Software Engineering Department

Otoscope output image analysis

Capstone Project Phase B

Software Engineering (Course 61999)

Karmiel – Jan 2023

Author

Shlomi Levi 305129652

Supervisor

Mrs. Natali Levi

Table of content

Abstract

1. Introduction.....3

2. Background.....4

2.1 Artificial Neural Networks.....4

2.2 Machine Learning 4

2.3 Deep learning.....4

2.4 Activation Function.....4

2.4.1 Sigmoid 5

2.4.2 ReLU.....5

2.5 Keras layers API.....6

2.5.1 Dense layer.....6

2.5.2 Flatten layer.....6

2.5.3 Dropout layer.....6

2.5.4 Batch normalization layer 6

2.5.5 Softmax layer.....7

2.5.6 Classification layer.....7

2.6 ResNet 7

2.6.1 ResNet-101.....7

2.6.2 ResNet-50.....8

2.7 Vgg-16.....8

3. Methodology.....9

4. Dataset.....10

5. Feature extraction.....10

6. Challenges.....11

7. Results.....12

8. Statistical analysis.....21

9. UML's.....22

10. Testing place.....23

11. Developer manual.....24

12. User manual.....27

13. Conclusions 30

14. Environment 30

15. References 31

Abstract.

Acute infections of the middle ear are the most commonly treated childhood diseases[6]. Because complications affect children's language learning and cognitive processes, it is essential to diagnose these diseases in a timely and accurate manner. The prevailing literature suggests that it is difficult to accurately diagnose these infections, even for experienced ear physicians[7]. Advanced care practitioners [8](e.g., nurse practitioners, and physician assistants) serve as first-line providers in many primary care settings and may benefit from additional guidance to appropriately determine the diagnosis and treatment of ear diseases [2][5]. For this purpose, I designed a content-based image retrieval (CBIR) system (called Otoscope output image analysis) for normal and middle ear effusion conditions. Operating on eardrum images captured with an external device used as an otoscope. I present a method that enables the conversion of any convolutional neural network (trained for classification) into an image retrieval model. As a proof of concept, I converted a pre-trained deep learning model into an image retrieval system. I accomplished this using a dataset of 338 eardrum images (169 normal and 169 effusion cases) used to train and test the system. The proposed method resulted in an average accuracy of 95%. These are promising results to demonstrate the feasibility of developing a CBIR system for eardrum images using the newly proposed methodology.

Introduction

In this project, I have chosen to meet a very primary need related to human health. Through development, I can minimize to the point of preventing daily functioning that is impaired due to a medical condition in the ear.

My software will process an image using a device connected to a mobile device, similar to the operation of an otoscope, and analyze whether a person suffers from an ear infection. In fact, the main change that my system creates is that it automatically captures, analyzes, and diagnoses ear infections.

The motivation is to address an engineering problem in the field of medicine. As part of the goals, I want to save the time required in examining the medical condition while using minimal manpower from the medical side, obtaining an immediate and convenient diagnosis without having to visually meet with a doctor.

Today, the examination of the ears and their treatment are performed in various medical centers, and the patient arrives on their own or needs assistance. The test is performed by a medical professional.

Among the devices offered for sale, there are portable/home otoscopes on the market and three-dimensional devices printed using printers, which diagnose various infections. In all systems there is a semi-automatic process from the moment of initial diagnosis of the patient to the status of inflammation.

I want to change the situation today and integrate a mobile device that would take pictures from the inner parts of the ear for otoscopic surgery performed by a doctor as part of receiving the inputs for further work. I will work in environments to organize the information and implement the software as a result of image processing. This will provide me with the medical status and the determination of whether there is an ear infection.

Background

2.1 Artificial Neural Networks

Artificial Neural Networks (or simply Neural Networks) are frameworks for different machine learning algorithms to work together and process data inputs. Neural Networks are built upon layers of Artificial Neurons that contain an Activation function. Activation functions define the output of a Neuron and map the resulting value to a desired range. Commonly, the Activation function's name is used as the Artificial Neuron's name. The first layer of a Neural Network is the input layer, which contains input Neurons. The last layer is the output layer, which contains output Neurons. The middle layers are called hidden layers since they are not representing input or output. Neural networks with multiple hidden layers are named Deep Neural Networks. The process of delivering data from the input layer, through the hidden layers, to the output layer is called feed-forward.

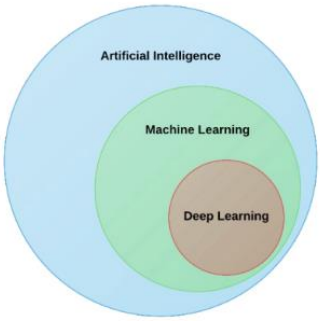
2.2 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves The process of learning begins with observations or data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the examples that I provide. The primary aim is to allow computers to learn automatically without human intervention or assistance and adjust actions accordingly.

