Combination process [29]

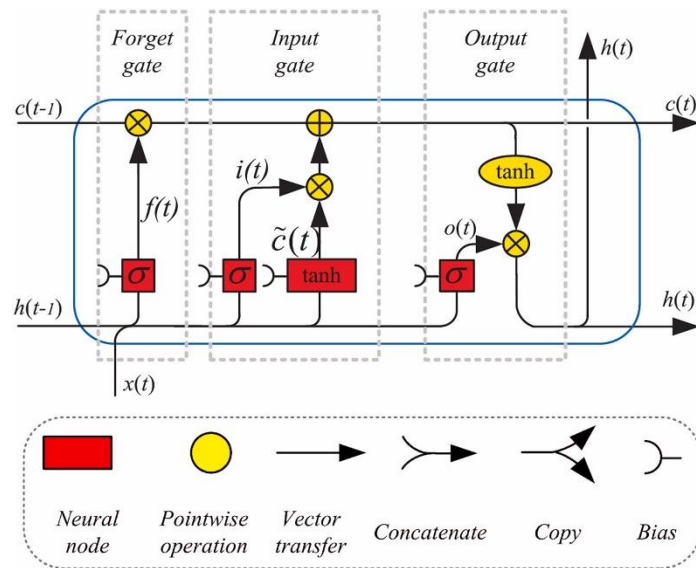


Figure 12. LSTM cell [30]

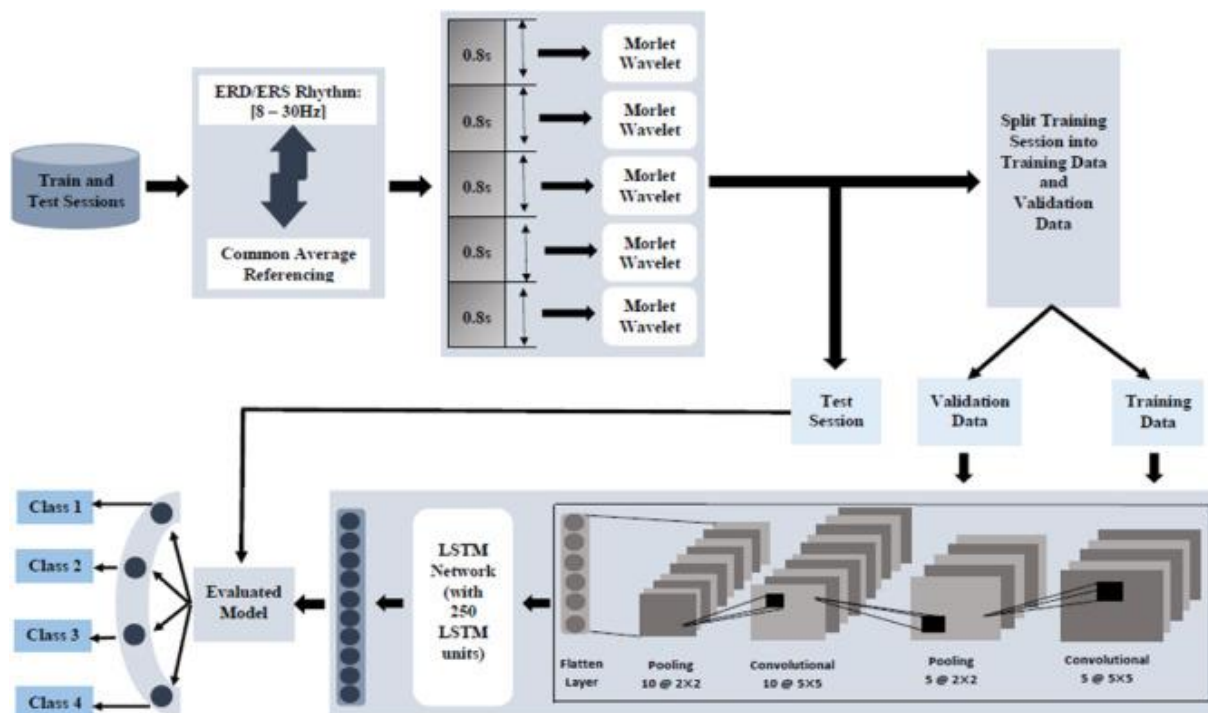


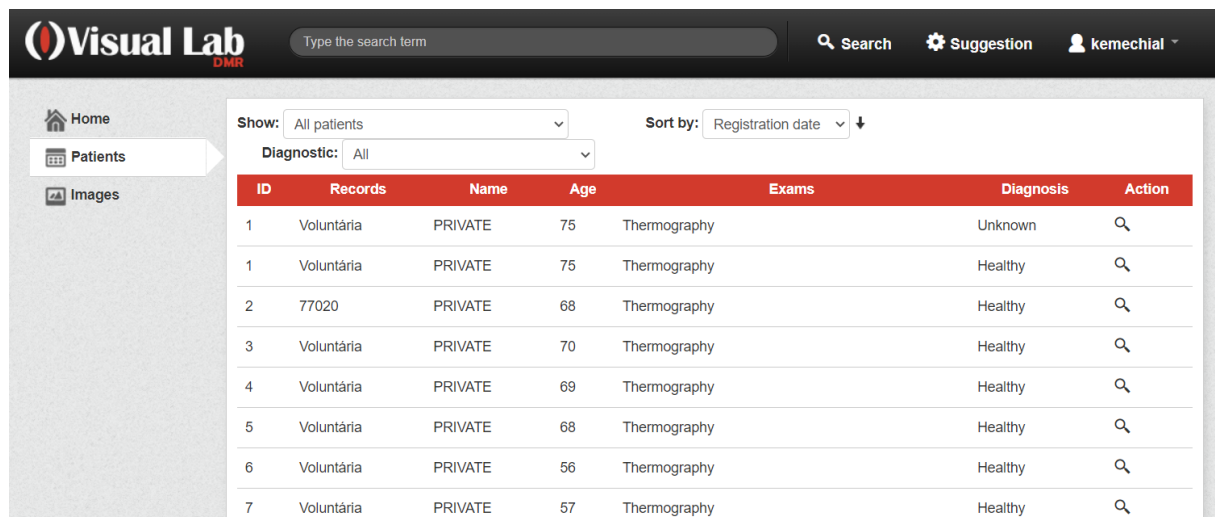
Figure 13. Evolutionary LSTM [30]

Experiment:

The procedures made according to the flow chart (Figure 1) are explained in this section. At the beginning of the project, Silva et al. The data set presented by [31] was obtained. The relevant data set has a website with a user interface, but the data cannot be downloaded as a single Rar file compressed. For each patient, the relevant images are on a separate page according to the patient number, the front appearance of the images can be monitored and the links of thermal matrix txt files are given.

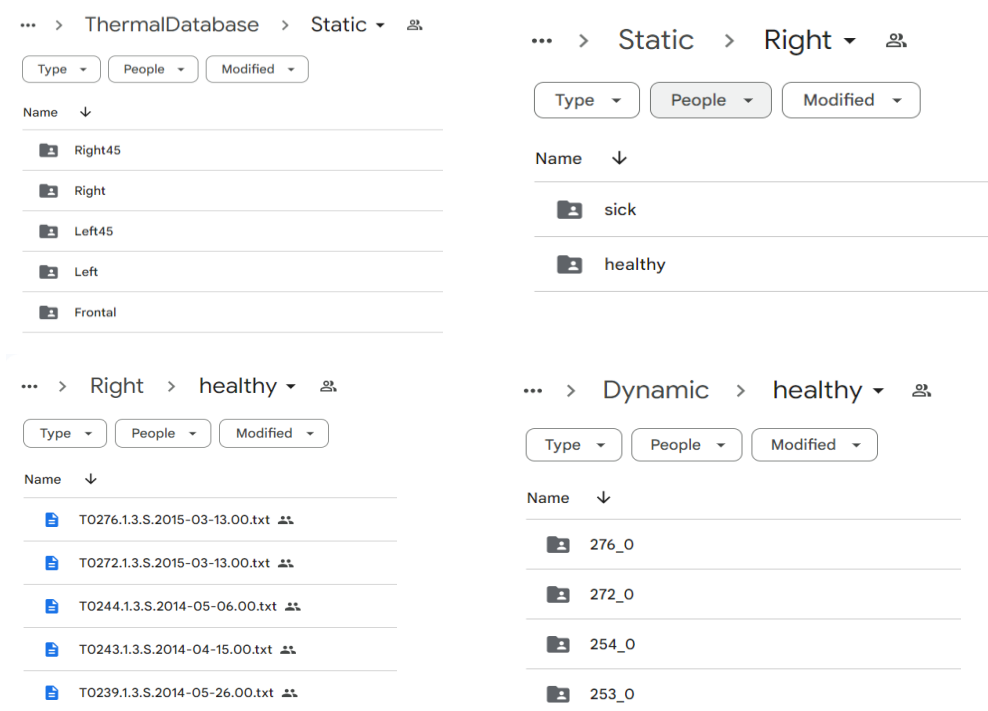
(Figure 13) Web scraping technique with the Txt file links on each page with the technique of the files on these links; The SIT and DIT protocol has been reduced to Google Drive in the form of a hierarchical file system according to patient and healthy situations. (Figure 14) The code that makes this process is given in Annex-1. Being hierarchical, the data is important for the correct creation of the dataset in the tensorflow environment in the next step.

After the data was collected, the thermal matrices, which were initially in txt file form, were converted to jpeg format. The "viridis" color map, which is preferred by Civilibal et al. [16] and provides better results, was used during the conversion process. Examples of the images converted to jpeg format are given in Figure 15. The images created were saved with the same hierarchical structure. The code for performing these operations is provided in Appendix 2.



ID	Records	Name	Age	Exams	Diagnosis	Action
1	Voluntària	PRIVATE	75	Thermography	Unknown	
1	Voluntària	PRIVATE	75	Thermography	Healthy	
2	77020	PRIVATE	68	Thermography	Healthy	
3	Voluntària	PRIVATE	70	Thermography	Healthy	
4	Voluntària	PRIVATE	69	Thermography	Healthy	
5	Voluntària	PRIVATE	68	Thermography	Healthy	
6	Voluntària	PRIVATE	56	Thermography	Healthy	
7	Voluntària	PRIVATE	57	Thermography	Healthy	

Figure 14. Data set user interface



Path	Files
ThermalDatabase > Static	Right45, Right, Left45, Left, Frontal
Static > Right	sick, healthy
Right > healthy	T0276.1.3.S.2015-03-13.00.txt, T0272.1.3.S.2015-03-13.00.txt, T0244.1.3.S.2014-05-06.00.txt, T0243.1.3.S.2014-04-15.00.txt, T0239.1.3.S.2014-05-26.00.txt
Dynamic > healthy	276_0, 272_0, 254_0, 253_0

Figure 15. The hierarchical folder structure of the data set in the Google Drive environment

In the description of the dataset, it is stated that the images of each patient were taken from five different angles, but some links to txt files on the web page are empty, resulting in some patients having fewer than 5 images. When the dataset is to be obtained from the images that are hierarchically separated by type, it was observed that there is not the same number of files in each folder. (Figure 16) This lack of data situation is SIT

When reviewing the protocol, the autoencoder and image merging process can be considered differently in terms of training procedures.

The autoencoder training and image merging training are independent of each other: In the autoencoder training, only the accuracy with which the image can be reconstructed is examined using the test data. The test result confirms that the autoencoder does not memorize. In this case, the feature vector formed from the coding part can represent the image in the best way. No labels are used in this unsupervised learning type of training. In the supervised learning type of image merging training, after the feature vectors obtained from the autoencoder are combined, the accuracy of predicting the labels is tested. Therefore, different training sets can be used for the two trainings.

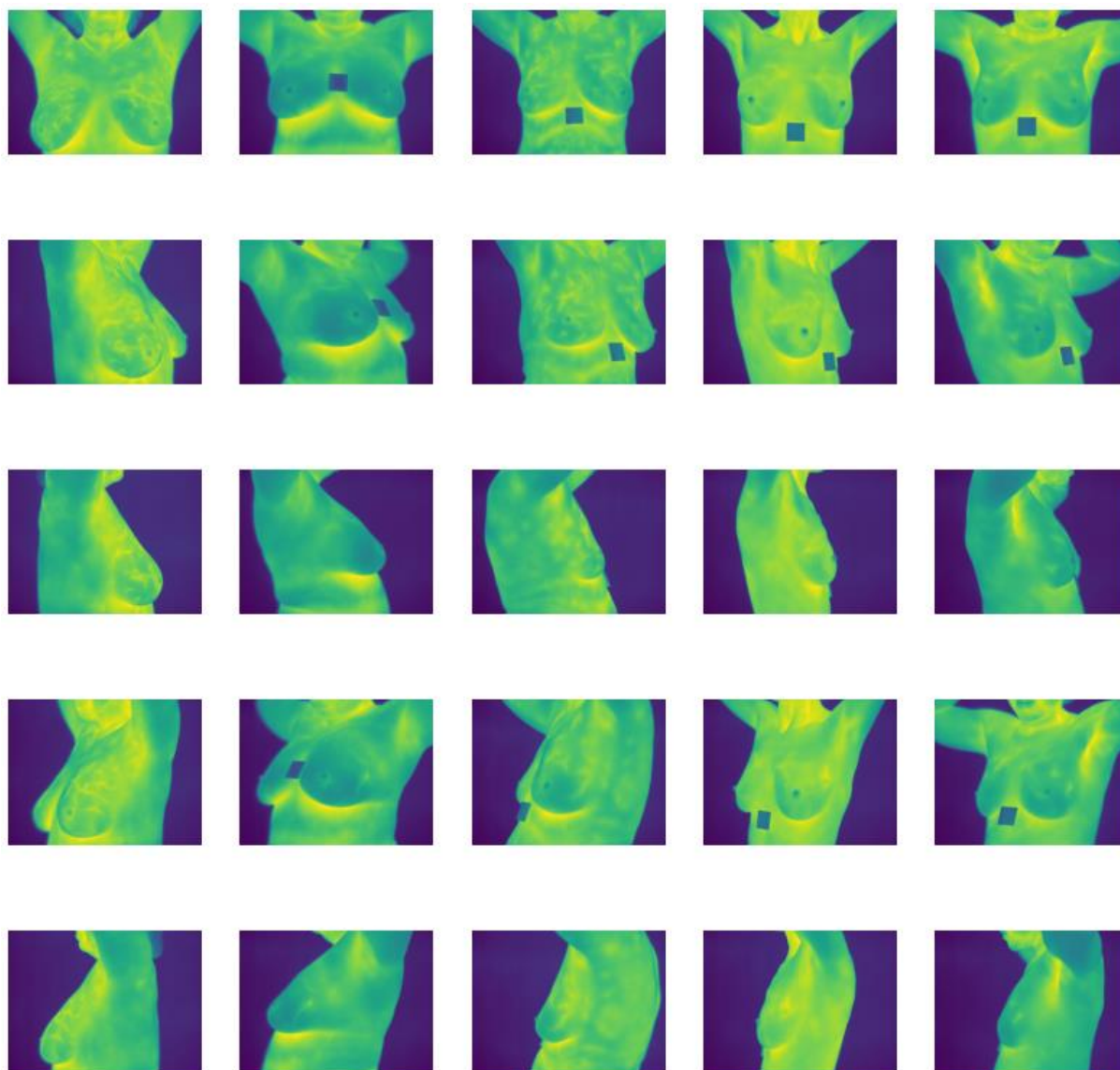


Figure 16. Views from 5 different angles of 5 different patients


```

✓ 1m print(class_names)
train_ds.save(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/train")
val_ds.save(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/val")

