

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING



CLASSIFICATION OF SPERM CELL MORPHOLOGIES
WITH DEEP LEARNING

16011110 – Ömer Lütfü TORTUMLU

SENIOR PROJECT

Advisor
Assist. Prof. Dr. Hamza Osman İLHAN

July, 2020

ACKNOWLEDGEMENTS

In the realization of this work, I will never forget the importance of every word he has used in my life, who shared his valuable knowledge with me for 4 months. I would like to express my endless thanks to Assist. Prof. Dr. Hamza Osman İLHAN

Ömer Lütfü TORTUMLU

TABLE OF CONTENTS

LIST OF ABBREVIATIONS	vi
LIST OF FIGURES	viii
LIST OF TABLES	ix
ABSTRACT	x
ÖZET	xi
1 Introduction	1
2 Preliminary Survey	2
3 Feasibility	4
3.1 Legal Feasibility	4
3.2 Economic Feasibility	4
3.3 Technical Feasibility	4
3.3.1 Hardware Feasibility	4
3.3.2 Software Feasibility	4
3.4 Workforce and Time Schedule	5
4 SYSTEM ANALYSIS	6
4.1 Tensorflow	6
4.2 Keras	6
4.3 CUDA Toolkit	7
4.4 PyQt5 Designer	7
4.5 Dataset Analysis	7
4.5.1 HuSHeM Dataset	7
4.5.2 SMIDS Dataset	7
4.5.3 SCIAN-MORPHO Dataset	8
4.6 Interface Flowchart	9
5 SYSTEM DESIGN	10
5.1 Dataset Design	10

5.2	Data Augmentation	11
5.3	Keras Models	11
5.3.1	Xception [6]	11
5.3.2	VGG16 [7]	11
5.3.3	VGG19 [7]	11
5.3.4	ResNet [8]	12
5.3.5	InceptionV3 [9]	12
5.3.6	InceptionResNetV2 [10]	12
5.3.7	MobileNet [11]	12
5.3.8	MobileNetv2 [12]	12
5.3.9	DenseNet [13]	13
5.3.10	NASNetMobile [14]	13
5.4	Activations	13
5.4.1	Softmax [15]	13
5.4.2	Softplus [15]	13
5.4.3	Softsign [15]	13
5.4.4	Sigmoid [15]	14
5.4.5	Hard Sigmoid [15]	14
5.4.6	Elu [15]	14
5.4.7	Selu [15]	14
5.4.8	Relu [15]	14
5.4.9	Tanh [15]	15
5.4.10	Exponential [15]	15
5.4.11	Linear [15]	15
5.5	Loss Functions	15
5.5.1	Categorical Crossentropy [16]	15
5.5.2	Binary Crossentropy [16]	15
5.5.3	Mean Squared Error [16]	15
5.5.4	Mean Absolute Error [16]	16
5.5.5	Mean Squared Logarithmic Error [16]	16
5.5.6	Squared Hinge [16]	16
5.5.7	Hinge [16]	16
5.5.8	Categorical Hinge [16]	16
5.5.9	Logcosh [16]	16
5.5.10	Sparse Categorical Crossentropy [16]	17
5.5.11	Kullback Leibler Divergence [16]	17
5.5.12	Poisson [16]	17
5.5.13	Cosine Proximity [16]	17
5.6	Optimizers [17]	17

5.6.1	SGD [17]	17
5.6.2	RMSprop [17]	17
5.6.3	Adagrad [17]	18
5.6.4	Adadelata [17]	18
5.6.5	Adam [17]	18
5.6.6	Adamax [17]	18
5.6.7	Nadam [17]	18
6	Application	19
6.1	Global Parameters Section	20
6.2	Model Parameters Section	20
6.3	Data Parameters Section	21
6.4	Data Generator Parameters Section	21
6.5	Training Section	22
6.6	Result Section	22
6.7	Test Section	23
6.7.1	Step 1	23
6.7.2	Step 2	23
6.7.3	Step 3	23
7	Performance Analysis	24
7.1	Augmentation Rate Test	25
7.2	Epoch Test	25
8	Result	26
	References	27
	Curriculum Vitae	29

LIST OF ABBREVIATIONS

API	Application programming Interface
CPU	Central Processing Unit
CNTK	The Microsoft Cognitive Toolkit
CASA	Computer Assisted Semen Analysis
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DNN	Deep Neural Network
ELU	Exponential Linear Unit
FC	Fully Connected
FLOPS	Floating Point Operations Per Second
GPU	Graphics Processing Unit
GPGPU	General Purpose computing on Graphics Processing Units
HuSHeM	Human Sperm Head Morphology
JSON	Java Script Object Notation
MAE	Mean Absolute Error
MSE	Mean Squared Error
RAM	Random Access Memory
RELU	Rectified Linear Unit
ROI	Region Of Interest
KL	Kullback Leibler
NAG	Nesterov Accelerated Gradient
SELU	Scaled Exponential Linear Unit
SGD	Stochastic Gradient Descent

SVM	Support Vector Machine
SMIDS	Sperm Morphology Image Data Set
TPU	Tensor Processing Unit
WHO	World Health Organization

LIST OF FIGURES

Figure 3.1	Workforce and Time Scheduler	5
Figure 4.1	Interface Flowchart	9
Figure 5.1	K-folded File Structure	10
Figure 5.2	Original File Structure	10
Figure 6.1	Application of Interface	19
Figure 6.2	Global Section of Interface	20
Figure 6.3	Model Section of Interface	20
Figure 6.4	Data Parameters Section of Interface	21
Figure 6.5	Data Generator Parameters Section of Interface	21
Figure 6.6	Training Section of Interface	22
Figure 6.7	Result Section of Interface	22
Figure 6.8	Test Section of Interface	23
Figure 7.1	Augmentation Rate Test	25
Figure 7.2	Epoch Test	25

LIST OF TABLES

Table 4.1	Datasets Class Information	8
Table 7.1	MobileNet Perfonmance Analysis	24
Table 7.2	MobileNetV2 Perfonmance Analysis	24

ABSTRACT

CLASSIFICATION OF SPERM CELL MORPHOLOGIES WITH DEEP LEARNING

Ömer Lütfü TORTUMLU

Department of Computer Engineering
Senior Project

Advisor: Assist. Prof. Dr. Hamza Osman İLHAN

Within the scope of this project, a locally working image classification interface has been developed using keras networks. To achieve this goal, interface development, deep learning and transfer learning techniques were used. The interface basically consists of 3 main structures; training the ready keras network, training the weights of the pre-trained model (h5 and JSON) and testing the trained models. With the interface, the original data sets can be randomly created and saved in the desired number. Datasets can be reproduced up to 10 times with tensorflow's data generator methods. Interface MobileNet and MobileNetV2 have been compared with the classification success of the networks tested on 3 datasets.

As a result of this study, it has been observed that the interface works correctly and is user friendly for deep learning studies.