2.3 Deep learning

Deep learning is a branch of machine learning that imitates the work of the human brain in processing data and creating patterns for use in decision-making. Deep learning architectures rely on the use of Artificial Neural Networks, which are capable of learning data that is instructed or unlabeled. Deep Learning networks are

fed with large data sets of diverse examples, from which the model learns to look for features in order to produce a probability vector that answers a classification problem. The class with the highest probability is commonly chosen as the answer.

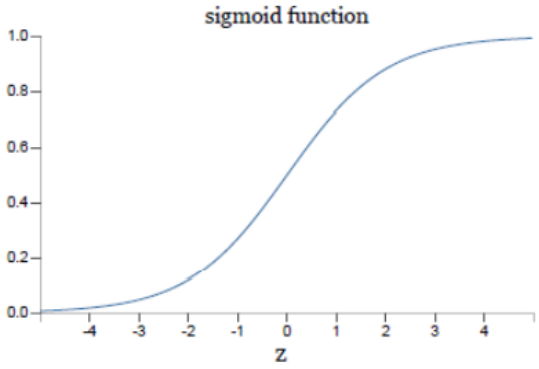


2.4 Activation Function

A mathematical function that theoretically models the system's output for each possible input is also known as Transfer Function. It is used to determine the output of neural networks like yes or no. It maps the resulting values between 0 to 1 or -1 to 1 etc. (depending upon the function). The Activation Functions can be divided into two types: Linear Activation Functions, and Non-linear Activation Functions.

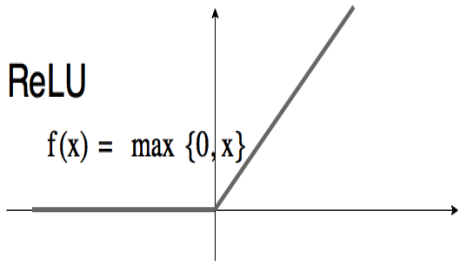
2.4.1 Sigmoid

Sigmoid Function or Binary Sigmoid is a logistic function where the output values are either binary or vary from 0 to 1 numeric weight and bias. It is differentiable, non-linear, and produces non-binary activations but the problem with Sigmoid is the vanishing gradients. Also, sigmoid activation is not a zero-centric function. A small change in the weights or bias will cause only a minor change in the output. This property allows the Neural Network to learn.



2.4.2 ReLU

ReLU Rectified Linear Unit (ReLU) is an Activation function that improves the training of Deep Neural Networks. The ReLU function has a better response than the Sigmoid function on a problem called Vanishing Gradient. The problem occurs when an Activation function squashes the input to a very small output range in a non-linear way. Large regions of inputs are being mapped to very small regions, thus, making large changes in the input reflect only small changes in the output (Small Gradient). On the other hand, a main problem with the ReLU function is that all the negative values become zero immediately, hence, decreasing the ability of the model to fit or train from the data properly. This affects the resulting graph by not mapping the negative values appropriately. In this project, the ReLU functions are being used within the Neural Network.



2.5 Keras layers API

Layers are the basic building blocks of neural networks in Keras. A layer consists of a tensor-in tensor-out computation function (the layer's call method) and some state, held in TensorFlow variables (the layer's weights). A Layer instance is callable, much like a function. Unlike a function, though, layers maintain a state, updated when the layer receives data during training and stored in 'layer.weights'. Creating custom layers while Keras offers a wide range of built-in layers, they don't cover every possible use case. Creating custom layers is very common.

2.5.1 Dense layer

Dense returns a list of two values: the kernel matrix and the bias vector. These can be used to set the weights of another Dense layer.

2.5.2 Flatten layer

Keras flatten is a way to provide input to add an extra layer for flattening using flatten class. Keras flatten flattens the input with no effect on the batch size.

2.5.3 Dropout layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1 (rate) such that the sum of all inputs is unchanged.

2.5.4 Batch normalization layer

Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. Importantly, batch normalization works differently during training and inference. Batch normalization solves a major problem called internal covariate shift. It helps by making the data flowing between intermediate layers of the neural network look, this means you can use a higher learning rate. It has a regularizing effect which means you can often remove 'dropout'.

2.5.5 Softmax layer

Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. Softmax is implemented through a neural network layer just before the output layer.

2.5.6 Classification layer

A classification layer computes the cross-entropy loss for classification and weighted classification tasks with mutually exclusive classes. The layer infers the number of classes from the output size of the previous layer.

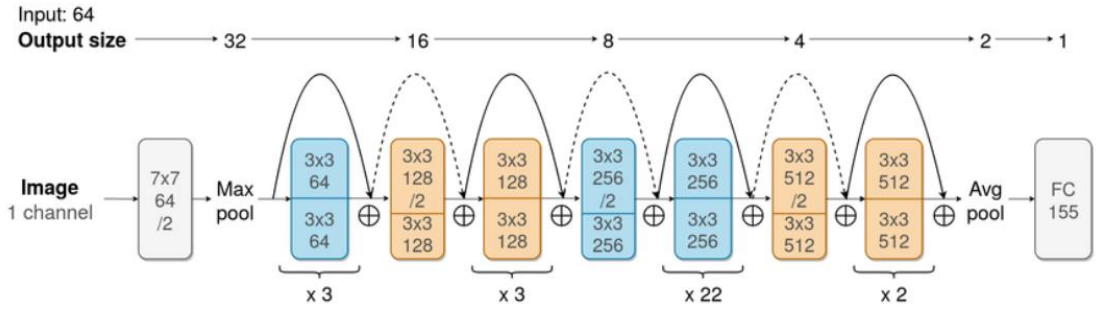
2.6 ResNet

A residual neural network (ResNet) is an artificial neural network (ANN). It is a gateless or open-gated variant of the HighwayNet, the first working very deep feedforward neural network with hundreds of layers, much deeper than previous neural networks. Skip connections or shortcuts are used to jump over some

layers (HighwayNets may also learn the skip weights themselves through an additional weight matrix for their gates). Typical ResNet models are implemented with double -or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