```

```

Frontal
Found 300 files belonging to 2 classes.
Using 210 files for training.
Using 90 files for validation.
['healthy', 'sick']
Right45
Found 290 files belonging to 2 classes.
Using 203 files for training.
Using 87 files for validation.
['healthy', 'sick']
Right
Found 296 files belonging to 2 classes.
Using 208 files for training.
Using 88 files for validation.
['healthy', 'sick']
Left45
Found 292 files belonging to 2 classes.
Using 205 files for training.
Using 87 files for validation.
['healthy', 'sick']
Left
Found 294 files belonging to 2 classes.
Using 206 files for training.
Using 88 files for validation.
['healthy', 'sick']

```

Figure 17. Differences in the number of examples between image modes due to dataset incompleteness

The coding parts of the autocoders trained in the previous stage and tested autocoder are used in the previous stage and patient samples with only 5 image mode can be used. However, there is no such obligation because the auto encoders of different modes are trained separately. Thus, as many examples can be used for auto encoder training. The general code that constitutes a training set from the data available is given in Annex-3. All data were used to create a training set for the auto encoder. In the image joining process, a separate code that makes filtering process was written and the patients who were missing images were filtered and only filtered data were used when creating a training set. (ANNEX-4) Another condition that should be considered when creating training sets filtered for image joining process is that the seed (Seed) parameter is always given the same and 1 for all image modes. Thus, the rankings of the images of different angles in the training sets are the same, and the images of the same patient are combined during the image merge.

Two different types of autocoders were used in the study. The first type used (V1) auto encoder scheme is shown in Figure 17. In the other type (V2), unlike the first, Maxpooling and Upsampling layers were replaced with Conv2D and Conv2DTranspose layers respectively and entered as step parameter (stride) 2, and size reduction and enlargement operations were performed in this way. (Figure 18) The aim of the Maxpooling process is to reduce the number of parameters, but the positional information of the features disappears when taking the features of the image with this layer. These are protected when dimensional reduction is performed with the step parameter [22]. Auto encoder codes are also given in V1 and V2 Annex-5 and Annex-6. For example

Auto encoder codes of the image are given, although the codes of other auto encoders are the same, only folder names change.

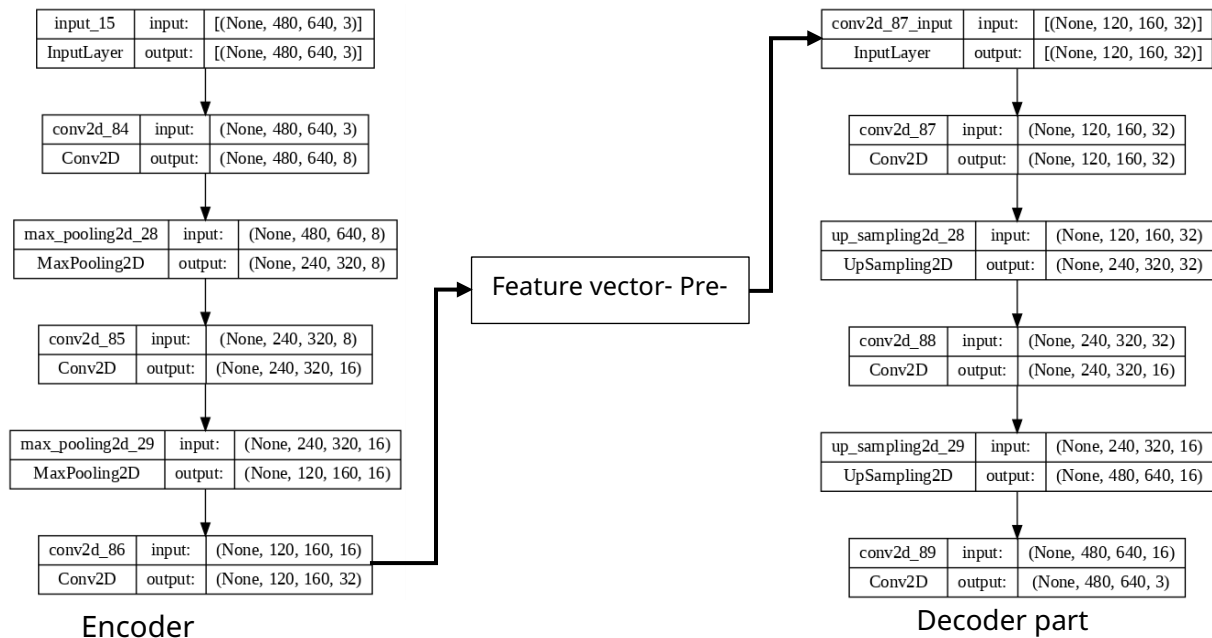


Figure 18. Autoencoder used for the front view (v1)

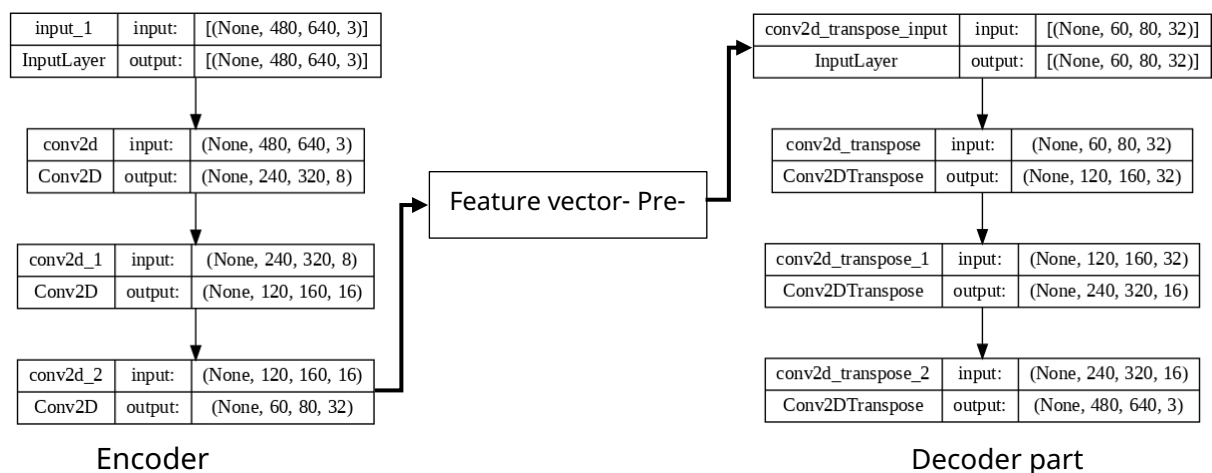


Figure 19. Autoencoder used for the front view (v2)

Normally, in an evolution nervous network, the system is compared with the tag vector (y_{train}) by giving the training tensor input (x_{train}) input. Therefore, data on a standard tensorflow data set, examples

and packed in the form of labels. However, the tensorflow data set cannot be given directly to auto encoders used in the project. Because the output obtained in auto encoder training tries to be likened to the input. In other words, the result is compared with the training tensor when giving the training tensor input. Therefore, by opening the standard data set created, a Python code that separates the training tensor should be written. This code is given in Annex-7.

For each image mode, the vectors obtained from separate auto coders were added to the end of the vector and a single feature vector was obtained and this feature vector was given to the support vector machine (SVM) and final classification results were obtained. (Figure 19) The code that makes the image joining process is given in Annex-8.

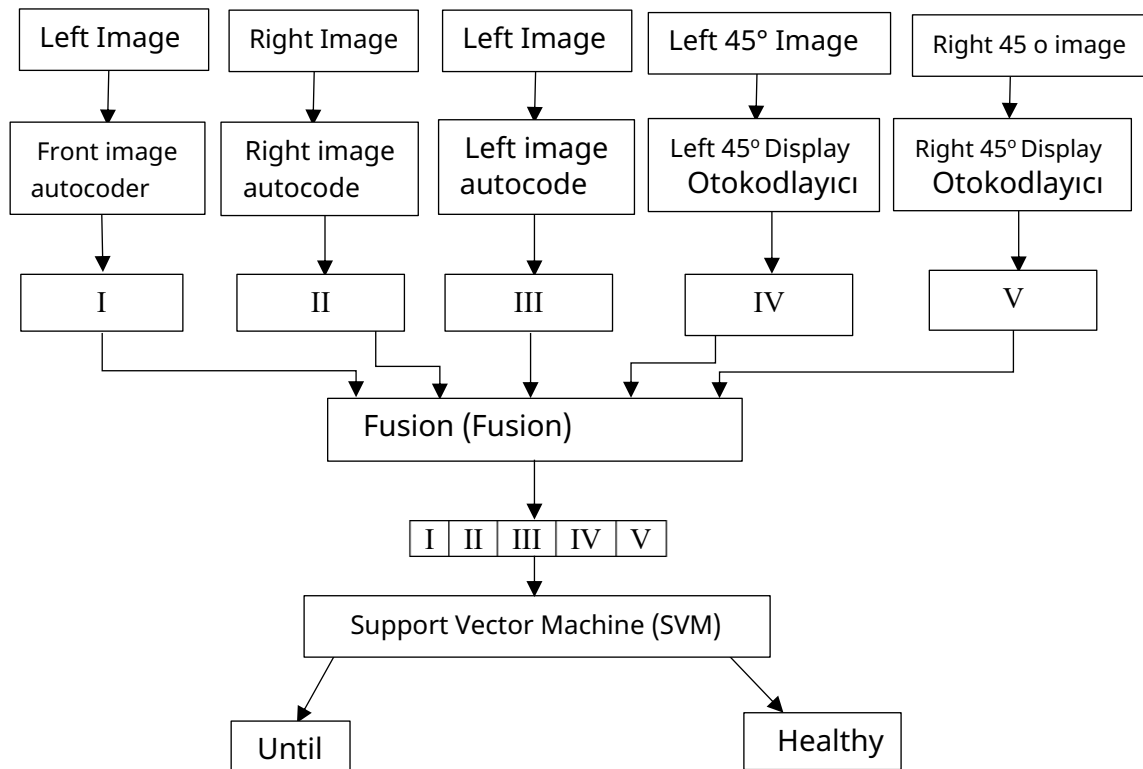


Figure 20. Image Fusion Model Flow Diagram (SVM)

In addition to the use of SVM for image joining, a separate model was examined by using an evolution nervous network and the status of image merge and classification. The scheme of the relevant model is shown in Figure 20. The relevant code is given in Annex-9. Here, for the joining of the image, the feature vectors from the coding parts of the autocoder were inserted into separate artificial neural networks, and then the resulting vectors were added to the end of the end and the results were inserted into the last nervous network that made the classification process.

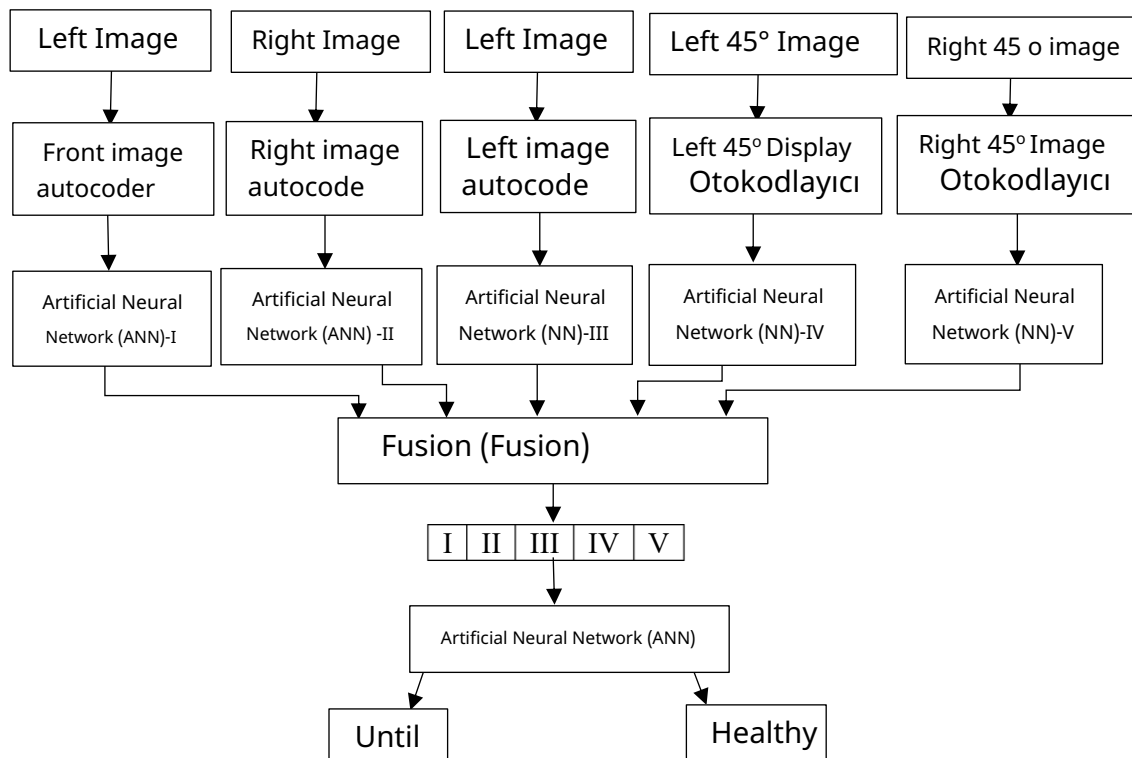


Figure 21. Deep Learning Model Flow Diagram (Artificial Neural Network)

SIT protocol was reviewed after that, and the DIT protocol was taken into consideration. In the DIT protocol, it was also noticed that the images of some patients were missing, so only patients with 20 complete images were examined. While reviewing the DIT protocol, an approach used in the video classification application by Ferlet [32] was followed. Twenty images taken for each patient with the DIT protocol were saved in separate folders in the Google Drive environment. The images of the DIT protocol were initially saved as "viridis" like in the SIT protocol, but different color maps were tried during the testing phase and the "prism" color map was preferred. (Figure 21) In this color map, even if the numerical value of the pixel changes very little, there are large color changes, so a more pronounced movement occurs in the video and the recognition becomes easier. The images in the folder have been converted into a separate avi video file for each patient and saved in the file system.