Keywords: image dataset classification, deep learning tool, keras applications, tensorflow, transfer learning, machine learning

SPERM HÜCRE MORFOLOJİLERİNİN DERİN ÖĞRENME İLE SINIFLANDIRILMASI

Ömer Lütfü TORTUMLU

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr.Öğr.Üyesi Hamza Osman İLHAN

Bu proje kapsamında, keras ağları kullanılarak, yerelde çalışan bir resim sınıflandırma arayüzü geliştirilmiştir. Bu amaca ulaşmak için arayüz geliştirme, derin öğrenme ve transfer öğrenme tekniklerinden yararlanılmıştır. Arayüz temelde 3 ana yapıdan oluşmaktadır; hazır keras ağı eğitme, önceden eğitilmiş modelin ağırlıklarını(h5 ve JSON) yükleyerek eğitme ve eğitilmiş modelleri test etme. Arayüz ile orjinal veri setleri istenen sayıda rastgele oluşturulup kaydedilebilir. Verisetleri tensorflowun veri üretici metodlarıyla 10 katına kadar çoğaltılabilir. Arayüz MobileNet ve MobileNetV2 ile 3 veriseti üzerinde test edilmiş ağların arayüzde sınıflandırma başarıları karşılaştırılmıştır.

Bu çalışma sonunda arayüzün derin öğrenme çalışmaları için doğru çalıştığı ve kullanıcı dostu olduğu gözlemlenmiştir.

Anahtar Kelimeler: görüntü veriseti sınıflandırma, derin öğrenme aracı, keras ağları, tensorflow, derin öğrenme, makine öğrenmesi

1

Introduction

Diagnosis of infertility in men is made by examining sperm cells in hospitals. This examination looks at the number, shape, and movement of sperm cells. These examinations, which are referred to as morphological analysis, are done manually with the help of a microscope. High margin of error in these analyses and objectivity of the results are discussed. In this study, morphology of sperm cells using two different datasets is intended. It is planned to design an advanced interface for training and testing at the local area for this review. Our datasets are painted with different painting techniques and it is aimed to make morphological analysis autonomously by training these pictures with deep learning. So faster and more objective results can be offered. There are previous studies in this area. Our aim success in these studies is not to pass, but different To develop an interface for testing networks and to determine a common network with acceptable performance in 3 datasets.

2 Preliminary Survey

Infertility is prevented from having a child after 1 year of regular sexual intercourse. According to WHO reports, 15% - 20% of all couples in the world suffer from some kind of infertility problem. The main reason for these problems is;

It is classified as

- i) male,
- ii) female,
- iii) unexplained and
- iv) double-bottomed.

Semen sample analysis, also known as spermiogram, is the most popular test in the diagnosis of infertility to observe problems related to the male factor. Performed in two steps

- i) The semen sample given is evaluated in terms of physical appearance such as viscosity, color and odor.
- ii) A specialist measures sperm characteristics in terms of morphology, concentration and motility parameters using computerized or manual evaluation techniques. Sperm morphology analysis focuses on dimensional evaluation for sperm head, middle part and tail. Abnormalities have a direct effect on infertility.

In the first edition published in 1980, the reference rate for normal sperm density over the entire sperm concentration was defined as 80%, but was later updated to 4% in the 2010 edition. Then, the sample is observed by a specialist in the manual evaluation step, also known as the visual evaluation technique, which is cheaper and more practical than computer-based systems. For this reason, many laboratories still perform the tests with a visual assessment technique. Alternatively,

tests can be performed on computer-based systems that isolate human factors. Such computer-based systems are known as CASA systems. Compared to visual analysis, the CASA system is more reliable, consistent and objective. In addition, considering morphological analysis, CASA systems require significant improvements in many respects.

Computer-based analysis systems basically consist of two consecutive steps. Sperm detection in a specific semen sample is a segmentation process and is the first step of the analysis. Then, each ROI patch should be assigned to a class by classification and machine learning techniques. There are different studies in the literature under these two main titles.

In CNNs, the raw pixels of fragmented patches are fed to the first level of a DNN, and the outputs of this layer can be interpreted as representing the presence of different low-level features such as lines and edges in possible sperm images. In the later layers of these CNNs, sperms are combined at a measure of the presence of higher-level features such as basic shapes (head, a sperm tail, or a noise component) that are then combined into shape clusters. And finally, using all this information, DNNs can provide the possibility that these top-level properties contain a particular object. Classification performances of all traditional feature extraction and traditional machine learning approaches and end-to-end DNN-based approaches are presented and the highest accuracy Ownership workflow with the lowest computing cost is chosen as the proposed hybrid model.

3.1 Legal Feasibility

The licenses of Python[1], Visual Studio Code[2], Keras[3] and Tensorflow[4], which are planned to be used within the scope of the project, have been reviewed and approved for use within the scope of the project.

3.2 Economic Feasibility

The project does not have an economic expense.

3.3 Technical Feasibility

3.3.1 Hardware Feasibility

The Keras[3] library, which is planned to be used within the scope of the project, does not need high GPU power for learning operations. My need for my work was met by my consultant.

System features of the computer where the project will be realized.

Processor: Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz

RAM : 16 GB

Display card : NVIDIA GeForce GTX 950M

3.3.2 Software Feasibility

The project will be implemented on Windows 10. The project will be written in Python[1] language (Python 3.7) and Tensorflow[4] (GPU), Keras[3] libraries will be used. Keras is an open source library for artificial neural networks written in Python.

The interface will be developed in Python using the PyQt5 Designer[5] library. Studies will be realized on this interface.

3.4 Workforce and Time Schedule

Within the scope of the project, first the interface was designed and then it was planned to work on datasets obtained by using various preprocesses and synthetic data generation methods. Time planning is shown in Figure 3.1 below.

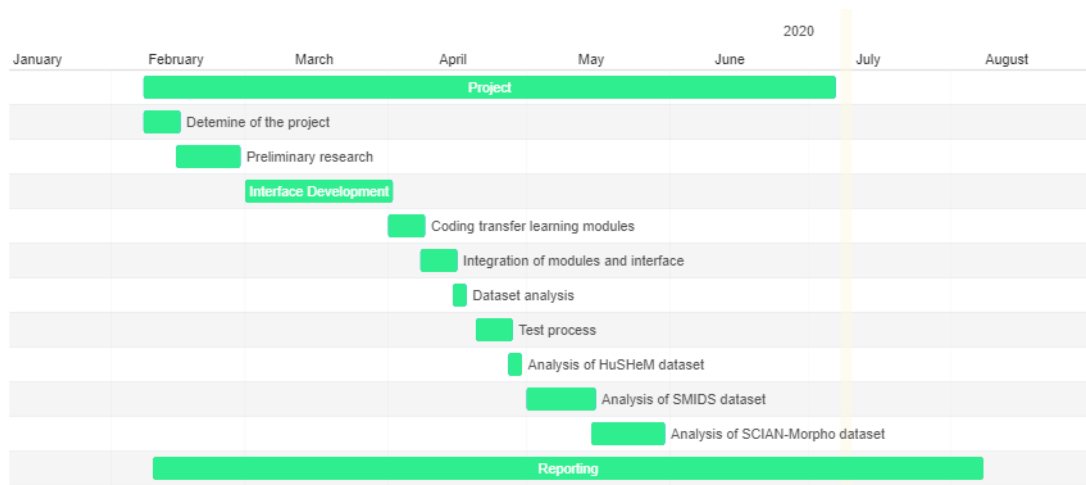


Figure 3.1 Workforce and Time Scheduler

4

SYSTEM ANALYSIS

Deep learning methods will be used to classify the data in my site, which is planned to be implemented. During the project, various deep learning methods are tested and it is planned to find methods with high success rates for HuSHeM, SMIDS and SCIAN-Morpho datasets.