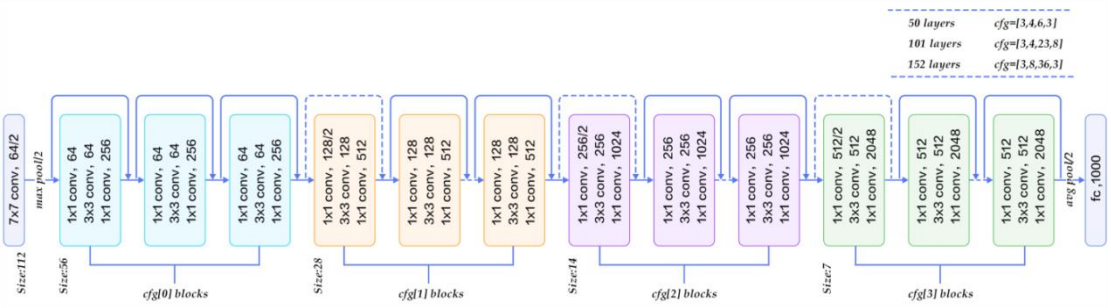
2.6.1 ResNet-101

ResNet-101 is a convolutional neural network that is 101 layers deep. The network has a default image input size of 224x224.



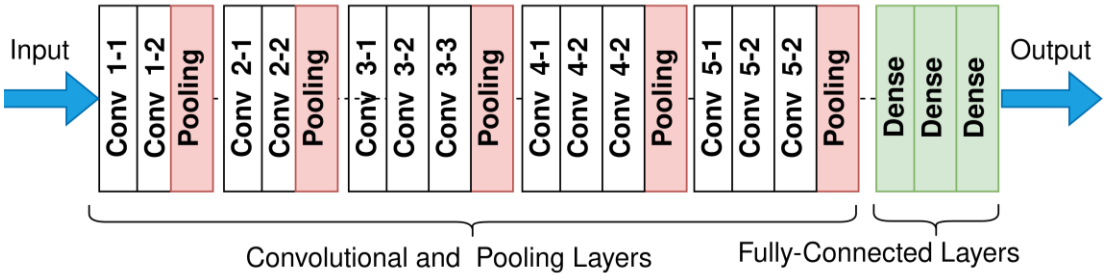
2.6.2 ResNet-50

ResNet-50 is a convolutional neural network that is 50 layers deep. The network has a default image input size of 224x224.



2.7 Vgg-16

VGG-16 is a convolutional neural network that is 16 layers deep. The network has a default image input size of 224x224.



Methodology

Otoscope output image analysis has 4 main phases: preprocessing, transfer learning for feature extraction, similarity measurement, and performance evaluation for image retrieval [1]. In the preprocessing part, I fit the dataset images to the models: ResNet101, ResNet50, and Vgg16 input. The second phase is the transfer learning that trains the model on images for both classes: Normal and effusion. A set of features is extracted from the query image and compared with the annotated set of cases in the database using different distance metrics to determine their similarity. The most relevant image classification is then shown to the user.

Hyperparameters for the models I used:

ResNet-101

```
Input: 224x224 24bit jpg images
Adam(learning_rate=1e-3)
loss="sparse_categorical_crossentropy"
metrics=['accuracy']
epochs=3
batch_size=32
```

ResNet-50

```
Input: 224x224 24bit jpg images
Adam(learning_rate=1e-4)
loss="sparse_categorical_crossentropy"
metrics=['accuracy']
epochs=30
batch_size=64
```

Vgg-16

```
Input: 224x224 24bit jpg images
Adam(learning_rate=1e-5)
loss="sparse_categorical_crossentropy"
metrics=['accuracy']
epochs=30
batch_size=32
```

Dataset

The images used in this project were captured at ear primary care clinics from both adult and pediatric patients at the Ohio State University (OSU) and Nationwide Children's Hospital (NCH) in Columbus, Ohio, US with IRB approval. Furthermore, all the samples were fully anonymized by the rules set by the Ohio State University, Institutional Review Board.

The input images are 224x224 jpg images with a 24bit depth 3 channels RGB. It was challenging to collect an equal number of images for each diagnostic category.