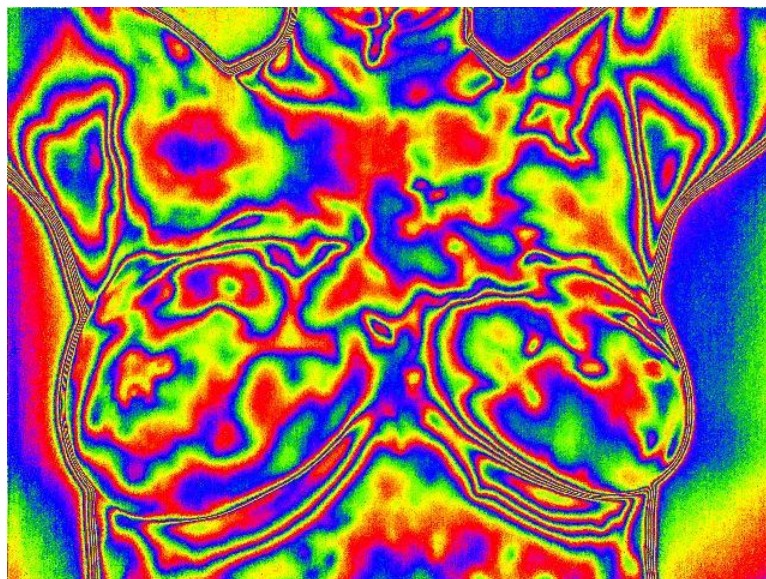


Figure 22. DIT protocol image (prism color map)

In video applications with TensorFlow, each sample is a multi-dimensional tensor consisting of many frames, therefore, instead of the standard TensorFlow dataset, a Frame Generator structure is used. Thus, instead of extracting training samples from a ready-made set, they are created on the fly and provided to the system. Frame generator function can be customized as needed. Generally, in video classification, instead of all frames, frames selected at intervals from the video are used. In this work, 10 frames were selected from a video consisting of 20 frames at intervals using the frame generator. The diagram of the evolvable LSTM model used in classification with the DIT protocol is shown in Figure 22. The code is provided in Appendix EK-10.

When the LSTM layer is implemented, the Time Distributed layer is needed. This layer takes an evolvable neural network model as input and sequentially passes 10 frames from an example through this network to produce an output ready for LSTM use. The operation of the Time Distributed layer is shown in Figure 23. Transfer learning has been used to extract image features. Thus, the system focuses only on the training of the LSTM part. The EfficientNetB5 model has been chosen as the evolvable neural network. EfficientNetB5 only works with the pre-trained image size (456,456,3). The resizing of the frames is done within the FrameGenerator function. Some additional layers have been added to the EfficientNetB5 model for customization. (Figure 24) Here, the output is converted into a vector of image shape. For this purpose, the GlobalAveragePooling2D layer has been chosen. Additionally, a BatchNormalization layer has been added.

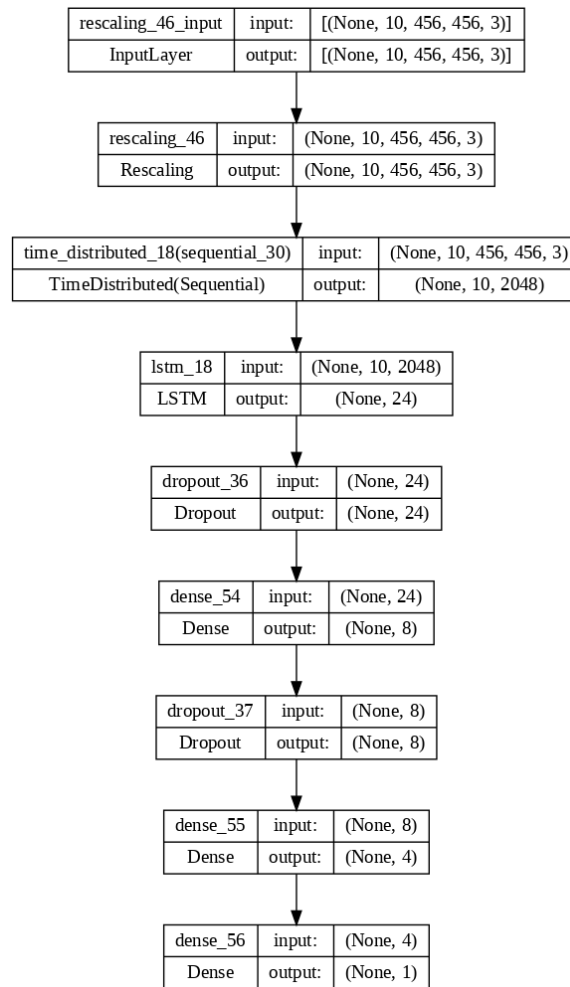


Figure 23. DIT protocol evolving LSTM model

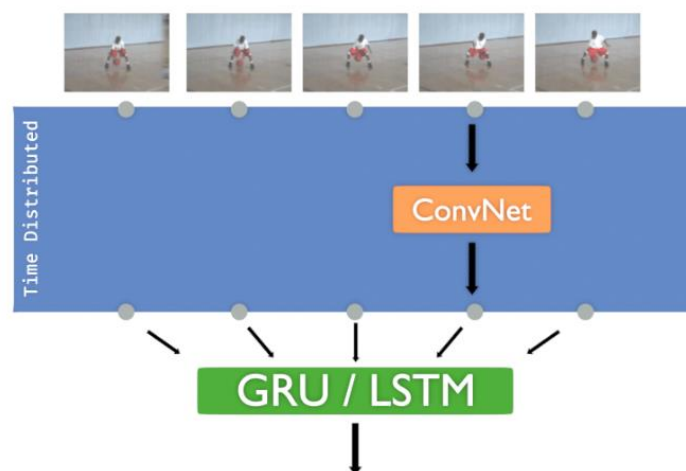


Figure 24. Work of Time Distributed layer [32]

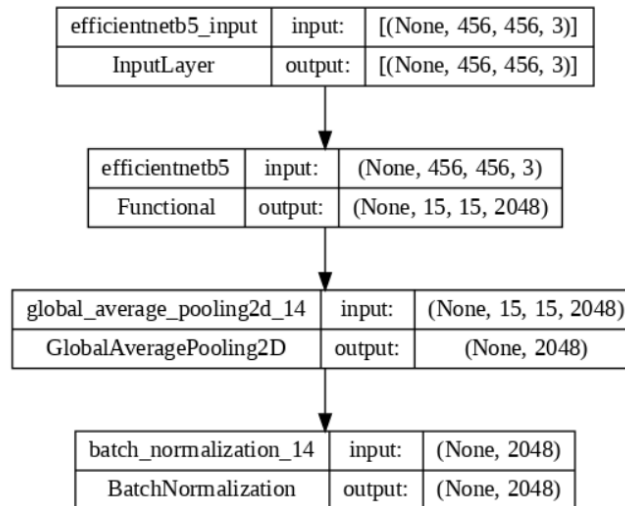


Figure 25. Layers added on Efficientnetb5

Measurements:

After the models to be used were created, the data sets created were given to the systems and the results were examined. The classification results are usually given as confusion matrix graphics. In this matrix, TN, FP, FN and TP values defined in the input section are shown. (Figure 25)

TN	FP
FN	TP

Figure 26. Confusion Matrix

Firstly, autocode learners used in the image fusion model have been trained, and their performance during training has been examined. For example, the precision graph of the preliminary image autocode learner is shown in Figure 26.

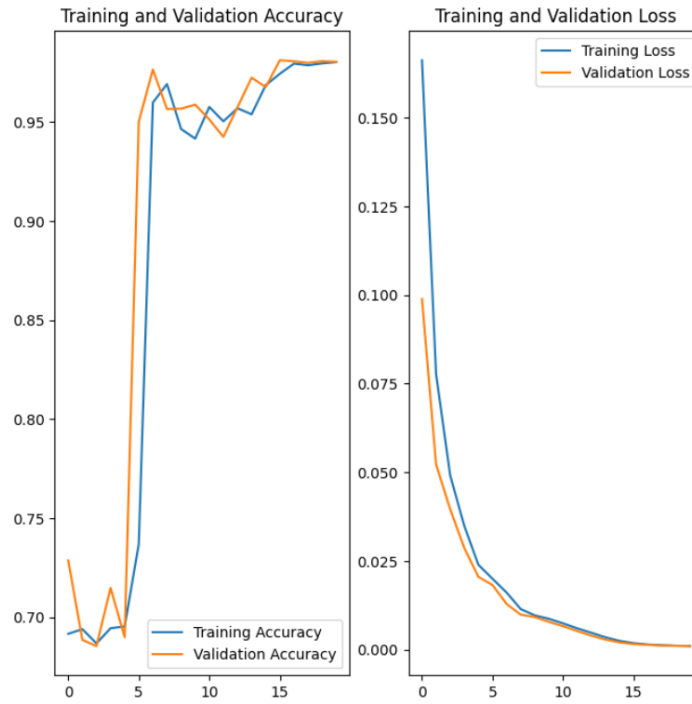


Figure 27. Otokodlayıcı Verification Graph (Preview Image)

Before transitioning to classification with image fusion, the disease recognition status was also examined using only the preliminary image. This separate examination status was carried out using the SVM machine learning method within the preliminary image auto-coding codes (v1 and v2) provided in EK-5 and EK-6. (Figure 27)

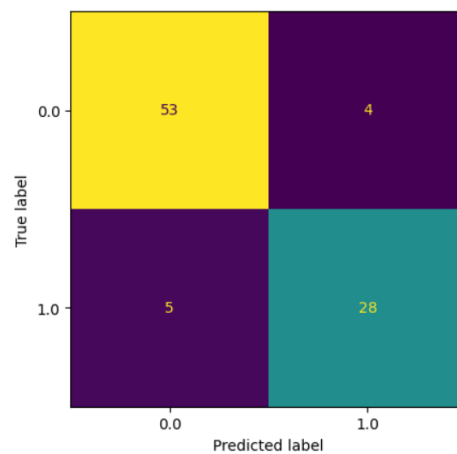


Figure 28. Classification using only the preliminary image (Otokodlayıcı v1)

In image fusion, while the number of FP (False Positives) decreases according to the classification using only the preliminary image, the number of FN (False Negatives) increases. In 5-image fusion, the number of FN is less than that in 3-image fusion. (Figure 28)

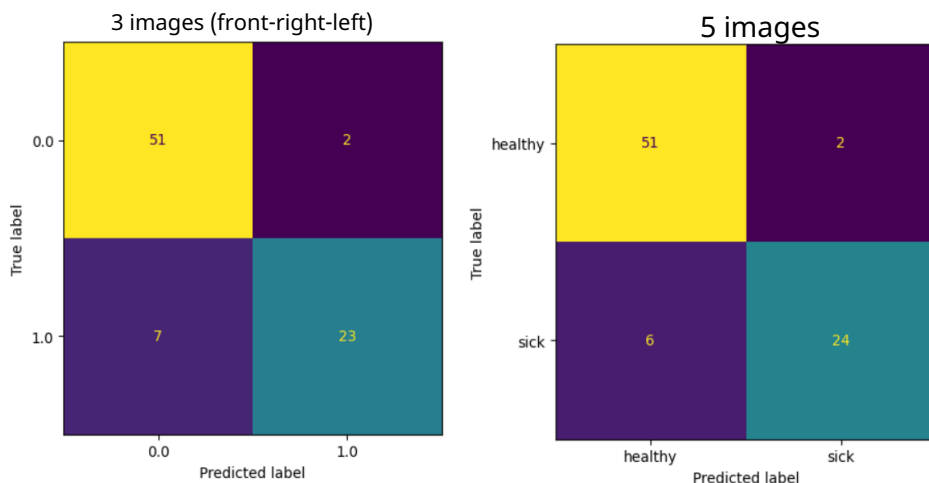


Figure 29. Classification with image fusion (SVM)

Vector machine (SVM) for combining images and artificial neural network (NN) has been used to achieve more successful results. The combinations made using 3 and 5 images in the neural network combination are shown in Figure 29. The number of false positives (FP) and false negatives (FN) in the neural network is close to the SVM, and these values are lower in the result obtained with 5 images.

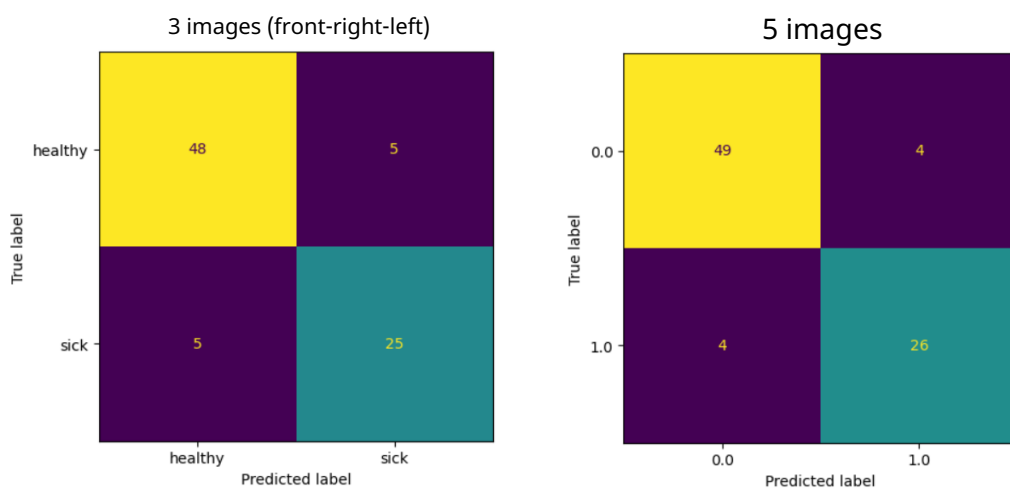


Figure 30.