Within the scope of the project, it is aimed to find a more general method that is not only successful for a single dataset, but also for both datasets. The methods planned to be used in the project will be implemented with the keras library running over Python.

4.1 Tensorflow

Tensorflow is a library developed by google for machine learning. Google uses this library in most technologies. As an example, we use this library when searching by voice or image in the search engine. There are many versions of the library. 1.14.0 version is used in the interface.

4.2 Keras

Keras is a deep learning framework for Python that provides a convenient way to define and train almost any type of deep learning model. Keras is a high-level neural network API written in Python, which can run on Tensorflow, Theano and CNTK. It has been developed for fast experiments. Keras is a learning framework for Python used to train the model with deep learning and transfer learning. Keras models are developed for fast experiments.

Keras has the following features :

- i)Allows for easy and fast prototyping.
- ii)Run seamlessly on CPU and GPU.

iii) Supports both convolutional networks (for computer vision) and recurrent networks (for sequence and time-series), as well as the combination of two.

iv) It supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing and so on. This means Keras is appropriate for building deep learning models, from generative adversarial networks to a neural Turing machine.

Keras is compatible with versions of Python from 2.7 to 3.6 till date.

4.3 CUDA Toolkit

CUDA was developed by Nvidia for programmers to use for parallel programming. There is a cuDNN library developed for deep learning networks. In order for CUDA to work actively, graphics cards must be compatible with its versions. CUDA 10.0 Toolkit and 7.4 cuDNN versions are used in the interface.

4.4 PyQt5 Designer

PyQt5 comes with a simple designer application for us to design our application. After designing the front face with PyQt5 Designer, the front face is converted from the command line to the python code, and the necessary codes for the button and other interface elements are coding.

4.5 Dataset Analysis

During this study, studies on HuSHeM, SMIDS and SCIAN-Morpho datasets will be carried out. Datasets were previously divided into manual classes. Below are table 4.1 datasets contents.

4.5.1 HuSHeM Dataset

The HuSHeM (Human Sperm Head Morphology) dataset is in a 4-class structure. classes: normal, tapered, pyriform and amorphous.

4.5.2 SMIDS Dataset

The SMIDS (Sperm Morphology Image Data Set) dataset has a 3 class structure. These classes are divided into: normal sperm, abnormal sperm and non-sperm.

4.5.3 SCIAN-MORPHO Dataset

SCIAN-MORPHO data set was used within the scope of the project. This dataset is classified according to the shape of the sperm. The data set has a 5-class structure consisting of normal, tapered, pyriform, small and amorphous sperm. Images are generally low resolution and gray scales with dimensions such as 33x33, 34x34, 35x35.

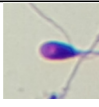
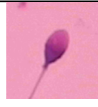
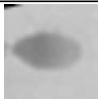


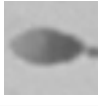

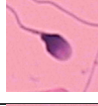

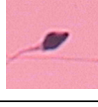

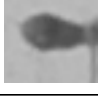
SMIDS			HuSHeM			SCIAN-MORPHO		
Normal	1021		Normal	54		Normal	100	
Abnormal	1005		Tapered	53		Tapered	228	
Non-Sperm	974		Pyriform	57		Pyriform	76	
			Amorphus	52		Small	72	
						Amorphus	656	
Total	3000		Total	216		Total	1132	

Table 4.1 Datasets Class Information

4.6 Interface Flowchart

The interface usage scheme is in figure 4.1 below.

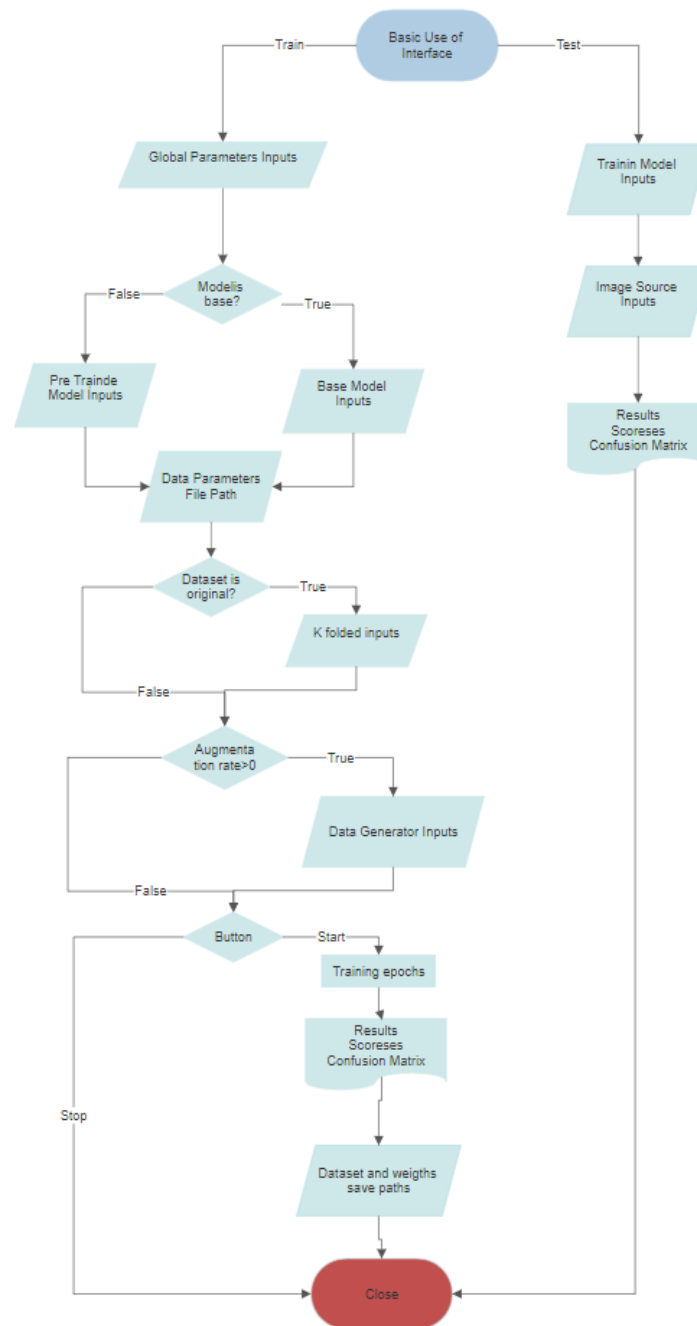


Figure 4.1 Interface Flowchart

5.1 Dataset Design

2 types of datasets can be loaded on the interface as original or k-folded. Original loaded datasets can be converted to k-fold structure in the interface. For this, the original dataset should be as in the Figure 5.1 of the file structure.