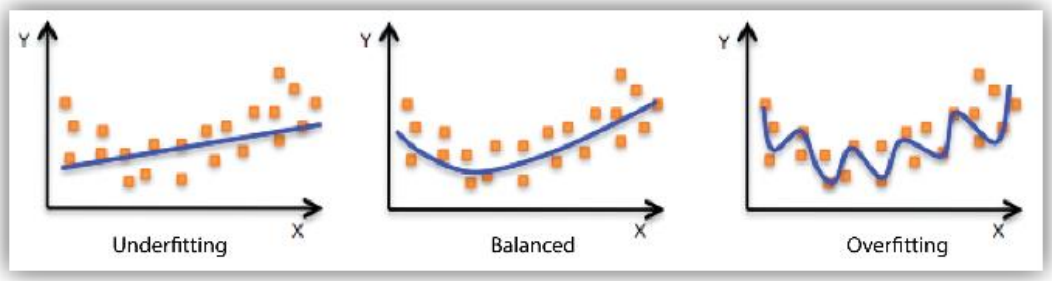
Feature extraction

Feature extraction is one of the most critical phases of my project that affects the similarity of the query image to the retrieved images. I propose a methodology to make use of a deep learning model to extract features for an image retrieval system. I showed how transfer learning can be converted into an image retrieval system.

Transfer learning is a machine learning method that uses features learned from a problem in one domain and applies them to a problem in a different domain.

Classification of the eardrum abnormalities is not always sufficient for a decision support system because the end user may not fully understand the decision made by the computerized analysis. Presenting several examples of similar images may increase the confidence of the end user. For this reason, I propose a method to retrieve similar images depending on their class of abnormality. The transfer learning is achieved by training the last few layers of ResNet101, ResNet50 , and Vgg16 neural networks. This method can acquire knowledge by automatically extracting features from eardrum images and using them to retrieve similar images to a query image.

I used ResNet101 Convolutional Neural Network (CNN). With my limited dataset, retraining the whole network is highly likely to result in overfitting. For this reason, I opted to freeze the first layers, limiting the number of parameters required to retrain the network. I retrained the last three layers (flatten, classification and softmax) of the pre-trained network with otoscope images in my database. The choice of the number of retraining layers was mainly driven by the number of training images in my database and the number of parameters that I can learn from the database.



6. Challenges

In the beginning, I found 'Teachable Machine'. This web-based tool makes creating machine learning models fast, easy, and accessible to everyone. That was a good option to understand how things are going to work but I faced a problem with that tool that is not letting me choose a specific model as I wished. Moreover, I wanted to adjust the hyper-parameters according to my needs but I couldn't do so. Hence I searched for other options. After deep research, I learned that it is possible to train a model in python environment and insert it into my application that was developed in android studio in java. I understood that I can train a model in Tensorflow – Google open sourced library for machine learning and creation. I had issues with my model predictions. The model was inaccurate and the loss function wasn't improved as I expected. In order to increase the model's accuracy I did several observations. I used outlier detection to remove unwanted images that were not close to the ideal shots (focused, bright, and cleared) [3][4]. I decreased the neuron's number of the top layers so the model had less weight. Moreover, I used dropout with a 0.5 ratio at ResNet50 and Vgg16 to tune them as much as possible for balance. After I found the

best model for my dataset and needs, I converted it to Tensorflow Lite – which provides a set of tools that enables on-device machine learning by allowing developers to run their trained models on mobile, embedded, and IoT devices and computers. Since the model's size before the conversion was 554MB and the maximum size for the Tensorflow Lite model should be a Maximum of 200 MB I had to quantize it to 139MB. I used dynamic quantized which is 4 times lighter and 2-3 times faster. This kind of quantized model works on the CPU. In order to save the CPU model I tried to process it first in Google colab. Due to the RAM limit on colab server (12GB), I have to change the way I evaluate my model. I chose Pycharm as the environment for that need so I was no longer limited by resources.

7. Results

The findings of Resnet-101 were the best of the models I tested on my dataset hense I implanted it in my application.

ResNet-101 Results

Epochs

```
Epoch 1/3
10/10 [==============================] - 57s 5s/step - loss: 7.9319 - accuracy: 0.6698
Epoch 2/3
10/10 [==============================] - 50s 5s/step - loss: 0.5216 - accuracy: 0.8836
Epoch 3/3
10/10 [==============================] - 49s 5s/step - loss: 0.0856 - accuracy: 0.9717
```

Prediction

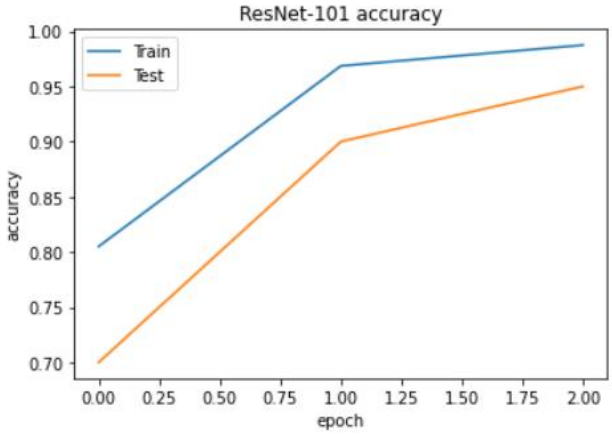
10 effusion test images were tested. 9/10 of was observed correctly.