The distribution of FN and FP numbers in the combination results of the artificial neural network is more balanced numerically, but there has been no change in the total number of FN and FP in the 5 image combination. Different results were tried using the alternative autocoder layer structure explained in the experiment section and shown in Figure 18 (v2). As in the first autocoder type, first, only the front image was used to test patient classification. (Figure 30) The number of FN is less in the v2 autocoder type compared to v1 (Figure 27).

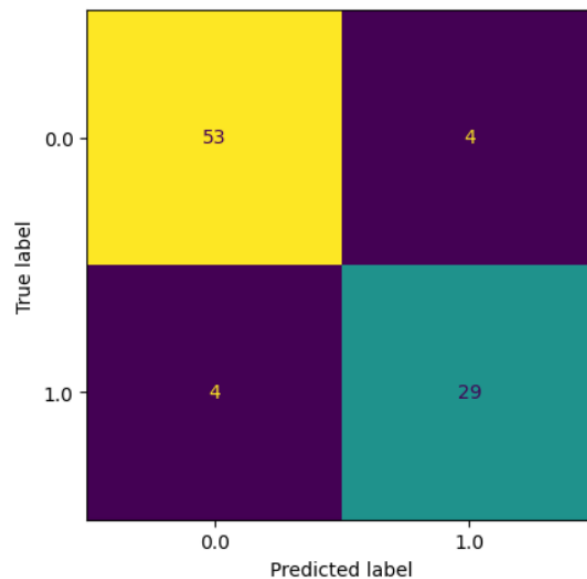


Figure 31.

Then, the results of the artificial neural network and 3 and 5 image merge results are given in Figure 31. The number of FN is less than 3 visual mergers and the number of FP is less than 5 video SVM merging results.

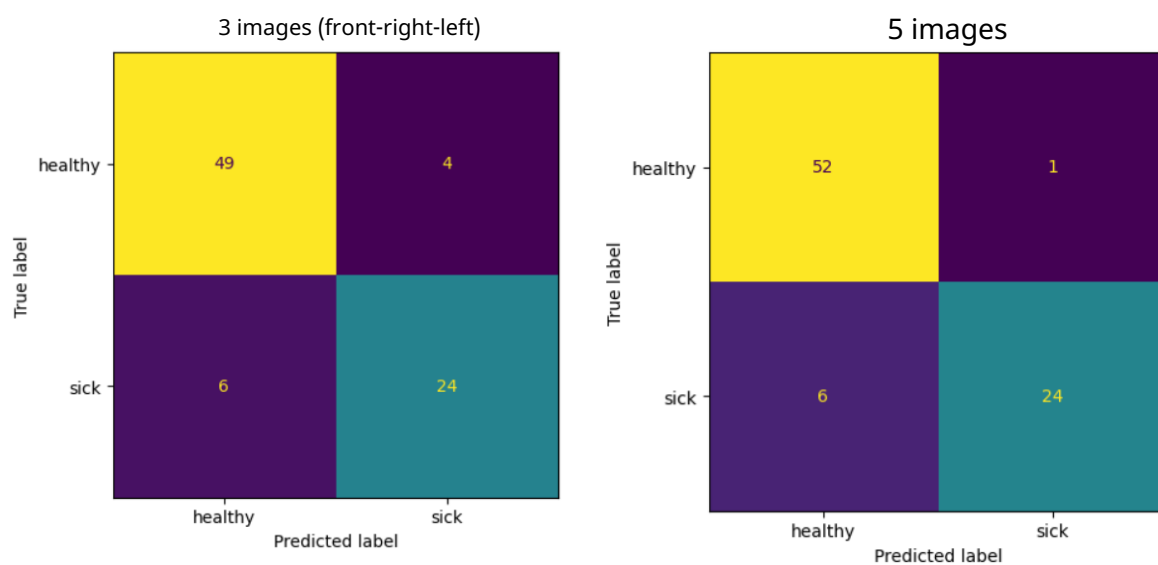


Figure 32. Classification with image fusion (NN- Autoencoder v2)

Since the better results were obtained in combining the autocoder V2 and artificial network, the consequences of joining processes were examined using the autocoder V2 again for merging with SVM. (Shape 32) There is no difference between 3 images and 5 images merge in the results obtained. Result success It is the same as the first layer of autocoder layer used in terms of 5 visuals.

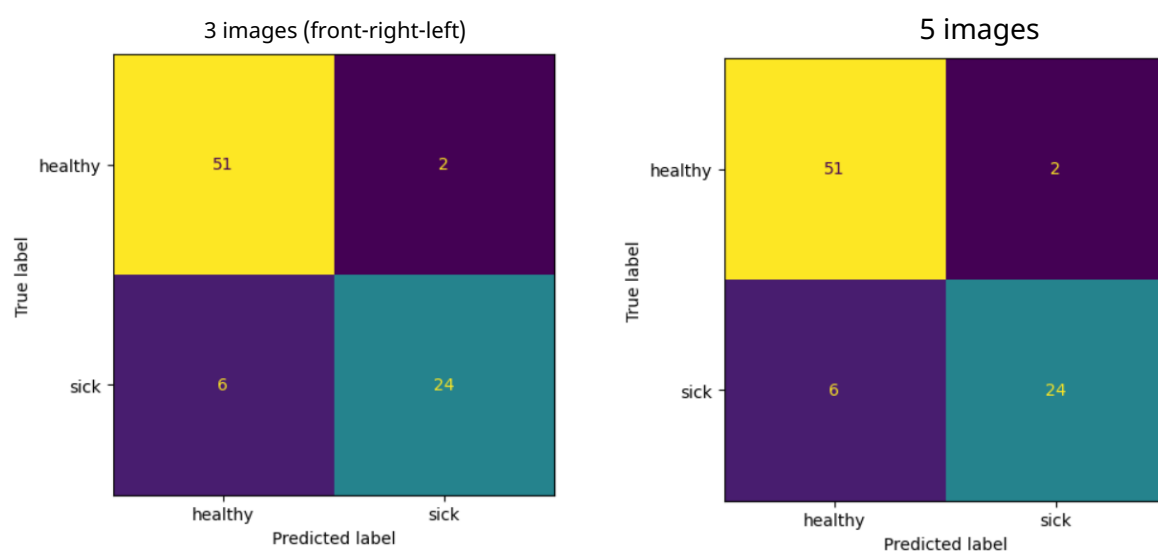


Figure 33. Classification with image fusion (SVM- Autoencoder v2)

SIT protocol measurements were taken after, and the first results obtained using the tunable LSTM network for DIT protocol images are shown in Figure 33. The number of FP and FN is close to each other. Subsequently, the layer structure was modified and the number of cells in the LSTM layer was increased from 12 to 24, resulting in a better outcome. (Figure 34)

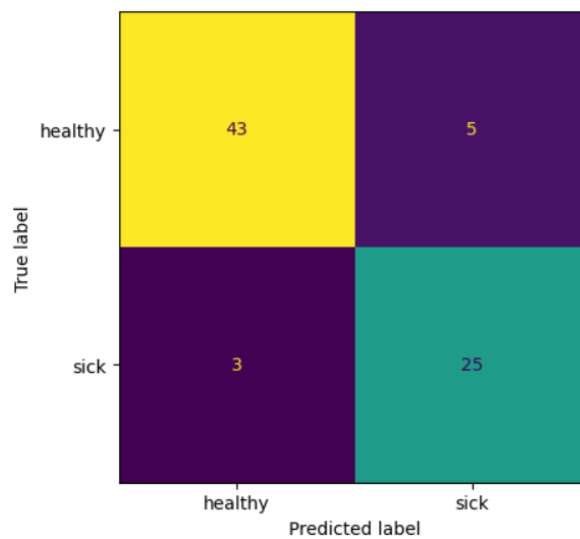


Figure 34. Classification of DIT images using an Evolvable LSTM (12-cell LSTM)

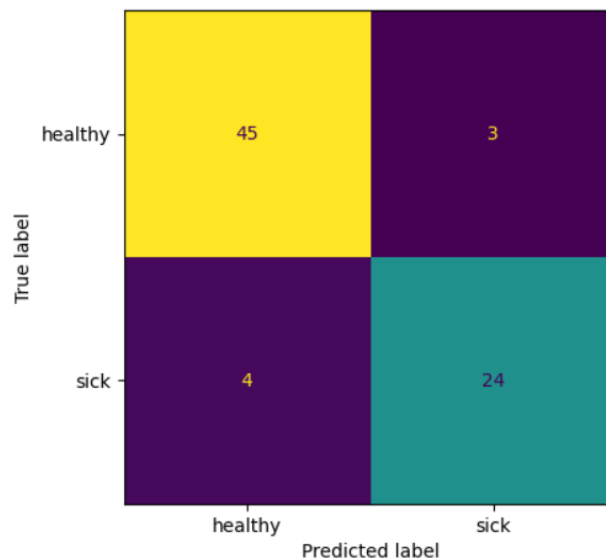


Figure 35. Image classification with a variable-length LSTM (24-cell LSTM)

Performance:

The performance metrics considered in the project and the results targeted to be achieved are given in Table 2 in the Introduction section. Performance analysis was carried out by calculating these metrics based on the TN, FP, FN, and TP numbers obtained from the measurements.

Firstly, the SIT protocol images were considered. Classification results obtained by using only the front view with two different autoencoders are given in Table 3. Here, the second type of autoencoder generally provided better results than the first type. The target sensitivity metric of 85% was achieved.

Table 3. Performance metrics for patient classification using only the preview

Metrics	Target Value	Encoder (v1)	Encoder (v2)
Precision (Precision)	93 ± 2	87,5	87,9
Sensitivity (Recall)	85 ± 3	84.9	87,9
Accuracy (Accuracy)	95 ± 3	90	91.1
F1 Score	0.92 ± 0.04	86.2	87,9

The results of the first autocoter type and SVM image merge results are given in Table 4. In terms of accuracy value, the results obtained by image merge and the results obtained with a single image are approximately the same. On the other hand, the combination of image at the value of certainty is significantly ahead and the targeted 92 %value has been reached. The sensitivity value is reduced according to a single image condition. While the value of the definite value increases in deep learning systems, a decrease in sensitivity is observed.

In general, 5 images in metrics give better results than 3 images.

When the second type of autocodeper (V2) and SVM image joining were applied, the difference was not observed between 3 and 5 images. (Table 5) The results obtained are the same as the result of 5 images using the first type of autocoder (V1).

Table 4. Performance metrics of image fusion using SVM (autoencoder v1)

Metrics	Target Value	3 images combination	5 images combination
Precision (Precision)	93 ± 2	92	92,3
Sensitivity (Recall)	85 ± 3	76.7	80
Accuracy (Accuracy)	95 ± 3	89.2	90.4
F1 Score	0.92 ± 0.04	83.6	85.7

Table 5. Performance metrics of image fusion using SVM (autoencoder v2)

Metrics	Target Value	3 images combination	5 images combination
Precision (Precision)	93 ± 2	92,3	92,3
Sensitivity (Recall)	85 ± 3	80	80
Accuracy (Accuracy)	95 ± 3	90.4	90.4
F1 Score	0.92 ± 0.04	85.7	85.7

Results obtained with the first type of autocoder in image fusion using artificial neural network have been provided in Table 6. Metrics are higher for 3 images out of 5, as in SVM. The accuracy metric is significantly lower than the SVM state while sensitivity increases. The accuracy value is approximately the same. This significant drop in accuracy was observed during the studies, so a second type of autocoder (v2) was developed and incorporated into the system.

Table 6. Image Marketing Performance Metriks with NN (Autocoter V1)

Metrics	Target Value	3 images combination	5 images combination
Precision (Precision)	93 ± 2	83.3	86.7
Sensitivity (Recall)	85 ± 3	83.3	86.7
Accuracy (Accuracy)	95 ± 3	87.95	90.4
F1 Score	0.92 ± 0.04	83.3	86.7

When applied in image fusion using the second type of auto-coder (v2) artificial neural network, the performance is significantly ahead of 3 image states in metrics. In the 5-image state, it has exceeded the best accuracy and precision values obtained in the SVM state.

Table 7. Image Marketing Performance Metriks with NN (Autocoter V2)

Metrics	Target Value	3 images combination	5 images combination
Precision (Precision)	93 ± 2	85.7	96
Sensitivity (Recall)	85 ± 3	80	80
Accuracy (Accuracy)	95 ± 3	87.95	91.6
F1 Score	0.92 ± 0.04	82.8	87.3

In terms of the SIT protocol, the system decides to detect the disease through multiple images, so the FP value decreases, which increases the value of certainty. However, since multiple appearances are needed for precise disease detection, this time the FN value increases, which reduces the sensitivity value.

The results of the LSTM nervous network model used for the examination of DIT protocol are given in Table 8. The result obtained for accuracy metric is similar to the SIT protocol. The certainty is lower. In terms of sensitivity metric, the DIT protocol has reached the targeted value.

Table 8.

Metrics	Target Value	Evrişimli LSTM
Precision (Precision)	93 ± 2	$88,9$
Sensitivity (Recall)	85 ± 3	85.7
Accuracy (Accuracy)	95 ± 3	90.8
F1 Score	0.92 ± 0.04	87.3

3) Result:

A deep learning model capable of detecting breast cancer from thermal images has been developed as a result of the work. The system can detect by integrating thermal images taken from different angles for the SIT protocol. The system has provided a high success rate in terms of accuracy metrics but is open to improvement in terms of sensitivity. SVM and artificial neural networks have been separately applied for image fusion. SVM is a machine learning method based on analytical relationships and therefore has few adjustable hyperparameters. In contrast, artificial neural networks with image fusion can analyze much more complex data. A model for image fusion with artificial neural networks has also been presented in this study, contributing to future work.

In this study, an additional deep learning model has also been developed for the use of images taken with the DIT protocol in disease detection. The developed model has implemented an evolutionary LSTM, which was not previously used for DIT protocol images. The working principle of the model is normally used for video classification. [32] Although this model does not provide as high metric values as the SIT protocol, it has achieved the desired goal in terms of sensitivity metrics and is open to improvement.