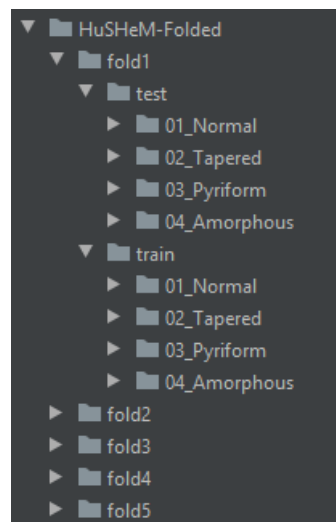


Figure 5.1 K-folded File Structure

The K-folded dataset structure should be as shown in the Figure 5.2.

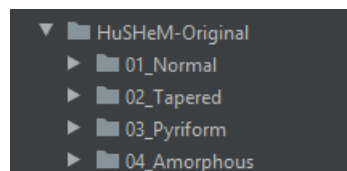


Figure 5.2 Original File Structure

5.2 Data Augmentation

Increasing the data by exposure to various distortion effects in order to increase the performance especially in small data sets. In this way, the model is provided to learn different conditions. The methods used in the interface are listed below;

- Rotation Range
- Height Shift Range
- Width Shift Range
- Zoom Range
- Shear Range
- Horizontal Flip
- Fill Mode

5.3 Keras Models

5.3.1 Xception [6]

Developed by Francois Chollet. Xception is a similar extension of the inception architecture. The image input size of the network is 224 x 224.

5.3.2 VGG16 [7]

It is a simple network model. The most important difference from previous models is the use of convolutional additions with 2 or 3. Softmax performance of 1000 classes is calculated on the output of two Fully Connected layers. Approximately 138 million parameters have been calculated. As with other models, the number of channels increases, while the height and width dimensions of the matrices from input to output decrease. The image input size of the network is 224 x 224.

5.3.3 VGG19 [7]

VGG-19 is a 19 fold deep curved neural network. You can download a pre-prepared version of the network trained on more than a million images from the ImageNet database. The network has learned rich feature representations for a wide variety of images. The image input size of the network is 224x224.

5.3.4 ResNet [8]

ResNet 50, 101 and 152 deep curved neural network models. Networks learned feature-rich presentations for a wide variety of images. The 2nd version of the networks was created as 50V2, 101V2 and 152V2. The image input size of the network is 224x224.

5.3.5 InceptionV3 [9]

InceptionV3 tried to find a solution to the correct core size selection problem caused by diversity in datasets. As a solution to this problem, it has adopted the approach of combining outputs with multiple different size liters in the same layer. The default input size for this model is 299x299.

5.3.6 InceptionResNetV2 [10]

They are very well performing architectures with relatively low computing cost for both Initial and Residual networks. Inception-ResNet combines two architectures to further improve performance. The default input size for this model is 299x299.

5.3.7 MobileNet [11]

MobileNet is a more suitable model for mobile applications that do not have computing power. It uses in-depth removable evolutions with normal folds of the same depth compared to other networks. The number of parameters has been reduced with the evolution used. The model consists of 91 layers. The default input size for this model is 224x224.

5.3.8 MobileNetv2 [12]

Suitable for mobile devices or any device with low computing power, the complexity cost and model size of the network are significantly reduced. In MobileNetV2, a better module is introduced with the reverse residual structure. Non-linear states in narrow layers are eliminated this time. State-of-the-art performances for object detection and semantic segmentation are achieved with MobileNetV2 as backbone for feature extraction. It runs slower than MobileNetV1. The default input size for this model is 224x224.

5.3.9 DenseNet [13]

DenseNet tries to eliminate color transition using a different approach. Instead of using shortcut links, all previous feature maps become the input of the next layer. There are 121, 169 and 201 layered versions. As the number of layers increases, the number of parameters increases and the duration of training increases. The default input size for this model is 224x224.

5.3.10 NASNetMobile [14]

Neural Architecture Search Network NASNet is developed by Google. The developers propose to look for an architectural building block in a small dataset and then transfer the block to a larger dataset. Finally, the NASNet model achieves the most advanced results with smaller model size and lower complexity (FLOPs). NasNetMobile for mobile applications and NasNetLarge for advanced applications have been developed to improve optimization.

5.4 Activations

The activation function is an operational "gate" between the input that feeds the current neuron and its output to the next layer. They decide whether the neuron is activated or not.

5.4.1 Softmax [15]

Some vector components can be negative or greater than one; After applying softmax, each component will have values in the range (0,1).

5.4.2 Softplus [15]

Input tensor: x

Returns: The softplus activation: $\log(\exp(x) + 1)$.

5.4.3 Softsign [15]

Input tensor: x

Returns: The softsign activation: $x / (\text{abs}(x) + 1)$.

5.4.4 Sigmoid [15]

Input tensor: x

Returns: The sigmoid activation: $1 / (1 + \exp(-x))$.

5.4.5 Hard Sigmoid [15]

Hard sigmoid is faster than sigmoid.

Input tensor: x

Returns:

- 0 if $x < -2.5$
- 1 if $x > 2.5$
- $0.2 * x + 0.5$ if $-2.5 \leq x \leq 2.5$.

5.4.6 Elu [15]

Exponential linear unit.

Input tensor: x

α : A scalar, slope of negative section.

Returns: x if $x > 0$ and $\alpha * (\exp(x)-1)$ if $x < 0$.

5.4.7 Selu [15]

Scaled Exponential Linear Unit. Input tensor: x

Returns: Selu: $\text{scale} * \text{elu}(x, \alpha)$.

5.4.8 Relu [15]

Rectified Linear Unit. With default values, it returns element-wise $\max(x, 0)$.

Otherwise, it follows:

$f(x) = \text{maxValue}$ for $x \geq \text{maxValue}$,

$f(x) = x$ for $\text{threshold} \leq x < \text{maxValue}$,

$f(x) = \alpha * (x - \text{threshold})$ otherwise.

Input tensor: x

α : float. Slope of the negative part. Defaults to zero.

maxValue : float. Saturation threshold.

threshold : float. Threshold value for thresholded activation.

Returns: A tensor.

5.4.9 Tanh [15]

Hyperbolic tangent activation function.

Input tensor: x

Returns: $\tanh(x) = (\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x))$

5.4.10 Exponential [15]

Input tensor: x

Returns: $\exp(x)$

5.4.11 Linear [15]

Linear activation function.

x : Input tensor.

Returns: Input tensor, unchanged.