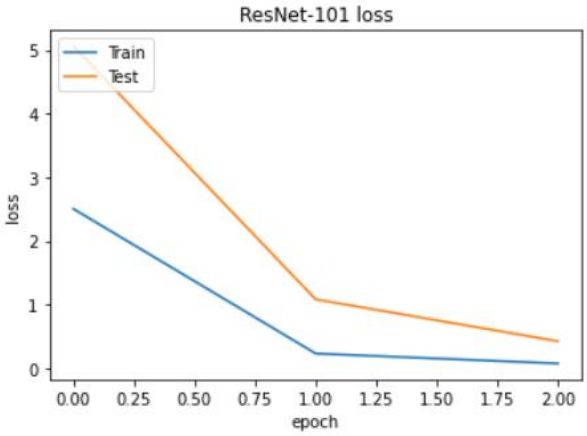
```
AM208R.jpg
1/1 [==============================] - 1s 831ms/step
This image is Effusion
AM157L.jpg
1/1 [==============================] - 1s 1s/step
This image is Effusion
AM156L.jpg
1/1 [==============================] - 1s 1s/step
This image is Effusion
AM225L.jpg
1/1 [==============================] - 1s 1s/step
This image is Effusion
AM222R.jpg
1/1 [==============================] - 0s 460ms/step
This image is Effusion
AM199L.jpg
1/1 [==============================] - 0s 443ms/step
This image is Effusion
AM185L.jpg
1/1 [==============================] - 0s 452ms/step
This image is Normal
AM203L.jpg
1/1 [==============================] - 0s 447ms/step
This image is Effusion
AM186L.jpg
1/1 [==============================] - 0s 454ms/step
This image is Effusion
AM162R.jpg
1/1 [==============================] - 0s 440ms/step
This image is Effusion
```

10 normal test images were tested. 10/10 of the data was observed correctly.

```
AM11R.jpg
1/1 [==============================] - 1s 527ms/step
This image is Normal
AM7R.jpg
1/1 [==============================] - 0s 491ms/step
This image is Normal
AM6L.jpg
1/1 [==============================] - 0s 489ms/step
This image is Normal
AM12L.jpg
1/1 [==============================] - 0s 491ms/step
This image is Normal
AM3R.jpg
1/1 [==============================] - 1s 504ms/step
This image is Normal
AM17L.jpg
1/1 [==============================] - 0s 496ms/step
This image is Normal
AM6R.jpg
1/1 [==============================] - 0s 477ms/step
This image is Normal
AM7L.jpg
1/1 [==============================] - 0s 480ms/step
This image is Normal
AM1L.jpg
1/1 [==============================] - 0s 489ms/step
This image is Normal
AM10L.jpg
1/1 [==============================] - 0s 472ms/step
This image is Normal
```