The developed model is important in terms of forming a foundation for subsequent studies.

In general, thermal images are examined in the study, but the models presented are also adapted for the analysis of different medical imaging or signals and can contribute to these areas. For example, brain waves or speech sound signals can be converted into spectrograms that can be shown as 2 -dimensional color images, and the evolutionary nervous network models presented in the study can be adapted to them.

Apart from these, TPU was used in the study and the codes given in the annexes were written specifically. TPU hardware has been developed by Google and is superior to speed than GPU, but its use is not as common as GPU. The codes given in the annexes section may create a template for the use of TPU for those who want to use this hardware but need experience.

Project Constraints at the Beginning: engineering constraints; economy, cost analysis, health, and producibility.

- 1.1. **Economy:** An more economical system has been developed in the project as an alternative to the targeted methods such as MRI and tomography. To keep the development cost at the lowest level, cloud computing has been used as mentioned. Additionally, by adding code during the system's training phase, the training time using TPU instead of the generally used GPU has been shortened. Thus, there is no need to purchase extra training time. The system that will be used on-site can also be connected to cloud computing and can operate, which means it can be implemented on standard and low-cost hardware.

2. Health: The developed system, as initially aimed, only operates based on thermal images and does not contain any element that could negatively affect human health during the detection phase. When the system is available, it will have a positive effect on human health with its contribution to the early detection of the disease.
3. Production: The system can only be realized in cloud computing environment as targeted. A standard hardware with internet connection will be sufficient for a system that can be used in the field. Thus, commercially common products can be used in the market.

4.4. Cost Analysis:

The cost analysis of the project was carried out as in the previously given table. There is no extra cost.

Design and Behavioral Features:

Parameter characteristics have been observed in the developed deep learning models. There are generally no specific starting rules within the optimal parameters of deep learning models. The optimal number of layers and the number of filters per layer have been determined through trial and error in this study.

In autocodians, the number of filters in the coder and decoder layers is very small number of the system reduces the efficiency, while a large number of parameters increases the number of parameters. As a general rule, as the layer progresses in the coder, the frame size decreases, while the number of filters increases, and in the decoder, the opposite is designed to be symmetrical.

Deep Learning Model The algorithm used for optimization during the training phase is given to the system with the hyperparameter of "optimizer". In general, it has been observed that the most successful results of the algorithm of Adaptive Moment Estimation.

The important parameters in the model training phase are "EPOCH" and "Batch". When the entire training set is passed through the trained system, an EPOCH is completed. BATCH determines the number of samples to be used in a training step. After this number of sample is given, the weight parameters are updated.

Epoch number is too small, the system may not be adequately trained. However, if this number is too large, the problem of overfitting (memorization) arises. In the study, the epoch number has been selected between 20 and 40 based on observations. To make the system more efficient, the batch size is generally chosen in the form of powers of two. [22] If this number is too small or too large, the training

is negatively affected, and the generally optimal value is found through trial and error. In the study, the optimal batch size has generally been used as 8, 16, and 32.

Forward-looking studies:

The next steps in this area could involve creating a new dataset from scratch with a large number of samples. One of the challenges encountered in the study is the limited number of samples in the used dataset, and a newly created dataset will facilitate easier generalization and obtaining better results.

The procedures examined have thermal images taken from a thermal camera in front of the patient. Mammography is a radiological imaging method where the chest tissue is placed between two plates for imaging. A device can be developed for thermal imaging to compress the tissue between two glass plates and capture the thermal image from above, thereby studying the effectiveness of this imaging method.

In addition, ultrasound is a widespread medical imaging technique, but it is generally used only in the abdominal area. A tumor in an unhealthy chest may be a denser tissue that ultrasound can possibly detect. The effectiveness of preliminary analyses using chest ultrasound images from both the right and left sides of the patient and healthy individuals can also be investigated.

4) References:

- [1] World Health Organization, "Breast Cancer," *who.int*, Mar. 13, 2024. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/breast-cancer>. [Accessed Apr. 11, 2024]
- [2] R. Lawson, "Implications of surface temperatures in the diagnosis of breast cancer," *Canadian Medical Association Journal* vol. 75, pp. 309-310, Aug. 1956.
- [3] M. B. Rakhunde, S. Gotarkar, S. G. Choudhari, M. B. Rakhunde, S. Gotarkar, and S. G. Choudhari, "Thermography as a Breast Cancer Screening Technique: A Review Article," *Cureus Journal of Medical Science*, vol. 14, no. 11, Nov. 2022, doi: <https://doi.org/10.7759/cureus.31251>.
- [4] D. Tsietso, A. Yahya, and R. Samikannu, "A Review on Thermal Imaging-Based Breast Cancer Detection Using Deep Learning," *Mobile Information Systems*, vol. 2022, pp. 119, 2022, doi: <https://doi.org/10.1155/2022/8952849>.
- [5] R. Roslidar, A. Rahman, R. Muharar et al., "A review on recent progress in thermal imaging and deep learning approaches for breast cancer detection," *IEEE Access*, vol. 8, pp. 116176–116194, 2020.
- [6] S. J. Mambou, P. Maresova, O. Krejcar, A. Selamat, and K. Kuca, "Breast cancer detection using infrared thermal imaging and a deep learning model," *Sensors*, vol. 18, no. 9, p. 2799, 2018.
- [7] G. Okal, "Meme Kanseri riskinin termal görüntüleme ve makine öğrenmesi ile saptanması," Yüksek Lisans tezi, Fen Bilimleri Enstitüsü, Pamukkale Üniversitesi, Denizli, Türkiye, 2019.
- [8] C. Cabioğlu, "Breast Cancer Diagnosis from Thermal Images," Master's Thesis, Faculty of Science, Başkent University, Ankara, Turkey, 2020.
- [9] T. Kirişken, "Classification of Thermal Images for Early Stage Breast Cancer Diagnosis Using Deep Learning," Master's Thesis, Faculty of Science, Ege University, İzmir, Turkey, 2022.
- [10] S. Ekici and H. Jawzal, "Breast cancer diagnosis using thermography and convolutional neural networks," *Medical Hypotheses*, vol. 137, p. 109542, 2020.
- [11] J. Zuluaga-Gomez, Z. Al Masry, K. Benagoune, S. Meraghni, and N. Zerhouni, "A CNN-based methodology for breast cancer diagnosis using thermal images," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 9, no. 2, pp. 131–145, 2020.

- [12] R. Sánchez-Cauce, J. Pérez-Martín, and M. Luque, "Multi-input convolutional neural network for breast cancer detection using thermal images and clinical data," *Computer Methods and Programs in Biomedicine*, vol. 204, p. 106045, 2021.
- [13] B. Yousefi, H. Akbari, M. Hershman, S. Kawakita, H.C. Fernandes, C. Ibarra-Castanedo, S. Ahadian, X. P. V. Maldague, "SPAER: Sparse Deep Convolutional Autoencoder Model to Extract Low Dimensional Imaging Biomarkers for Early Detection of Breast Cancer Using Dynamic Thermography," *Applied Sciences*, vol. 11, no. 7, p. 3248, 2021, doi: <https://doi.org/10.3390/app11073248>.
- [14] M. J. Mammoottil, L. J. Kulangara, A. S. Cherian, P. Mohandas, K. Hasikin, and M. Mahmud, "Detection of breast cancer from five-view thermal images using convolutional neural networks," *Journal of Healthcare Engineering*, vol. 2022, pp. 115, 2022.
- [15] E. A. Mohamed, E. A. Rashed, T. Gaber, and O. Karam, "Deep Learning Model for Fully Automated Breast Cancer Detection System from Thermograms," *PLOS ONE*, vol. 17, no. 1, 2022, doi: 10.1371/journal.pone.0262349
- [16] S. Civilibal, K. K. Cevik, A. Bozkurt, "A Deep Learning Approach for Automatic Detection, Segmentation, and Classification of Breast Lesions from Thermal Images," *Expert Systems with Applications*, vol. 212, p. 118774, 2023.
- [17] D. Alsaedi, A. Melnikov, K. Muzaffar, A. Mandelis and O. M. Ramahi, "A Microwave-Thermography Hybrid Technique for Breast Cancer Detection," *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, vol. 6, no. 1, pp. 153-163, 2022, doi: 10.1109/JERM.2021.3072451.
- [18] J. L. Gonzalez-Hernandez, A. N. Recinella, S. G. Kandlikar, D. Dabydeen, L. Medeiros, and P. Phatak, "Technology, application and potential of dynamic breast thermography for the detection of breast cancer," *International Journal of Heat and Mass Transfer*, vol. 131, pp. 558–573, 2019.
- [19] Y. Ohashi and I. Uchida, "Applying dynamic thermography in the diagnosis of breast cancer," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, no. 3, pp. 42-51, May-June 2000, doi: 10.1109/51.844379.
- [20] M. Abdel-Nasser, A. Moreno, and D. Puig, "Breast Cancer Detection in Thermal Infrared Images Using Representation Learning and Texture Analysis Methods," *Electronics*, vol. 8, no. 1, p. 100, Jan. 2019, doi: <https://doi.org/10.3390/electronics8010100>.

- [21] T. A. E. da Silva, L. F. da Silva, D. C. Muchaluat-Saade, and A. Conci, "A Computational Method to Assist the Diagnosis of Breast Disease Using Dynamic Thermography," *Sensors*, vol. 20, no. 14, p. 3866, July 2020, doi: <https://doi.org/10.3390/s20143866>.
- [22] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2021.
- [23] R. Pramoditha, "The Concept of Artificial Neurons (Perceptrons) in Neural Networks," *towardsdatascience.com*, Dec. 29, 2021. [Online]. Available: <https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc>. [Accessed Apr. 19, 2024]
- [24] IBM, "What Are Neural Networks?," *www.ibm.com*, 2023. [Online]. Available: <https://www.ibm.com/topics/neural-networks>. [Accessed Apr. 19, 2024]
- [25] E. Grossi and M. Buscema, "Introduction to artificial neural networks," *European Journal of Gastroenterology & Hepatology*, vol. 19, no. 12, pp. 1046–1054, Dec. 2007, doi: <https://doi.org/10.1097/meg.0b013e3282f198a0>.
- [26] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional Neural networks: an Overview and Application in Radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: <https://doi.org/10.1007/s13244-018-0639-9>.
- [27] J. Jain, "What Are Tensors?," *medium.com*, Jan. 13, 2023. [Online]. Available: https://medium.com/@jayeshjain_246/what-are-tensors-495cf37c18e6. [Accessed Apr. 20, 2024]
- [28] S. Bishayee, "The Basic Concept of Autoencoder — The Self-supervised Deep Learning," *medium.com*, Apr. 13, 2023. [Online]. Available: <https://medium.com/@soumallya160/the-basic-concept-of-autoencoder-the-self-supervised-deep-learning-454e75d93a04>. [Accessed Apr. 20, 2024]
- [29] M. Seeland and P. Mäder, "Multi-view classification with Convolutional Neural Networks," *PLOS ONE*, vol. 16, no. 1, 2021. doi:10.1371/journal.pone.0245230.
- [30] Z. Khademi, F. Ebrahimi, and H. M. Kordy, "A transfer learning-based CNN and LSTM hybrid deep learning model to classify motor imagery EEG signals," *Computers in Biology and Medicine*, vol. 143, p. 105288, Apr. 2022, doi: 10.1016/j.compbiomed.2022.105288.
- [31] L. F. Silva et al., "A New Database for Breast Research with Infrared Image," *Journal of Medical Imaging and Health Informatics*, vol. 4, no. 1, pp. 92–100, Mar. 2014, doi: <https://doi.org/10.1166/jmihi.2014.1226>.

- [32] P. Ferlet, “Training neural network with image sequence, an example with video as input,” medium.com, Dec. 05, 2019. [Online]. <https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f>. [Accessed Apr. 20, 2024]

5) License Completion Project Self-Evaluation Form:

1.1.	<p>What is the scope of your project design (prototyping, simulation, or analysis)? <i>The project is software-centric. Deep learning models have been created using the Keras and Tensorflow libraries in Python within the Google Colab cloud computing environment. The created models have been trained using TPU processing units within the Google Colab environment.</i></p>
2.	<p>Explain the design methods you use. <i>When designing deep learning models, models that have been developed previously have been reviewed first. Ensure that the modular components are tested and working correctly within themselves before being combined with each other. The necessary parameters have been arranged to provide the most accurate results.</i></p>
3.	<p>What engineering standards do you use or consider? <i>The following standards have been considered for the project: ITU-T F.748.12 "Deep learning software framework evaluation methodology", ITU-T Y.3531 "Cloud computing – Functional requirements for machine learning as a service", and DIN SPEC 13266 "Guideline for the development of deep learning image recognition systems".</i></p>
4.4.	<p>What are the realistic constraints that you use or take into account (at least two of the others should be selected to be compulsory. The restrictions that are not covered by the project should be deleted.)</p> <p><i>a) Economy:</i></p> <p><i>In order to make the study more economical in terms of budget, cloud computing service was used for calculation by obtaining lower cost and standard equipment instead of high capacity hardware. This method will be more economical than MR or mammography, as the last model will only use the thermal image for disease detection. With the same Colab Pro membership cost, both GPU and TPU can be used evenly, but TPU that calculates about 3 times faster instead of GPU for calculation has been preferred because the cost of labor is an important input.</i></p> <p><i>b) Cost analysis:</i></p> <p><i>The current total cost of the project is 327,810.12 TL. (Table 3) If the cloud computing constraint mentioned in the Economics section were not used, a high-capacity hardware would have to be purchased, and the cost would be much higher.</i></p> <p><i>c) Scalability:</i></p> <p><i>All materials used in the project are standard materials available in the market and used in various fields. Similarly, since the trained model can work based on cloud computing, the system can also be set up with more standard and cost-effective hardware. This would also shrink the system, thereby increasing mobility. Without using clinical data for classification, the system can be used on-site without the need for this data, thereby increasing its applicability. Otherwise, during the screening, data related to the patient to be scanned would also need to be entered into the system by an element. This would increase the number of personnel in the field team.</i></p> <p><i>d) Health:</i></p> <p><i>The project will be developed in the software environment, just like a computer thermal camera</i></p>

It is not possible to contain a component that will affect human health. Mammography method exposes the patient to radiation, standard thermal camera No invasive (non-invasive) operation and has no side effects on health. Also, it will be easy and free to use when the system is put into operation, so it will be widely used on site to enable early detection of breast cancer in more individuals, thereby benefiting human health.