5.5 Loss Functions

When you try your algorithm to try and develop your model, your lost function will tell you if you have reached anywhere. You should use the appropriate loss function to reach the highest result according to the dataset you use.

5.5.1 Categorical Crossentropy [16]

Categorical cross entropy is used in multi-classification. In this classification structure, classes are represented by 0 and 1.

5.5.2 Binary Crossentropy [16]

Binary Cross-Entropy Loss sigmoid activation plus Cross Entropy loss. It is independent for each vector component, meaning that the loss calculated for each CNN output vector is not affected by other component values.

5.5.3 Mean Squared Error [16]

Mean Squared Error is the working area of basic loss functions, it works pretty well. The predicted and essential real values are not checked, the result is always positive, the best value is 0.

5.5.4 Mean Absolute Error [16]

Mean Absolute Error (MAE) is only slightly different in definition from the MSE, but interestingly provides almost exactly opposite properties. To calculate the MAE, you take the difference between your model's predictions and the ground truth, apply the absolute value to that difference, and then average it out across the whole dataset.

5.5.5 Mean Squared Logarithmic Error [16]

There may be regression problems in which the target value has a spread of values and when predicting a large value, you may not want to punish a model as heavily as mean squared error.

Instead, you can first calculate the natural logarithm of each of the predicted values, then calculate the mean squared error. This is called the Mean Squared Logarithmic Error loss.

5.5.6 Squared Hinge [16]

It is simply called square hinge loss, which calculates the square of the score hinge loss. It softens the surface of the error function and facilitates numerical operation. Using a hinge loss provides better performance in a particular binary classification problem.

5.5.7 Hinge [16]

The hinge loss function encourages samples to have the correct mark and gives more errors when there is a difference in the mark between the actual and predicted class values.

5.5.8 Categorical Hinge [16]

$[-1, 0, 1]$ is a valid target range for your activation function. Generally, Keras models do not work well with the classification in a binary output. Instead, it is recommended to use three single hot vectors with softmax classifiers.

5.5.9 Logcosh [16]

Logcosh works mostly like an average square error, it is not affected much by its false prediction.

5.5.10 Sparse Categorical Crossentropy [16]

This loss function is almost similar to Categorical Crossentropy, except for a change. When using the Sparse Categorical Crossentropy function, we do not need to make a hot coding of the target vector. If the target image is a person, you pass 0, otherwise 1.

5.5.11 Kullback Leibler Divergence [16]

Kullback Leibler Divergence is a measure of how a probability distribution differs from a baseline distribution.

Calculates how much information will be lost if the estimated distribution is used to estimate the desired target probability distribution.

5.5.12 Poisson [16]

The Poisson loss function is a measure of how the estimated distribution differs from the expected distribution.

5.5.13 Cosine Proximity [16]

The higher cosine similarity shows higher accuracy. Opposite vectors have a -1 cosine similarity.

5.6 Optimizers [17]

An optimizer is one of the two arguments required for compiling a Keras model.

5.6.1 SGD [17]

Stochastic Gradient Descent is one of the simplest optimization algorithms. All parameters use only one static learning rate during the training phase. As optimizers approach the optimal value, they reduce gradients.

5.6.2 RMSprop [17]

AdaGrad sometimes has a problem where learning rates decrease and result in a long training period. Root Mean Square Propagation tries to solve this problem by reducing the learning rate. RMSProp uses momentum with past learning rates for updating.

5.6.3 Adagrad [17]

AdaGrad has a different learning rate for each parameter in the neural network. Since all parameters in a network are not equally important, they update them differently. Frequently updated parameters are trained very carefully with a low learning rate. These updates can cause a corruption that frustrates the usefulness of the parameters.

5.6.4 Adadelata [17]

Adadelata is an extension of Adagrad that tries to reduce the declining learning rate. It does not accumulate all the square gradients in the past, it limits the fixed size of the gradients to w .

5.6.5 Adam [17]

Man is the method that calculates adaptive learning rates for each parameter. Adaptive moment estimation, Adam also uses past learning rates. However, Adam also uses past gradients to speed learning.

5.6.6 Adamax [17]

Adamax, a type of man, is an optimization method with a gradient extension. It is suitable for learning speech data with noise conditions as it can adjust the learning speed according to the data characteristics.

5.6.7 Nadam [17]

The man can be called a combination of RMSprop and momentum. RMSprop contributes to the exponentially decreasing average of past square gradients. Momentum, on the other hand, explains the exponentially decreasing average of past gradients.

6

Application

The interface consists of two parts as training and testing. You can use Keras models or the model you created in the education section. You can change the structure of your dataset and duplicate its data. If data replication will not take place, you can blank the Data Generator section. You have to press start to start training. The output of the epoch information will be written on the screen during the training. When the training is over, you can review the details from the results section and record the successful dataset, models and weights. You can experiment with different test data by loading the model and weight you recorded in the test section. The main page of the interface is in figure 6.1 below.

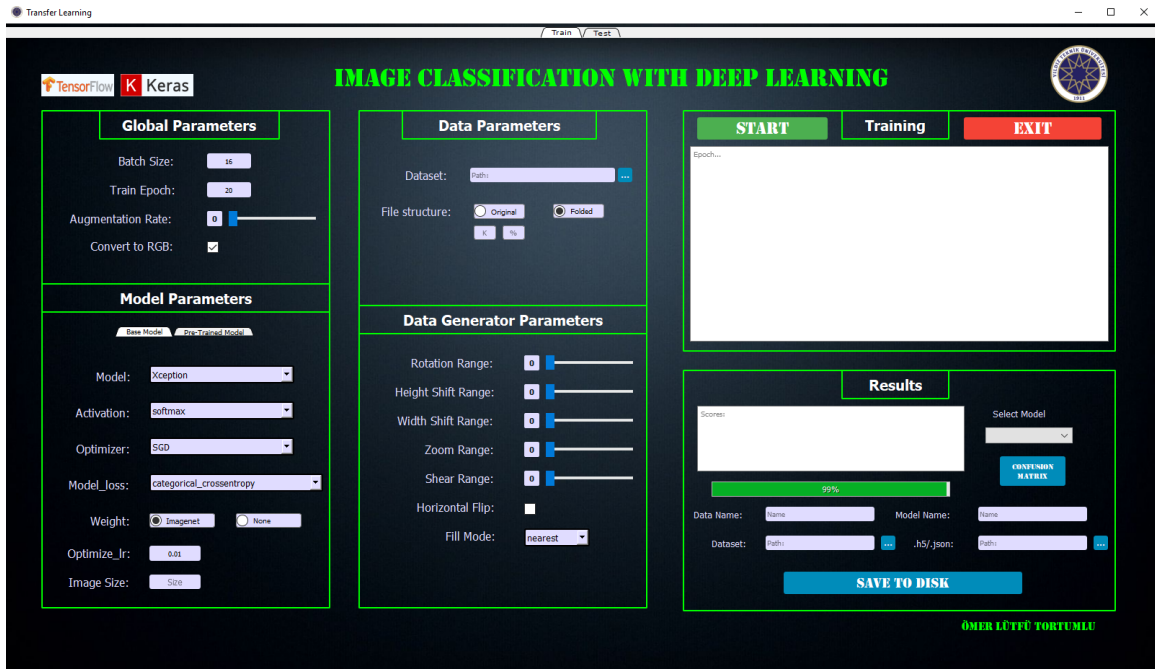
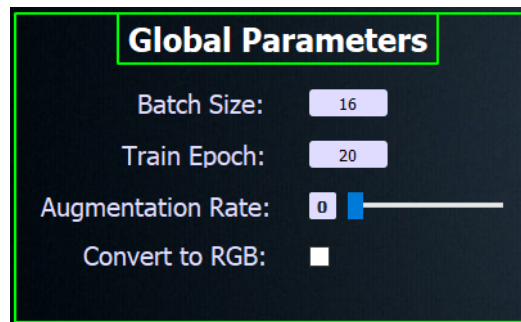