Accuracy and loss





ResNet-50 Results

Epochs

```
Epoch 1/30
10/10 [==============================] - 8s 251ms/step - loss: 0.9328 - accuracy: 0.5975 - val_loss: 0.9686 - val_accuracy: 0.6500
Epoch 2/30
10/10 [==============================] - 2s 147ms/step - loss: 0.7029 - accuracy: 0.6761 - val_loss: 1.1440 - val_accuracy: 0.5500
Epoch 3/30
10/10 [==============================] - 2s 146ms/step - loss: 0.6272 - accuracy: 0.7233 - val_loss: 1.1352 - val_accuracy: 0.5000
Epoch 4/30
10/10 [==============================] - 2s 146ms/step - loss: 0.5368 - accuracy: 0.7044 - val_loss: 0.9832 - val_accuracy: 0.5000
Epoch 5/30
10/10 [==============================] - 2s 145ms/step - loss: 0.5323 - accuracy: 0.7358 - val_loss: 0.8943 - val_accuracy: 0.5500
Epoch 6/30
10/10 [==============================] - 2s 147ms/step - loss: 0.5802 - accuracy: 0.7264 - val_loss: 0.8578 - val_accuracy: 0.5500
Epoch 7/30
10/10 [==============================] - 2s 146ms/step - loss: 0.4640 - accuracy: 0.7987 - val_loss: 0.8810 - val_accuracy: 0.5000
Epoch 8/30
10/10 [==============================] - 2s 147ms/step - loss: 0.3890 - accuracy: 0.8302 - val_loss: 0.8890 - val_accuracy: 0.5000
Epoch 9/30
10/10 [==============================] - 2s 146ms/step - loss: 0.4192 - accuracy: 0.8145 - val_loss: 0.9066 - val_accuracy: 0.4500
Epoch 10/30
10/10 [==============================] - 2s 146ms/step - loss: 0.3868 - accuracy: 0.8270 - val_loss: 0.8908 - val_accuracy: 0.5000
Epoch 11/30
10/10 [==============================] - 2s 147ms/step - loss: 0.3539 - accuracy: 0.8333 - val_loss: 0.8821 - val_accuracy: 0.5000
Epoch 12/30
10/10 [==============================] - 2s 147ms/step - loss: 0.3611 - accuracy: 0.8553 - val_loss: 0.8851 - val_accuracy: 0.5500
Epoch 13/30
10/10 [==============================] - 2s 146ms/step - loss: 0.2767 - accuracy: 0.8774 - val_loss: 0.8797 - val_accuracy: 0.5500
Epoch 14/30
10/10 [==============================] - 2s 146ms/step - loss: 0.3194 - accuracy: 0.8428 - val_loss: 0.8735 - val_accuracy: 0.5500
Epoch 15/30
10/10 [==============================] - 2s 147ms/step - loss: 0.3668 - accuracy: 0.8428 - val_loss: 0.8741 - val_accuracy: 0.5500
Epoch 16/30
10/10 [==============================] - 2s 146ms/step - loss: 0.2414 - accuracy: 0.9151 - val_loss: 0.8635 - val_accuracy: 0.5500
Epoch 17/30
10/10 [==============================] - 2s 148ms/step - loss: 0.2664 - accuracy: 0.8931 - val_loss: 0.8608 - val_accuracy: 0.5500
Epoch 18/30
10/10 [==============================] - 2s 148ms/step - loss: 0.2889 - accuracy: 0.8774 - val_loss: 0.8618 - val_accuracy: 0.5500
Epoch 19/30
10/10 [==============================] - 2s 149ms/step - loss: 0.2023 - accuracy: 0.9277 - val_loss: 0.8629 - val_accuracy: 0.5500
Epoch 20/30
10/10 [==============================] - 2s 147ms/step - loss: 0.2359 - accuracy: 0.8931 - val_loss: 0.8406 - val_accuracy: 0.5500
Epoch 21/30
10/10 [==============================] - 2s 147ms/step - loss: 0.2186 - accuracy: 0.8994 - val_loss: 0.8346 - val_accuracy: 0.5500
Epoch 22/30
10/10 [==============================] - 2s 148ms/step - loss: 0.2503 - accuracy: 0.9151 - val_loss: 0.8307 - val_accuracy: 0.5500
Epoch 23/30
10/10 [==============================] - 2s 148ms/step - loss: 0.2124 - accuracy: 0.9057 - val_loss: 0.7951 - val_accuracy: 0.5500
Epoch 24/30
10/10 [==============================] - 2s 148ms/step - loss: 0.2164 - accuracy: 0.9151 - val_loss: 0.7640 - val_accuracy: 0.5500
Epoch 25/30
10/10 [==============================] - 2s 149ms/step - loss: 0.1607 - accuracy: 0.9371 - val_loss: 0.7443 - val_accuracy: 0.6000
Epoch 26/30
10/10 [==============================] - 2s 148ms/step - loss: 0.1583 - accuracy: 0.9434 - val_loss: 0.7668 - val_accuracy: 0.6000
Epoch 27/30
10/10 [==============================] - 2s 148ms/step - loss: 0.1373 - accuracy: 0.9560 - val_loss: 0.7862 - val_accuracy: 0.6000
Epoch 28/30
10/10 [==============================] - 2s 149ms/step - loss: 0.1411 - accuracy: 0.9591 - val_loss: 0.7869 - val_accuracy: 0.6000
Epoch 29/30
10/10 [==============================] - 2s 148ms/step - loss: 0.1739 - accuracy: 0.9340 - val_loss: 0.7802 - val_accuracy: 0.5500
Epoch 30/30
10/10 [==============================] - 2s 149ms/step - loss: 0.1555 - accuracy: 0.9560 - val_loss: 0.7790 - val_accuracy: 0.5500
```

Prediction

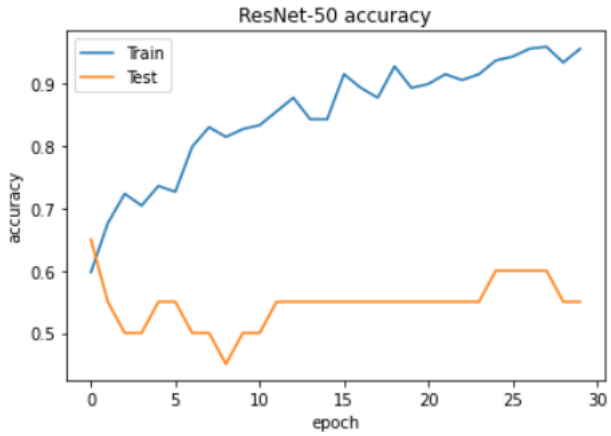
10 effusion test images were tested. 4/10 of the data was observed correctly.