5. Explain how you manage the project. Specify the tasks that each student is responsible for, along with the workload (task description and percentage) and time frames.

The project has been carried out by a single student. All tasks have been completed 100% by the single student. The time spent on the work packages is as follows:

Gain experience through example applications – 2 weeks

Obtaining and downloading the dataset from the internet – 2 weeks

Designing deep learning neural networks – 2 weeks

Obtaining initial results from the neural networks – 2 weeks

Improving the neural network based on the results – 6 weeks

Preparing the development report – 2 weeks

Developing additional models to nervous networks and testing these 6 weeks to prepare 6 weeks

Presentation and Poster Preparation 2 Weeks

6. Explain the reason if there is a change/deviation according to the calendar and/or subject made in the suggestion report.

Yes, the calendar has been changed according to the proposal report. The autocoder design was changed to ensure better results from the deep learning model developed for the SIT protocol, and an alternative to SVM, an artificial neural network, was also tried for the image fusion process. Since no similar work had been done for the DIT protocol before, a model example was taken for video classification. Additional time was needed to adapt this model to the work.

7. What kind of problems and challenges did you encounter during the project work? What types of solutions did you provide for these problems and challenges?

Data set used in the project was not in a ready-to-use rar file. It was provided on a website with a user interface. It would be difficult to download the files one by one, so the links were extracted using webscraping methods, and the data was downloaded and saved hierarchically with a separate Python script. In addition, the data was normally very large in size, but this problem was also solved by using cloud computing.

In the development of deep learning models, there are no acceptable rules for initial conditions as in classical circuit designs, so the development of models is generally based on trial and error. Since TPUs were preferred instead of GPUs in the project, the test times spent on trial and error were minimized, and the project was completed within the working schedule.

6) Attachments

Download Code

```
import os.path

from os import path

import re

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

dir_path = "/gdrive/My Drive/BitirmeTezi/"

database_path=dir_path+'ThermalDatabase\'

os.mkdir(database_path)

static_path=database_path+'\'+"Static"

dynamic_path=database_path+'\'+"Dynamic"

frontal_path=static_path+'\'+"Frontal"

right45_path=static_path+'\'+"Right45"

right_path=static_path+'\'+"Right"

left45_path=static_path+'\'+"Left45"

left_path=static_path+'\'+"Left"

frontal_path_healthy=frontal_path+'\'+"healthy"

right45_path_healthy=right45_path+'\'+"healthy"

right_path_healthy=right_path+'\'+"healthy"

left45_path_healthy=left45_path+'\'+"healthy"

left_path_healthy=left_path+'\'+"healthy"

frontal_path_sick=frontal_path+'\'+"sick"

right45_path_sick=right45_path+'\'+"sick"
```

```
right_path_sick=right_path+'/'+"sick"
left45_path_sick=left45_path+'/'+"sick"
left_path_sick=left_path+'/'+"sick"
dynamic_path_healthy=dynamic_path+'/'+'healthy\'
dynamic_path_sick=dynamic_path+'/'+'sick\'
directions=[frontal_path,right45_path,right_path,left45_path,left_path]

os.mkdir(static_path)

os.mkdir(dynamic_path)

os.mkdir(frontal_path)

os.mkdir(right45_path)

os.mkdir(right_path)

os.mkdir(left45_path)

os.mkdir(left_path)

os.mkdir(frontal_path_healthy)

os.mkdir(right45_path_healthy)

os.mkdir(right_path_healthy)

os.mkdir(left45_path_healthy)

os.mkdir(left_path_healthy)


os.mkdir(frontal_path_sick)

os.mkdir(right45_path_sick)

os.mkdir(right_path_sick)

os.mkdir(left45_path_sick)

os.mkdir(left_path_sick)

os.mkdir(dynamic_path_healthy)
```

```
os.mkdir(dynamic_path_sick)

!pip install mechanicalsoup

!pip install html5lib

import mechanicalsoup

import html5lib

browser = mechanicalsoup.StatefulBrowser()

browser.open("http://visual.ic.uff.br/dmi/")

browser.select_form('form[action="login.php"]')

browser.form.print_summary()

browser["usuario"] = 'xxx'

browser["password"] = 'xxx'

browser.submit_selected()

browser.url

#list_pages=page.find("div",{"class":"pagination pagination-centered"}).find("ul").find_all("li")

#tag=soup.find_all("div",{"class": "span5 form-inline"})

healthy_id_list=[]

patient_mode="S"

page=browser.request('\GET',f'http://visual.ic.uff.br/dmi/prontuario/index.php?filtro=0&diag={patient_mode}&order=1\').soup

list_pages=page.find("div",{"class":"pagination pagination-centered"}).find("ul").find_all("li")

list_pages

pages=[]

for page_li in list_pages:

    page_num=page_li.find(["a","span"]).getText()

    if page_num!="Next":
```

```

    pages.append(page_num)

else:

    break

pages

for page_num in pages:

    page=browser.request('\GET',f'http://visual.ic.uff.br/dmi/prontuario/index.php?diag={patient_mode}&filtro=0&order=1&by=1&pagina={page_num}\').soup

    list_patients=page.find_all("tr")

    for patient in list_patients[1:]:

        healthy_id_list.append(patient.find("td").getText())

print(healthy_id_list)

len(healthy_id_list)

healthy_id_list=list(set(healthy_id_list))

healthy_id_list

#page=browser.request('GET',f'http://visual.ic.uff.br/dmi/prontuario/details.php?id={id_num}').soup

#print(page.prettify())

for id_num in healthy_id_list:

    print(id_num)

    page=browser.request('GET',f'http://visual.ic.uff.br/dmi/prontuario/details.php?id={id_num}').

    soupsections=page.find_all("section",{ "class": "toggle", "id": re.compile(r'^exame-termico.*')})

    number_of_visits=len(sections)

    if number_of_visits>1:

        print("Dikkat!!!! Birden fazla visit!!!!")

        print(f"number of visits: {number_of_visits}")

    for i,section in enumerate(sections):

        print(i)

```

```

image_divs=section.find_all("div",{"class": "patient_images"})

txt_refs=image_divs[1].find("ul").find_all("li")

id_path=dynamic_path_healthy+'\'\''+str(id_num)+'\'\''+str(i)

os.mkdir(id_path)

for txt_li in txt_refs:

    filename=txt_li.a.getText()

    parameters=txt_li.a.getText().split(".")

    print(txt_li.a)

    if(parameters[3]==\'S\'):

        direction=int(parameters[2])

        browser.download_link(link=txt_li.a,file=directions[direction-1]+'/'+'healthy/'+'filename)

    else:

        browser.download_link(link=txt_li.a,file=id_path+'/'+'filename)

type(image_divs)

sick_id_list=[]

patient_mode="D"

page=browser.request(\'GET\',f\'http://visual.ic.uff.br/dmi/prontuario/index.php?filtro=0&diag={patient_mode}&order=1\').soup

list_pages=page.find("div",{"class":"pagination pagination-centered"}).find("ul").find_all("li")

list_pages

pages=[]

for page_li in list_pages:

    page_num=page_li.find(["a","span"]).getText()

    if page_num!="Next":

        pages.append(page_num)

    else:

```

```

    break

pages

for page_num in pages:

    page=browser.request('\GET',f'http://visual.ic.uff.br/dmi/prontuario/index.
    php?diag={patient_mode}&filtro=0&order=1&by=1&pagina={page_num}\').soup

    list_patients=page.find_all("tr")

    for patient in list_patients[1:]:

        sick_id_list.append(patient.find("td").getText())

print(sick_id_list)

len(sick_id_list)

sick_id_list=list(set(sick_id_list))

sick_id_list

#page=browser.request('GET',f'http://visual.ic.uff.br/dmi/prontuario/details.php?id={id_num}').soup

#print(page.prettify())

for id_num in sick_id_list:

    print(id_num)

    page=browser.request('GET',f'http://visual.ic.uff.br/dmi/prontuario/details.php?id={id_num}').

    soupsections=page.find_all("section",{ "class": "toggle", "id":re.compile(r'^exame-termico.*')})

    if len(sections)>1:

        print ("Attention !!!! Multiple Visit !!!!")

    for i,section in enumerate(sections):

        print(i)

        image_divs=section.find_all("div",{ "class": "patient_images"})

        txt_refs=image_divs[1].find("ul").find_all("li")

        id_path=dynamic_path_sick+'/' +str(id_num)+'_'+str(i)

        os.mkdir(id_path)

```

```
for txt_li in txt_refs:

    filename=txt_li.a.getText()

    parameters=txt_li.a.getText().split(".")

    print(txt_li.a)

    if(parameters[3]=='S\'):

        direction=int(parameters[2])

        browser.download_link(link=txt_li.a,file=directions[direction-1]+'/'+'sick/'+filename)

    else:

        browser.download_link(link=txt_li.a,file=id_path+'/'+'filename)
```


EC-2 JPG Conversion Code

```

import numpy as np
import os
import matplotlib.pyplot as plt
from urllib import request
from urllib.error import HTTPError
import re
from google.colab import drive
drive.mount("/gdrive")
# %cd /gdrive
directory_path = "/gdrive/My Drive/Yedek/ThermalDatabase"
save_path = '/gdrive/My Drive/Yedek/Thermal_Image\'
def convertImage(target,destination,delimiter=None):
    m1=np.loadtxt(target,dtype='float\',delimiter=delimiter)
    plt.set_cmap('viridis')
    plt.imsave(destination,m1)
front_exclusion_list=[]
"""# **Create a recursive function to walk subdirectories**"""
def walk_subdirectories(d_path,s_path):
    subdirectories, files = next(os.walk(d_path))[1:]
    print(d_path, s_path)
    print(subdirectories)
    print(files)
    if(len(files)==0):
        print("No files yet, keep walking...")
        for subdirectory in subdirectories:
            os.mkdir(s_path+'\'\''+subdirectory)
            walk_subdirectories(d_path+'\'\''+subdirectory,s_path+'\'\''+subdirectory)
    else:
        print("Files found, converting files...")
        for file in files:
            t=d_path+'\'\''+file
            filename=file.split('txt\')[0]
            d=s_path+'\'\''+filename+'\'\''+'.jpg\'
            try:
                convertImage(t,d)

```

```

    print("File is OK...")
except ValueError as v1:
    print("File is not OK... Try delimiter ';'")
    try:
        convertImage(t,d,delimiter=";")
        print("File is now OK... delimiter is ';' !!!!!!!")
    except ValueError as v2:
        print("File still not OK, Try to download again...")
        patient_id=re.split('[A-Z]',file.split(".")[0])[1]
        d_link=f"http://visual.ic.uff.br/dmi/bancovl/{patient_id}/"
        print(f"Download Link:{d_link}")
        try:
            request.urlretrieve(d_link+file,t)
        try:
            convertImage(t,d)
            print("File is now OK after download...")
        except ValueError as v3:
            print("File is not OK after download... Try delimiter ';'")
            try:
                convertImage(t,d,delimiter=";")
                print("File is now OK after download... delimiter is ';' !!!!!!!")
            except ValueError as v4:
                print("\'Very exceptional case\'!!!! File is not ok even after download and delimiter .... : SS")
                print(file)
    except HTTPError as h:
        print("No file exists, HTTP 404 Error!!!!!!")
"""# **Initiate Recursion**"""
walk_subdirectories(directory_path,save_path)

```

Obtaining Tensorflow Dataset Code

```
import os
from os.path import join
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras
import tensorflow.keras.layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/gdrive")
# %cd /gdrive
img_height, img_width = (480, 640)
image_size = (480, 640)
batch_size=16
d_path = "/gdrive/My Drive/Yedek/Thermal_Image/Static"
subdirectories = next(os.walk(d_path))[1]
print(subdirectories)
for directory in subdirectories:
    print(directory)
    train_ds, val_ds = tf.keras.utils.image_dataset_from_directory(
        join(d_path,directory),
        labels='inferred',
        label_mode='int',
        color_mode='rgb',
        batch_size=batch_size,
        image_size=image_size,
        shuffle=True,
        seed=1,
        validation_split=0.3,
        subset='both',
        interpolation='bilinear',
        follow_links=False,
        crop_to_aspect_ratio=False,
    )
class_names = train_ds.class_names
```

```
print(class_names)
train_ds.save(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/train")
val_ds.save(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/val")
```

Image Filtering Code

```

import numpy as np

import os

import matplotlib.pyplot as plt

import shutil

import re

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

reference_id_list={'Frontal':[],

                  'Right45':[],

                  'Right':[],

                  'Left45':[],

                  'Left':[]}

final_id_list=set()

directory_path='/gdrive/My Drive/Yedek/Thermal_Image/Static'

save_path='/gdrive/My Drive/Yedek/Thermal_Image/Static_filtered_v1'

root,subdirectories,files=next(os.walk(directory_path))

for subdirectory in subdirectories:

    root2, subdirectories2, files2 = next(os.walk(os.path.join(root, subdirectory)))

    for subdirectory2 in subdirectories2:

        files = next(os.walk(os.path.join(root2, subdirectory2)))[2]

        for file in files:

            file_id = re.split('[A-Z]0*\', file.split(".")[0])[1]

            file_id += '\_' + file.split(".")[1]

```

```
        reference_id_list[subdirectory].append(file_id)

setList=[]

for subdirectory in subdirectories:

    setList.append(set(reference_id_list[subdirectory]))

final_id_list=setList[0]

print(len(final_id_list))

for set_el in setList:

    final_id_list=final_id_list.intersection(set_el)

print(final_id_list)

print(len(final_id_list))

final_id_list=list(final_id_list)

print(final_id_list)

print(sorted(final_id_list))

def walk_subdirectories(d_path,s_path):

    subdirectories, files = next(os.walk(d_path))[1:]

    print(d_path, s_path)

    print(subdirectories)

    print(files)

    if(len(files)==0):

        print("No files yet, keep walking...")

        for subdirectory in subdirectories:

            os.mkdir(s_path+'\\'+subdirectory)

            walk_subdirectories(d_path+'/'+subdirectory,s_path+'/'+subdirectory)

    else:

        print("Files found, checking ids...")
```

```
for file in files:

    file_id = re.split('\[A-Z]0*\', file.split(".")[0])[1]

    file_id += '\_'+file.split(".")[1]

    if file_id in final_id_list:

        print(f"{file_id} is consistent, copying the file to filtered folder...")

        shutil.copy(d_path+'\'+file, s_path+'\'+f"image_{file_id}.jpg")

    else:

        print(f"{file_id} is not consistent!!!")

walk_subdirectories(directory_path,save_path)
```