Figure 6.1 Application of Interface

6.1 Global Parameters Section

In this section, the necessary values are taken from the user before compiling the model. If the dataset is to be replicated, it can be adjusted from the augmentation rate value. Convert to RGB content should be marked for black and white images. The global section interface is in figure 6.2 below.

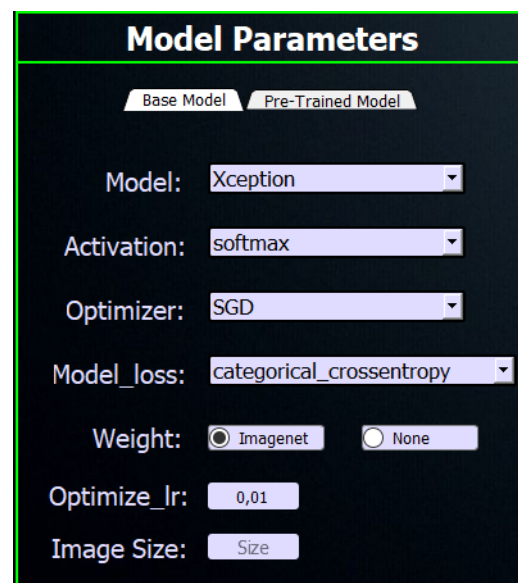


The figure shows a dark-themed interface titled "Global Parameters" in a yellow box. Below the title, there are four settings: "Batch Size" with a value of 16, "Train Epoch" with a value of 20, "Augmentation Rate" with a slider set to 0, and "Convert to RGB" with an unchecked checkbox.

Figure 6.2 Global Section of Interface

6.2 Model Parameters Section

In this section, there are 2 methods to select the model, using basic and pre-recorded weights. The model is created by selecting one of the models defined in Keras in the basic model. To use the pre-trained method, h5 and json files must be loaded. Since the dataset will take the input values of the previously trained model, this method may decrease success in those with high data quality. The model section interface is in figure 6.3 below.



The figure shows a dark-themed interface titled "Model Parameters" in a yellow box. Below the title, there are two tabs: "Base Model" and "Pre-Trained Model". Under the "Base Model" tab, there are five settings: "Model" (Xception), "Activation" (softmax), "Optimizer" (SGD), "Model_loss" (categorical_crossentropy), and "Weight" (Imagenet selected, None unselected). Below these are "Optimize_lr" (0,01) and "Image Size" (Size).

Figure 6.3 Model Section of Interface

6.3 Data Parameters Section

In this section, dataset file path and file structure are selected. For the original dataset, training can be realized with different variations with the k-fold method. The data parameters section of interface is in figure 6.4 below.

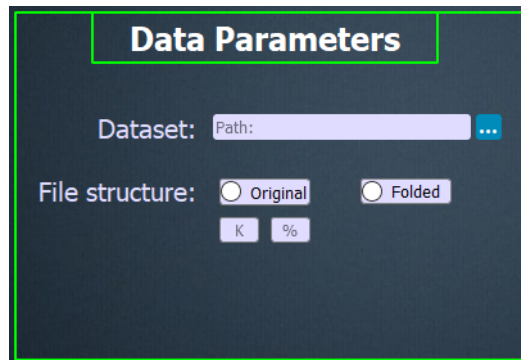


Figure 6.4 Data Parameters Section of Interface

6.4 Data Generator Parameters Section

The data augmentation value of this section should be used at values other than zero. Data augmentation is performed according to the entered values. The data generator parameters section of interface is in figure 6.5 below.

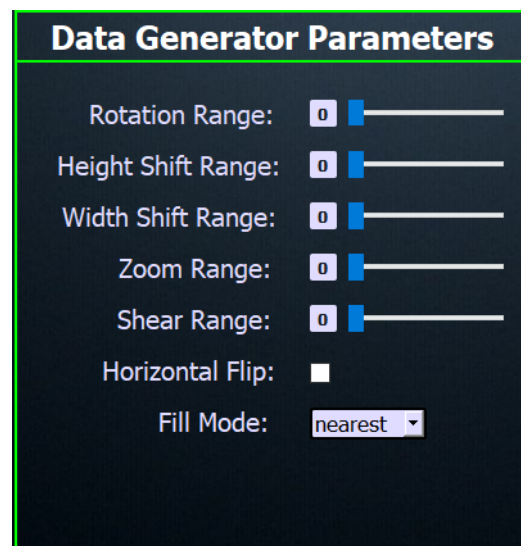


Figure 6.5 Data Generator Parameters Section of Interface

6.5 Training Section

The ready model and dataset are trained in this section and epoch outputs are displayed on the screen. There is an opportunity to discontinue training for bad training. The training section of interface is in figure 6.6 below.

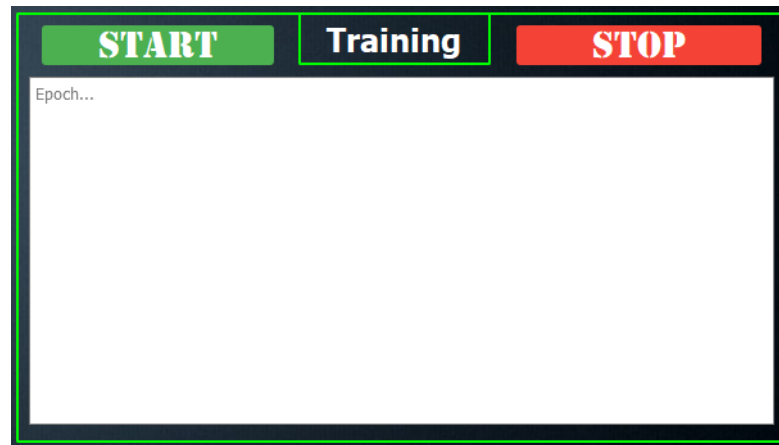


Figure 6.6 Training Section of Interface

6.6 Result Section

This section provides details of the training results. You can also review the results via the confusion matrix. You can save the K-folded dataset and model weights where you set them on the disc. The result section of interface is in figure 6.7 below.