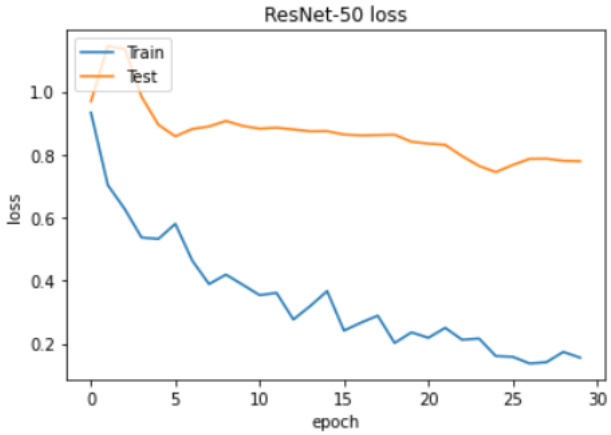
```
AM157L.jpg
1/1 [==============================] - 0s 23ms/step
This image is Normal
AM203L.jpg
1/1 [==============================] - 0s 23ms/step
This image is Effusion
AM162R.jpg
1/1 [==============================] - 0s 21ms/step
This image is Normal
AM222R.jpg
1/1 [==============================] - 0s 22ms/step
This image is Effusion
AM208R.jpg
1/1 [==============================] - 0s 24ms/step
This image is Normal
AM199L.jpg
1/1 [==============================] - 0s 21ms/step
This image is Normal
AM185L.jpg
1/1 [==============================] - 0s 23ms/step
This image is Normal
AM186L.jpg
1/1 [==============================] - 0s 20ms/step
This image is Effusion
AM156L.jpg
1/1 [==============================] - 0s 20ms/step
This image is Effusion
AM225L.jpg
1/1 [==============================] - 0s 20ms/step
This image is Normal
```

10 normal test images were tested. 9/10 of the data was observed correctly.

```
AM11R.jpg
1/1 [==============================] - 0s 22ms/step
This image is Normal
AM17L.jpg
1/1 [==============================] - 0s 20ms/step
This image is Normal
AM6L.jpg
1/1 [==============================] - 0s 20ms/step
This image is Normal
AM1L.jpg
1/1 [==============================] - 0s 22ms/step
This image is Normal
AM10L.jpg
1/1 [==============================] - 0s 23ms/step
This image is Normal
AM7R.jpg
1/1 [==============================] - 0s 22ms/step
This image is Normal
AM6R.jpg
1/1 [==============================] - 0s 22ms/step
This image is Normal
AM12L.jpg
1/1 [==============================] - 0s 24ms/step
This image is Normal
AM3R.jpg
1/1 [==============================] - 0s 20ms/step
This image is Effusion
AM7L.jpg
1/1 [==============================] - 0s 21ms/step
This image is Normal
```