EK-5: Oto Kodlayıcı Kodu (v1)

```

import tensorflow as tf
import os
import tensorflow_datasets as tfds
resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu='')
tf.config.experimental_connect_to_cluster(resolver)
This is the TPU initialization code that has to be at the beginning.
tf.tpu.experimental.initialize_tpu_system(resolver)
print("All devices: ", tf.config.list_logical_devices('TPU'))
strategy = tf.distribute.TPUStrategy(resolver)
Commented out IPython magic to ensure Python compatibility.
!pip install hickle
import hickle as hkl
import os
import matplotlib.pyplot as plt
from keras import layers, Input, Sequential
from keras.layers import
    Dense, Flatten, Reshape, Conv2DTranspose, Conv2D, MaxPooling2D, Reshape, Resizing, Dropout, Up
    Sampling2D, BatchNormalization
from tensorflow.keras import datasets, layers, models, losses, Model
from random import randint
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/gdrive")
# %cd /gdrive
load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_v1/Static/Frontal.hkl")
X_train = load_hkl['xtrain'] / 255.
X_test = load_hkl['xtest'] / 255.
y_train = load_hkl['ytrain']
y_test = load_hkl['ytest']
X_train.shape
class CNN_Autoencoder(Model):
    def __init__(self):
        super(CNN_Autoencoder, self).__init__()

```



```

self.encoder = tf.keras.Sequential([
    layers.Input(shape=X_train.shape[1:]),
    layers.Conv2D(8, 3, strides=1, padding='same', activation='relu'),
    layers.MaxPooling2D(2, strides=2),
    layers.Conv2D(16, 3, strides=1, padding='same', activation='relu'),
    layers.MaxPooling2D(2, strides=2),
    layers.Conv2D(32, 3, strides=1, padding='same', activation='relu')])

self.decoder = tf.keras.Sequential([
    layers.Conv2D(32, 3, strides=1, padding='same', activation='relu'),
    layers.UpSampling2D(2),
    layers.Conv2D(16, 3, strides=1, padding='same', activation='relu'),
    layers.UpSampling2D(2),
    layers.Conv2D(3, 3, strides=1, padding='same', activation='relu')])

def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

'''
cnn_autoencoder=CNN_Autoencoder()
dot_img_file = '/tmp/model_1.png'
keras.utils.plot_model(cnn_autoencoder.encoder,to_file=dot_img_file, show_shapes=False)
'''

epochs=20
with strategy.scope():
    cnn_autonecoder_front = cnn_autonecoder () cnn_autonescront.COME
    ']'

    batch_size=32, shuffle=True,validation_data=(X_test,X_test))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(epochs)
plt.figure(figsize=(8, 8))

```

```

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
save_locally = tf.saved_model.SaveOptions(experimental_io_device='/job:localhost')
cnn_autoencoder_front.save('/gdrive/My Drive/Models_Static_v1/Frontal', options=save_locally)
with strategy.scope():
    load_locally = tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
    reconstructed_front = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v1/Frontal',
    options=load_locally)
xm_train=reconstructed_front.encoder(X_train)
xm_test=reconstructed_front.encoder(X_test)
xm_train.shape
xm_train.shape[0]
xm_train=tf.reshape(xm_train,[xm_train.shape[0],-1])
xm_test=tf.reshape(xm_test,[xm_test.shape[0],-1])
"""## **Machine Learning**"""
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
xm_train=scaler.fit_transform(xm_train)
xm_test=scaler.transform(xm_test)
from sklearn.svm import SVC
svc=SVC(kernel="sigmoid")
svc.fit(xm_train, y_train)
y_pred=svc.predict(xm_test)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm=confusion_matrix(y_test, y_pred)
print(cm)

```

```
print(classification_report(y_test, y_pred))  
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)  
disp.plot()
```

EC-6: Automotive Coding Code (v2)

```
import tensorflow as tf
import os
import tensorflow_datasets as tfds
```

```
resolver = tf.distribute.cluster_resolver.TPUClusterResolver()
tf.config.experimental_connect_to_cluster(resolver)
This is the TPU initialization code that has to be at the beginning.
tf.tpu.experimental.initialize_tpu_system(resolver)
print("All devices: ", tf.config.list_logical_devices('TPU'))
```

```
strategy = tf.distribute.TPUStrategy(resolver)
```

Commented out IPython magic to ensure Python compatibility.

```
!pip install hickle
import hickle as hkl
import os
import matplotlib.pyplot as plt
from keras import layers, Input, Sequential
from keras.layers import
    Dense, Flatten, Reshape, Conv2DTranspose, Conv2D, MaxPooling2D, Reshape, Resizing, Dropout, Up
    Sampling2D, BatchNormalization
from tensorflow.keras import datasets, layers, models, losses, Model
from random import randint
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/gdrive")
# %cd /gdrive

load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_v1/Static/Frontal.hkl")
X_train = load_hkl['xtrain'] / 255.
X_test = load_hkl['xtest'] / 255.
y_train = load_hkl['ytrain']
y_test = load_hkl['ytest']
```

```

class CNN_Autoencoder(Model):
    def __init__(self):
        super(CNN_Autoencoder, self).__init__()
        self.encoder = tf.keras.Sequential([
            layers.Input(shape=X_train.shape[1:]),
            layers.Conv2D(8, 3, strides=2, padding='same', activation='relu'),
            layers.Conv2D(16, 3, strides=2, padding='same', activation='relu'),
            layers.Conv2D(32, 3, strides=2, padding='same', activation='relu')])

        self.decoder = tf.keras.Sequential([
            layers.Conv2DTranspose(32, 3, strides=2, padding='same', activation='relu'),
            layers.Conv2DTranspose(16, 3, strides=2, padding='same', activation='relu'),
            layers.Conv2DTranspose(3, 3, strides=2, padding='same', activation='relu')])

    def call(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

'''
cnn_autoencoder=CNN_Autoencoder()
dot_img_file = '/tmp/model_1.png'

keras.utils.plot_model(cnn_autoencoder.encoder,to_file=dot_img_file, show_shapes=False)
'''

epochs=40

with strategy.scope():
    cnn_autonecoder_front = cnn_autonecoder () cnn_autonescront.COME
    '

    batch_size=32, shuffle=True,validation_data=(X_test,X_test))

acc = history.history['accuracy']

```

```

val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title("Training and Validation Accuracy")

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title("Training and Validation Loss")
plt.show()

save_locally = tf.saved_model.SaveOptions(experimental_io_device=tf.io.local_device_name())
cnn_autoencoder_front.save('/gdrive/My Drive/Models_Static_v1/Frontal', options=save_locally)

with strategy.scope():
    load_locally = tf.saved_model.LoadOptions(experimental_io_device=tf.io.local_device_name())
    reconstructed_front = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v1/Frontal',
    options=load_locally)

xm_train=reconstructed_front.encoder(X_train)
xm_test=reconstructed_front.encoder(X_test)

xm_train=tf.reshape(xm_train,[xm_train.shape[0],-1])
xm_test=tf.reshape(xm_test,[xm_test.shape[0],-1])

"""## **Machine Learning**"""

from sklearn.metrics import classification_report

```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
xm_train=scaler.fit_transform(xm_train)
xm_test=scaler.transform(xm_test)

from sklearn.svm import SVC
svc=SVC(kernel="sigmoid")
svc.fit(xm_train, y_train)
y_pred=svc.predict(xm_test)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm=confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test, y_pred))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svc.classes_)
disp.plot()
```

Auto Encoder Training Set Acquisition Code

```
!pip install hickle

import hickle as hkl

import os

import numpy as np

import tensorflow as tf

from tensorflow import keras

import matplotlib.pyplot as plt

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

image_size = (480,640)

d_path = "/gdrive/My Drive/Yedek/TF_Datasets_v1"

subdirectories=next(os.walk(d_path))[1]

print(subdirectories)

for directory in subdirectories:

    train_ds=tf.data.Dataset.load(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/train")

    val_ds=tf.data.Dataset.load(f"/gdrive/My Drive/Yedek/TF_Datasets_v1/{directory}/val")

    X_train=np.empty((0,image_size[0],image_size[1],3),dtype='float32')

    y_train=np.empty((0,),dtype='float32')

    X_test=np.empty((0,image_size[0],image_size[1],3),dtype='float32')

    y_test=np.empty((0,),dtype='float32')

    for image_batch, labels_batch in train_ds:

        X_train=np.concatenate((X_train,image_batch),axis=0)

        y_train=np.concatenate((y_train,labels_batch),axis=0)
```



```
for image_batch, labels_batch in val_ds:

    X_test=np.concatenate((X_test,image_batch),axis=0)

    y_test=np.concatenate((y_test,labels_batch),axis=0)

data = {'xtrain\': X_train, \'xtest\': X_test,\'ytrain\': y_train,\'ytest\':y_test}

hkl.dump(data,f"/gdrive/My Drive/Yedek/Autoencoder_Datasets_v1/Static/{directory}.hkl")
```

Image Merging Code (SVM)**import tensorflow as tf****import os****import tensorflow_datasets as tfds****resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu='')****tf.config.experimental_connect_to_cluster(resolver)**

This is the TPU initialization code that has to be at the beginning.

tf.tpu.experimental.initialize_tpu_system(resolver)**print("All devices: ", tf.config.list_logical_devices('TPU'))****strategy = tf.distribute.TPUStrategy(resolver)**

Commented out IPython magic to ensure Python compatibility.

!pip install hickle**import hickle as hkl****import os****import matplotlib.pyplot as plt****from keras import layers, Input, Sequential****from keras.layers import****Dense, Flatten, Reshape, Conv2DTranspose, Conv2D, MaxPooling2D, Reshape, Resizing, Dropout, UpSampling2D, BatchNormalization****from tensorflow.keras import datasets, layers, models, losses, Model****from random import randint****import numpy as np****from tensorflow.keras import layers****from tensorflow.keras.models import Sequential, Model****import matplotlib.pyplot as plt****from google.colab import drive****drive.mount("/gdrive")**

```

# %cd /gdrive

load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_v1/Static/Frontal.hkl")

X_train = load_hkl['xtrain'] / 255.

X_test = load_hkl['xtest'] / 255.

y_train = load_hkl['ytrain']

y_test = load_hkl['ytest']

X_train.shape

class CNN_Autoencoder(Model):

    def __init__(self):

        super(CNN_Autoencoder, self).__init__()

        self.encoder = tf.keras.Sequential([

            layers.Input(shape=X_train.shape[1:]),

            layers.Conv2D(8, 3, strides=1, padding='same', activation='relu'),

            layers.MaxPooling2D(2, strides=2),

            layers.Conv2D(16, 3, strides=1, padding='same', activation='relu'),

            layers.MaxPooling2D(2, strides=2),

            layers.Conv2D(32, 3, strides=1, padding='same', activation='relu')])

        self.decoder = tf.keras.Sequential([

            layers.Conv2D(32, 3, strides=1, padding='same', activation='relu'),

            layers.UpSampling2D(2),

            layers.Conv2D(16, 3, strides=1, padding='same', activation='relu'),

            layers.UpSampling2D(2),

            layers.Conv2D(3, 3, strides=1, padding='same', activation='relu')])

    def call(self, x):

        encoded = self.encoder(x)

```

```

        decoded = self.decoder(encoded)

        return decoded

'''

cnn_autoencoder=CNN_Autoencoder()

dot_img_file = '/tmp/model_1.png'

keras.utils.plot_model(cnn_autoencoder.encoder,to_file=dot_img_file, show_shapes=False)

'''

epochs=20

with strategy.scope():

    cnn_autoencoder_front=CNN_Autoencoder()

    cnn_autoencoder_front.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

    history=cnn_autoencoder_front.fit(X_train,X_train,epochs=epochs,

        batch_size=32, shuffle=True,validation_data=(X_test,X_test))

    acc = history.history['accuracy']

    val_acc = history.history['val_accuracy']

    loss = history.history['loss']

    val_loss = history.history['val_loss']

    epochs_range = range(epochs)

    plt.figure(figsize=(8, 8))

    plt.subplot(1, 2, 1)

    plt.plot(epochs_range, acc, label='Training Accuracy')

    plt.plot(epochs_range, val_acc, label='Validation Accuracy')

    plt.legend(loc='lower right')

    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)

```

```

plt.plot(epochs_range, loss, label='Training Loss')

plt.plot(epochs_range, val_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

save_locally = tf.saved_model.SaveOptions(experimental_io_device='/job:localhost')

cnn_autoencoder_front.save('/gdrive/My Drive/Models_Static_v1/Frontal', options=save_locally)

with strategy.scope():

    load_locally = tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')

    reconstructed_front = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v1/Frontal',
options=load_locally)

xm_train=reconstructed_front.encoder(X_train)

xm_test=reconstructed_front.encoder(X_test)

xm_train.shape

xm_train.shape[0]

xm_train=tf.reshape(xm_train,[xm_train.shape[0],-1])

xm_test=tf.reshape(xm_test,[xm_test.shape[0],-1])

"""## **Machine Learning**"""

from sklearn.metrics import classification_report

from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()

transforming training data

transforming test data

from sklearn.svm import SVC

creating SVM classifier with sigmoid kernel

training SVM classifier

```

```
y_pred=svc.predict(xm_test)

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm=confusion_matrix(y_test, y_pred)

print(cm)

print(classification_report)plot confusion matrix with class labels
```

Image Merging Code (NN)

```
import tensorflow as tf
```

```
import garbage collector
```

```
import os
```

```
import numpy as np
```

```
import tensorflow_datasets as tfds
```

```
resolver = tf.distribute.cluster_resolver.TPUClusterResolver()
```

```
tf.config.experimental_connect_to_cluster(resolver)
```

This is the TPU initialization code that has to be at the beginning.

```
tf.tpu.experimental.initialize_tpu_system(resolver)
```

```
print("All devices: ", tf.config.list_logical_devices('TPU'))
```

```
strategy = tf.distribute.TPUStrategy(resolver)
```

Commented out IPython magic to ensure Python compatibility.