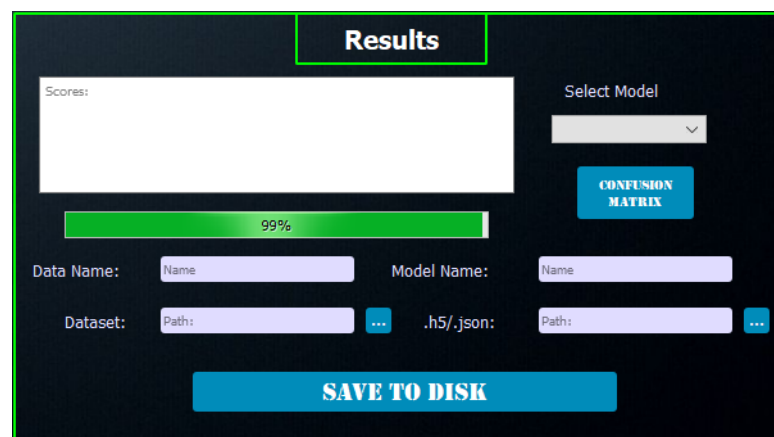


Figure 6.7 Result Section of Interface

6.7 Test Section

In this section, you can test the pre-trained model weights by loading them. For the correct classification of models that train with file structure, they should be the same as the file structure used in training. The test section of interface is in figure 6.8 below.

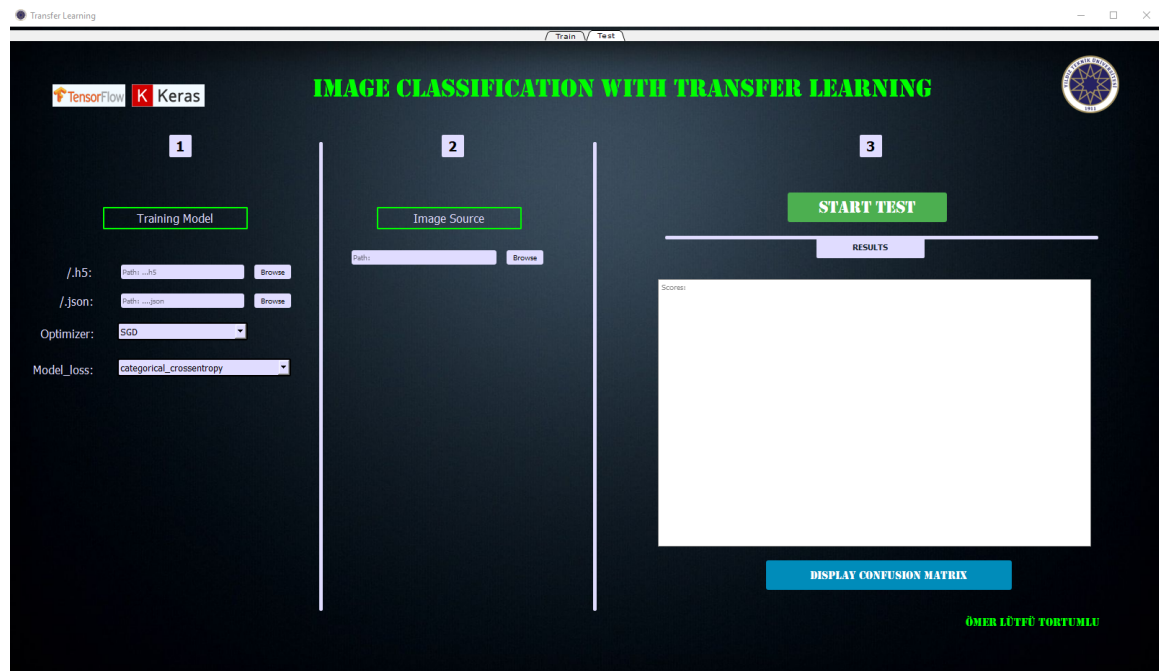


Figure 6.8 Test Section of Interface

6.7.1 Step 1

In this step, the path of the model's h5 and json files must be entered to compile the model. Once the optimizer and loss function are received, the model is ready for testing.

6.7.2 Step 2

In this step, the file path of the test set is entered. Gray scale images must be RGB.

6.7.3 Step 3

You can get the results by pressing the button to start the test.

7 Performance Analysis

Epoch and augmentation rate tests were performed on the HuSHeM, SMIDS and SCIAN-Morpho databases with MobileNet and MobileNetv2 keras networks, which have few layers, to measure interface performance and keep test time short.

The most optimized results in the system in these tests have been achieved by using (softplus, adamax, lr=0.0001, batch size=8) values. It has been observed that MobileNetV2 generally produces bad results compared to MobileNet. In MobileNetV2 tests, it was observed that as the number of epoch and augmantation rate increased, it approached MobileNet.

	MobileNet								
	HuSHeM			SMIDS			SCIAN		
Aug/Epo	10	30	50	10	30	50	10	30	50
1x	45	54	59	84	84	84	57	58	58
5x	69	72	74	86	88	88	62	66	64
10x	75	71	75	85	86	87	66	65	67

Table 7.1 MobileNet Perfomance Analysis

	MobileNetV2								
	HuSHeM			SMIDS			SCIAN		
Aug/Epo	10	30	50	10	30	50	10	30	50
1x	38	44	39	64	85	84	48	54	59
5x	62	73	75	85	87	87	54	62	62
10x	72	77	77	86	86	87	65	66	66

Table 7.2 MobileNetV2 Perfomance Analysis

7.1 Augmentation Rate Test

The augmentation rate values at 50 epoch are shown in figure 7.1 below.