```
!pip install hickle
```

```
import hickle as hkl
```

```
import os
```

```
import matplotlib.pyplot as plt
```

```
from keras import layers, Input, Sequential
```

```
from keras.layers import
```

```
Dense, Flatten, Reshape, Conv2DTranspose, Conv2D, MaxPooling2D, Reshape, Resizing, Dropout, UpSampling2D
```

```
from tensorflow.keras import datasets, layers, models, losses, Model
```

```
from random import randint
```

```
import numpy as np
```

```
from tensorflow.keras import layers
```

```
from tensorflow.keras.models import Sequential, Model
```

```

import matplotlib.pyplot as plt

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

with strategy.scope():

    load_locally = tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')

    reconstructed_front = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v3/Frontal',
options=load_locally)

    reconstructed_left= tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v3/Left',
options=load_locally)

    reconstructed_right = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v3/Right',
options=load_locally)

    reconstructed_left45= tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v3/Left45',
options=load_locally)

    reconstructed_right45 = tf.keras.models.load_model('/gdrive/My Drive/Models_Static_v3/Right45',
options=load_locally)

load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_filtered_v1/Static/Frontal.hkl")

X_train_frontal=load_hkl['xtrain'] / 255.

X_test_frontal=load_hkl['xtest'] / 255.

y_train_frontal = load_hkl['ytrain']

y_test_frontal = load_hkl['ytest']

del load_hkl

gc.collect()

load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_filtered_v1/Static/Left.hkl")

X_train_left = load_hkl['xtrain'] / 255.

X_test_left = load_hkl['xtest'] / 255.

del load_hkl

```



```

gc.collect()

load_hkl = hkl.load("/gdrive/My\\ Drive/Yedek/Autoencoder_Datasets_filtered_v1/Static/Right.hkl")

X\\_train\\_right=load\\_hkl['xtrain'] / 255.

X\\_test\\_right=load\\_hkl['xtest'] / 255.

del load_hkl

gc.collect()

load\\_hkl = hkl.load("/gdrive/My\\ Drive/Yedek/Autoencoder_Datasets_filtered_v1/Static/Left45.hkl")

X\\_train\\_left45=load\\_hkl['xtrain'] / 255.

X_test_left45 = load_hkl['xtest'] / 255.

del load_hkl

gc.collect()

load_hkl = hkl.load("/gdrive/My Drive/Yedek/Autoencoder_Datasets_filtered_v1/Static/Right45.hkl")

X_train_right45 = load_hkl['xtrain'] / 255.

X_test_right45 = load_hkl['xtest'] / 255.

del load_hkl

gc.collect()

xm_train_frontal = reconstructed_front.encoder(X_train_frontal)

xm_test_frontal = reconstructed_front.encoder(X_test_frontal)

del X_train_frontal

delete X_test_frontal

gc.collect()

xm_train_left=reconstructed_left.encoder(X_train_left)

xm_test_left=reconstructed_left.encoder(X_test_left)

delete X_train_left

delete X_test_left

```

```

gc.collect()

xm_train_right=reconstructed_right.encoder(X_train_right)

xm_test_right=reconstructed_right.encoder(X_test_right)

delete X_train_right

del X_test_right

gc.collect()

xm_train_left45=reconstructed_left45.encoder(X_train_left45)

xm_test_left45=reconstructed_left45.encoder(X_test_left45)

del X_train_left45

del X_test_left45

gc.collect()


xm_train_right45=reconstructed_right45.encoder(X_train_right45)

xm_test_right45=reconstructed_right45.encoder(X_test_right45)

del X_train_right45

del X_test_right45

gc.collect()


"""Fusion of front, left, right, left45, and right45 vectors**"""

class CNN_VIEWS(Model):

    def __init__(self):

        super(CNN_VIEWS, self).__init__()

        self.front = tf.keras.Sequential([

            layers.Input(shape=xm_train_frontal.shape[1:]),

            layers.Conv2D(16, 3, strides=1, padding='same', activation='relu'),

```

```
layers.MaxPooling2D(2, strides=2),

layers.Conv2D(8, 3, strides=1, padding='same', activation='relu'),

layers.MaxPooling2D(2, strides=2),

layers.Conv2D(4, 3, strides=1, padding='same', activation='relu'),

layers.Flatten(),

layers.Dense(400, activation='relu')


self.left = tf.keras.Sequential([

    layers.Input(shape=xm_train_left.shape[1:]),

    layers.Conv2D(8, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(4, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(2, 3, strides=1, padding='same', activation='relu'),

    layers.Flatten(),

    layers.Dense(200, activation='relu')])


self.right = tf.keras.Sequential([

    layers.Input(shape=xm_train_right.shape[1:]),

    layers.Conv2D(8, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(4, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(2, 3, strides=1, padding='same', activation='relu'),

    layers.Flatten(),
```

```

layers.Dense(200,activation='relu'))

self.left45 = tf.keras.Sequential([

    layers.Input(shape=xm_train_left45.shape[1:]),

    layers.Conv2D(3, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(1, 3, strides=1, padding='same', activation='relu'),

    layers.Flatten(),

    layers.Dense(200,activation='relu'))

self.right45 = tf.keras.Sequential([

    layers.Input(shape=xm_train_right45.shape[1:]),

    layers.Conv2D(3, 3, strides=1, padding='same', activation='relu'),

    layers.MaxPooling2D(2, strides=2),

    layers.Conv2D(1, 3, strides=1, padding='same', activation='relu'),

    layers.Flatten(),

    layers.Dense(200,activation='relu'))

self.combine = tf.keras.Sequential([

    layers.Input(shape=(1200,)),

    layers.Dropout(0.2),

    Layers.Dense (600, Activation =
'Relu'), Layers.Dense (300, Activation
= 'Relu'), Layers.Dense (150,
Activation = 'Relu'), Layers.Dense (80,
Activation = 'Relu' ), Layers.Dense
(40, activation = 'reread'), layers.
dese (20, activation = 'reread'),
layers.dese (10, activation = 'relu'),

```

```

        layers.Dense(1, activation='sigmoid'))

def call(self, x):

    front = self.front(x[0])

    left = self.left(x[1])

    right = self.right(x[2])

    left45 = self.left45(x[3])

    right45 = self.right45(x[4])

    out = self.combine(layers.Concatenate(axis=1)([front, left, right, left45, right45]))

    return out

"""## **Deep Learning**"""

epochs = 40

with strategy.scope():

    cnn_combined=CNN_Views()

    cnn_combined.compile(optimizer = tf.keras.optimizers.Adam(),
loss='binary_crossentropy', metrics=tf.keras.metrics.BinaryAccuracy(name='binary_accuracy',
dtype=None, threshold=0.5))

    history=cnn_combined.fit([xm_train_frontal, xm_train_left, xm_train_right, xm_train_left45,
xm_train_right45], y_train_frontal, epochs=epochs,

        batch_size=8, shuffle=True)

    #, validation_data=([xm_test_frontal, xm_test_left, xm_test_right, xm_test_left45,
xm_test_right45], y_test_frontal)

    cnn_combined.evaluate([xm_test_frontal, xm_test_left, xm_test_right, xm_test_left45,
xm_test_right45], y_test_frontal,return_dict=True)

import seaborn as sns

y_pred=cnn_combined.predict([xm_test_frontal, xm_test_left, xm_test_right, xm_test_left45,
xm_test_right45])

y_pred=np.where(y_pred > 0.5, 1,0)

```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

cm=confusion_matrix(y_test_frontal, y_pred)

print(cm)

print(classification_report(y_test_frontal, y_pred))

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['healthy','sick'])

sns.reset_orig()

disp.plot()
```

EC-10: Evrişimli LSTM Ağı Kodu

```
import tensorflow as tf

import os

import tensorflow_datasets as tfd

resolver = tf.distribute.cluster_resolver.TPUClusterResolver()

tf.config.experimental_connect_to_cluster(resolver)

This is the TPU initialization code that has to be at the beginning.

tf.tpu.experimental.initialize_tpu_system(resolver)

print("All devices: ", tf.config.list_logical_devices('\TPU\'))

strategy = tf.distribute.TPUStrategy(resolver)

Commented out IPython magic to ensure Python compatibility.

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

directory_path='/gdrive/My Drive/Yedek/Video_Dataset_hsv'

subset_paths={'train':directory_path+'/train','test':directory_path+'/test'}

!pip install hickle

import hickle as hkl

import random

from pathlib import Path

import os

import cv2

import matplotlib.pyplot as plt

from random import randint

import numpy as np
```

```
import matplotlib.pyplot as plt

def frames_from_video_file(video_path):

    result = []

    src = cv2.VideoCapture(str(video_path))

    for _ in range(0,20,2):

        ret, frame = src.read()

        frame = tf.image.convert_image_dtype(frame, tf.float32)

        tf.image.resize(frame, [456,456])

        result.append(frame)

    src.release()

    result = np.array(result)[..., [2, 1, 0]]

    return result

class FrameGenerator:

    def __init__(self, path, training = False):

        self.path = path

        self.training = training

        self.class_names = sorted(set(p.name for p in self.path.iterdir() if p.is_dir()))

        self.class_ids_for_name = dict((name, idx) for idx, name in enumerate(self.class_names))

    def get_files_and_class_names(self):

        video_paths = list(self.path.glob('*/*.avi'))

        classes = [p.parent.name for p in video_paths]

        return video_paths, classes

    def __call__(self):

        video_paths, classes = self.get_files_and_class_names()

        pairs = list(zip(video_paths, classes))
```



```

if self.training:

    random.shuffle(pairs)

    for path, name in pairs:

        video_frames = frames_from_video_file(path)

        label = self.class_ids_for_name[name] # Encode labels

        yield video_frames, label

train_path = Path(subset_paths['train'])

test_path = Path(subset_paths['test'])

fg = FrameGenerator(train_path, training=True)

print(fg.class_ids_for_name)

output_signature = (tf.TensorSpec(shape = (None, None, None, 3), dtype = tf.float32),

                    tf.TensorSpec(shape = (), dtype = tf.float32))

train_ds = tf.data.Dataset.from_generator(FrameGenerator(train_path, training=True),

                                         output_signature = output_signature)

test_ds = tf.data.Dataset.from_generator(FrameGenerator(test_path, training=False),

                                         output_signature = output_signature)

AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)

train_ds = train_ds.batch(16)

test_ds = test_ds.batch(16)

train_frames, train_labels = next(iter(train_ds))

print(f'Shape of training set of frames: {train_frames.shape}')

print(f'Shape of training labels: {train_labels.shape}')

test_frames, test_labels = next(iter(test_ds))

print(f'Shape of validation set of frames: {test_frames.shape}')

```

```

print(f'Shape of validation labels: {test_labels.shape}')

def build_efficient():

    inputs = tf.keras.layers.Input(shape=(456, 456, 3))

    model = tf.keras.applications.EfficientNetB5(include_top=False, input_tensor=inputs,
weights="imagenet")

    # Freeze the pretrained weights

    model.trainable = False

    # Rebuild top

    model = tf.keras.Sequential([model, tf.keras.layers.GlobalAveragePooling2D(), tf.keras.layers.BatchNormalization()])

    return model

with strategy.scope():

    net = build_efficient()

    model = tf.keras.Sequential([

        tf.keras.layers.Rescaling(scale=255),

        tf.keras.layers.TimeDistributed(net),

        tf.keras.layers.LSTM(24),

        tf.keras.layers.Dropout(0.2),

        tf.keras.layers.Dense(8),

        tf.keras.layers.Dropout(0.2),

        tf.keras.layers.Dense(4),

        tf.keras.layers.Dense(1, 'sigmoid')

    ])

    model.compile(optimizer = tf.keras.optimizers.AdamW(learning_rate=1e-3, use_ema=True,
ema_momentum = 0.8, weight_decay=0.1),

```

```
        loss='mse', metrics=tf.keras.metrics.BinaryAccuracy(name='binary_accuracy', dtype=None,
threshold=0.5))

epochs=20

model.fit(train_ds, epochs = epochs)

def get_actual_labels(dataset):

    actual = [labels for _, labels in dataset.unbatch()]

    actual = tf.stack(actual, axis=0)

    return actual

model.evaluate(test_ds, return_dict=True)

import seaborn as sns

actual = get_actual_labels(test_ds)

y_pred=model.predict(test_ds)

y_pred=np.where(y_pred > 0.5, 1,0)

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

cm = confusion_matrix(actual, y_pred)

print(cm)

print(classification_report(actual, y_pred))

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['healthy', 'sick'])

sns.reset_orig()

disp.plot()
```

AVI Video Conversion Code EK-11

```
pip install opencv-python

from google.colab import drive

drive.mount("/gdrive")

# %cd /gdrive

directory_path = "/gdrive/My Drive/Yedek/Thermal_Image_hsv"

save_path="/gdrive/My Drive/Yedek/Thermal_Video_hsv"

import os

from PIL import Image

import cv2

def create_video(image_path_folder,s_path,video_name):

    codec = cv2.VideoWriter_fourcc(*'DIVX')

    vid_writer = cv2.VideoWriter(s_path+'/'+video_name+'.avi', codec, 5, (640,480))

    print('Creating video...')

    for image in image_path_folder:

        print(image)

        loaded_img = cv2.imread(image)

        vid_writer.write(loaded_img)

    vid_writer.release()

    print(f"video created {video_name}...")

subdirectories=next(os.walk(directory_path))[1]

print(subdirectories)

for subdirectory in subdirectories:

    os.mkdir(save_path+'\\'+subdirectory)

    video_save_path=save_path+'\\'+subdirectory
```

```

subdirectories2=next(os.walk(directory_path+'/'+subdirectory))[1]

for subdirectory2 in subdirectories2:

    images=[]

    files=next(os.walk(directory_path+'/'+subdirectory+'/'+subdirectory2))[2]

    dynamic_count=0

    print("checking folder...")

    if len(files) != 0:

        for file in files:

            print(file)

            number1 = file.split('.')[1]

            number2 = file.split('.')[2]

            if number1 == "1" and number2 == "1":

                print(f"{number1}.{number2}, dynamic front image...")

                dynamic_count += 1

            file_path = directory_path + '/' + subdirectory + '/' + subdirectory2 + '/' + file

            images.append(file_path)

        else:

            print("static side view file...")

    if dynamic_count==20:

        print(f'\there are {dynamic_count} dynamic files, generate video...\')

        create_video(images,video_save_path, subdirectory2)

    else:

        print(f'\there are {dynamic_count} dynamic files, insufficient quantity...\')

else:

    print('empty folder!...')

```