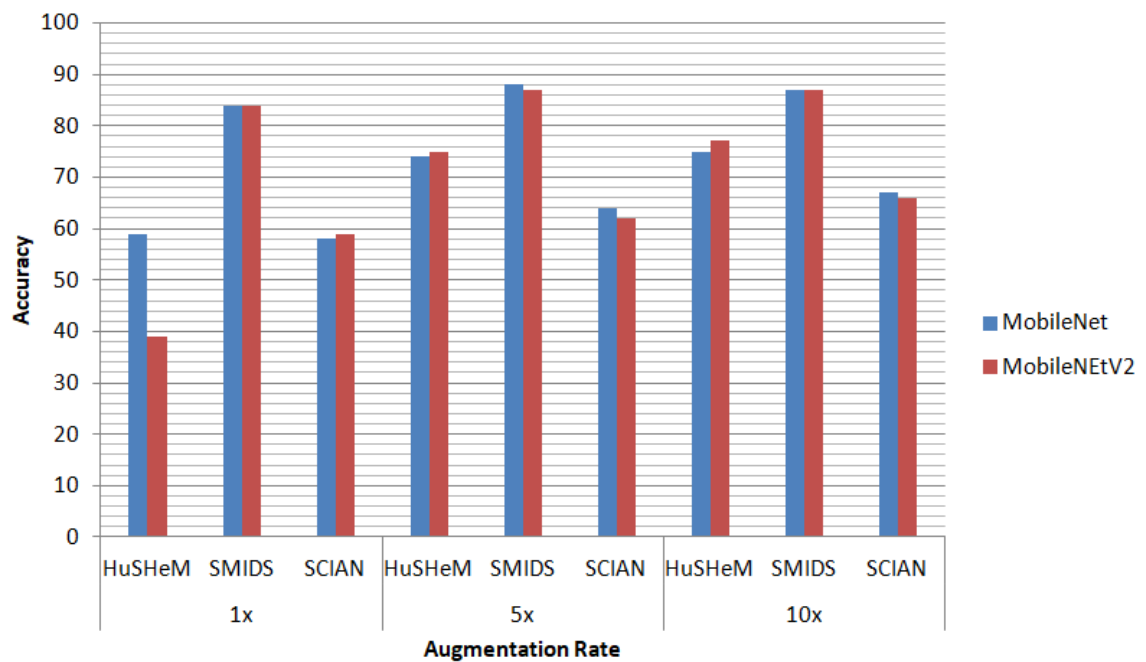


Figure 7.1 Augmentation Rate Test

7.2 Epoch Test

Epoch values at 10x augmentation rate are shown in figure 7.2 below.

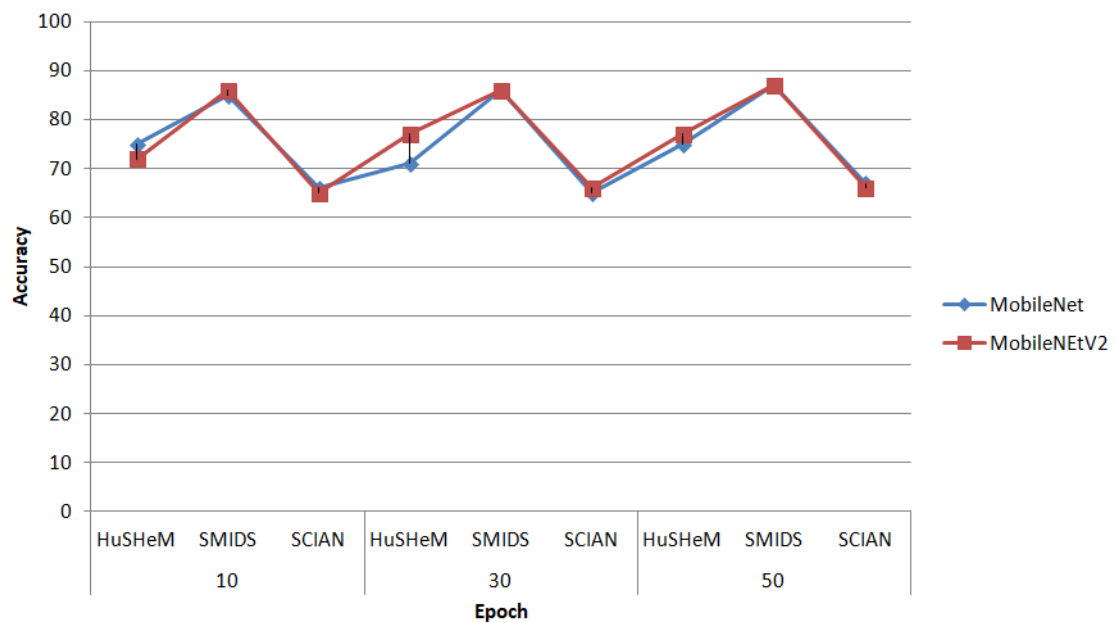


Figure 7.2 Epoch Test

8 Result

Keras networks is a library developed for quick trials. With the developed interface, it has been observed that it can make faster experiments and optimize the problems in the fastest way. We can record the trained model and test it at any time and train it with other data.

References

- [1] P. S. Foundation. (). Python is a programming language, [Online]. Available: <https://www.python.org/> (visited on 04/19/2020).
- [2] M. Corporation. (). The python ide for professional developers by visual studio code, [Online]. Available: <https://code.visualstudio.com/> (visited on 04/19/2020).
- [3] K. Team. (). The python deep learning library, [Online]. Available: <https://keras.io/> (visited on 04/19/2020).
- [4] T. team. (). TensorFlow: Large-scale machine learning on heterogeneous systems, [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/ (visited on 04/19/2020).
- [5] P. Thompson. (). Create simple gui applications with python, [Online]. Available: <https://pypi.org/project/PyQt5Designer/> (visited on 04/19/2020).
- [6] F. Chollet. (2017). Xception: Deep learning with depthwise separable convolutions, [Online]. Available: <https://arxiv.org/pdf/1610.02357v3.pdf> (visited on 11/04/2019).
- [7] K. Simonyan and A. Zisserman. (2015). Very deep convolutional networks for large-scale image recognition, [Online]. Available: <https://arxiv.org/pdf/1409.1556.pdf> (visited on 04/19/2020).
- [8] M. Research. (2015). Deep residual learning for image recognition, [Online]. Available: <https://arxiv.org/pdf/1512.03385.pdf> (visited on 04/19/2020).
- [9] Google. (2014). Going deeper with convolutions, [Online]. Available: <https://arxiv.org/pdf/1409.4842v1.pdf> (visited on 11/04/2019).
- [10] V. V. Christian Szegedy Sergey Ioffe and A. Alemi. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning, [Online]. Available: <https://arxiv.org/pdf/1602.07261.pdf> (visited on 04/19/2020).
- [11] G. Inc. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications, [Online]. Available: <https://arxiv.org/pdf/1704.04861.pdf> (visited on 11/04/2019).
- [12] —, (2019). Mobilenetv2: Inverted residuals and linear bottlenecks, [Online]. Available: <https://arxiv.org/pdf/1801.04381.pdf> (visited on 04/19/2020).

- [13] Z. L. Gao Huang and L. van der Maaten. (2018). Densely connected convolutional networks, [Online]. Available: <https://arxiv.org/pdf/1608.06993.pdf> (visited on 04/19/2020).
- [14] G. Brain. (2018). Learning transferable architectures for scalable image recognition, [Online]. Available: <https://arxiv.org/pdf/1707.07012.pdf> (visited on 04/19/2020).
- [15] Keras. (), [Online]. Available: <https://keras.io/activations/> (visited on 04/19/2020).
- [16] —, (), [Online]. Available: <https://keras.io/losses/> (visited on 04/19/2020).
- [17] —, (), [Online]. Available: <https://keras.io/optimizers/> (visited on 04/19/2020).

Curriculum Vitae

FIRST MEMBER

Name-Surname: Ömer Lütfü TORTUMLU

Birthdate and Place of Birth: 08.09.1997, Erzurum

E-mail: omertortumlu@gmail.com

Phone: 0535 877 3989

Practical Training: Uyumsoft Uyumakademi

Project System Informations

System and Software: Windows İşletim Sistemi, Python

Required RAM: 8GB

Required Disk: 120GB