Accuracy and loss





Vgg-16 Results

Epochs

```
Epoch 1/30
10/10 [==============================] - 3s 203ms/step - loss: 5.0019 - accuracy: 0.5126 - val_loss: 3.0650 - val_accuracy: 0.6000
Epoch 2/30
10/10 [==============================] - 2s 183ms/step - loss: 4.3714 - accuracy: 0.4969 - val_loss: 2.8168 - val_accuracy: 0.5500
Epoch 3/30
10/10 [==============================] - 2s 183ms/step - loss: 3.7144 - accuracy: 0.5031 - val_loss: 2.7250 - val_accuracy: 0.5000
Epoch 4/30
10/10 [==============================] - 2s 183ms/step - loss: 2.6593 - accuracy: 0.5975 - val_loss: 2.7027 - val_accuracy: 0.4500
Epoch 5/30
10/10 [==============================] - 2s 185ms/step - loss: 2.7501 - accuracy: 0.5629 - val_loss: 2.5141 - val_accuracy: 0.4500
Epoch 6/30
10/10 [==============================] - 3s 185ms/step - loss: 2.3949 - accuracy: 0.5912 - val_loss: 2.3808 - val_accuracy: 0.4500
Epoch 7/30
10/10 [==============================] - 2s 183ms/step - loss: 1.9941 - accuracy: 0.6101 - val_loss: 2.3241 - val_accuracy: 0.4500
Epoch 8/30
10/10 [==============================] - 2s 183ms/step - loss: 1.7709 - accuracy: 0.6541 - val_loss: 2.3416 - val_accuracy: 0.4000
Epoch 9/30
10/10 [==============================] - 2s 182ms/step - loss: 1.3862 - accuracy: 0.6730 - val_loss: 2.2814 - val_accuracy: 0.4000
Epoch 10/30
10/10 [==============================] - 2s 183ms/step - loss: 1.4005 - accuracy: 0.6950 - val_loss: 2.2795 - val_accuracy: 0.4500
Epoch 11/30
10/10 [==============================] - 2s 183ms/step - loss: 1.2141 - accuracy: 0.7107 - val_loss: 2.2210 - val_accuracy: 0.4500
Epoch 12/30
10/10 [==============================] - 2s 181ms/step - loss: 1.0686 - accuracy: 0.7170 - val_loss: 2.0997 - val_accuracy: 0.5000
Epoch 13/30
10/10 [==============================] - 2s 182ms/step - loss: 0.9536 - accuracy: 0.7138 - val_loss: 2.1219 - val_accuracy: 0.5000
Epoch 14/30
10/10 [==============================] - 2s 181ms/step - loss: 0.9991 - accuracy: 0.7044 - val_loss: 2.1198 - val_accuracy: 0.5000
Epoch 15/30
10/10 [==============================] - 2s 183ms/step - loss: 0.9486 - accuracy: 0.7044 - val_loss: 2.1252 - val_accuracy: 0.4500
Epoch 16/30
10/10 [==============================] - 2s 183ms/step - loss: 0.9339 - accuracy: 0.7044 - val_loss: 2.0643 - val_accuracy: 0.4500
Epoch 17/30
10/10 [==============================] - 2s 184ms/step - loss: 0.8603 - accuracy: 0.7138 - val_loss: 1.9740 - val_accuracy: 0.4500
Epoch 18/30
10/10 [==============================] - 2s 182ms/step - loss: 0.8352 - accuracy: 0.7421 - val_loss: 1.8970 - val_accuracy: 0.5000
Epoch 19/30
10/10 [==============================] - 2s 185ms/step - loss: 0.7333 - accuracy: 0.7547 - val_loss: 1.8044 - val_accuracy: 0.5000
Epoch 20/30
10/10 [==============================] - 2s 183ms/step - loss: 0.5740 - accuracy: 0.7736 - val_loss: 1.7798 - val_accuracy: 0.5000
Epoch 21/30
10/10 [==============================] - 2s 184ms/step - loss: 0.5750 - accuracy: 0.7925 - val_loss: 1.7462 - val_accuracy: 0.5000
Epoch 22/30
10/10 [==============================] - 2s 189ms/step - loss: 0.5680 - accuracy: 0.7767 - val_loss: 1.7764 - val_accuracy: 0.5500
Epoch 23/30
10/10 [==============================] - 3s 185ms/step - loss: 0.7527 - accuracy: 0.7421 - val_loss: 1.7839 - val_accuracy: 0.5500
Epoch 24/30
10/10 [==============================] - 2s 183ms/step - loss: 0.6097 - accuracy: 0.7484 - val_loss: 1.7631 - val_accuracy: 0.5500
Epoch 25/30
10/10 [==============================] - 2s 182ms/step - loss: 0.5945 - accuracy: 0.7862 - val_loss: 1.7249 - val_accuracy: 0.5500
Epoch 26/30
10/10 [==============================] - 2s 183ms/step - loss: 0.4792 - accuracy: 0.8113 - val_loss: 1.6482 - val_accuracy: 0.5500
Epoch 27/30
10/10 [==============================] - 2s 181ms/step - loss: 0.4288 - accuracy: 0.8270 - val_loss: 1.5898 - val_accuracy: 0.5500
Epoch 28/30
10/10 [==============================] - 2s 192ms/step - loss: 0.4798 - accuracy: 0.7956 - val_loss: 1.5757 - val_accuracy: 0.5500
Epoch 29/30
10/10 [==============================] - 2s 183ms/step - loss: 0.6124 - accuracy: 0.7925 - val_loss: 1.5742 - val_accuracy: 0.5000
Epoch 30/30
10/10 [==============================] - 2s 184ms/step - loss: 0.4696 - accuracy: 0.8082 - val_loss: 1.5859 - val_accuracy: 0.5000
```

Prediction

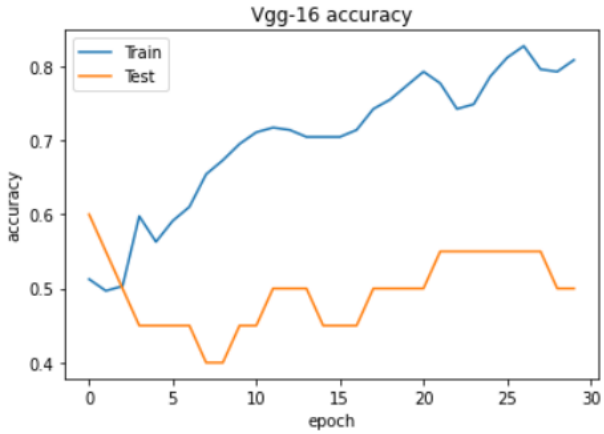
10 effusion test images were tested. 4/10 of the data was observed correctly.

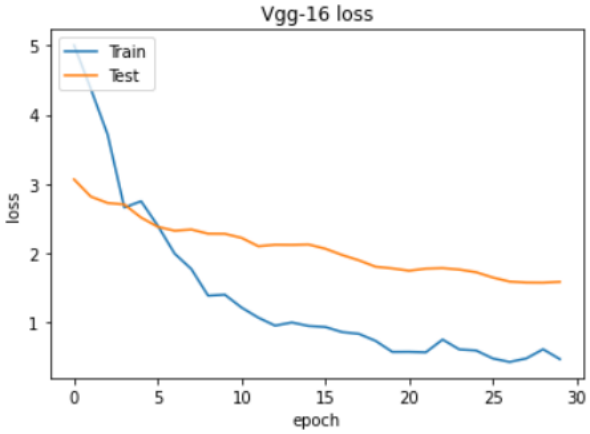
```
AM157L.jpg
1/1 [==============================] - 1s 770ms/step
This image is Effusion
AM203L.jpg
1/1 [==============================] - 0s 16ms/step
This image is Normal
AM162R.jpg
1/1 [==============================] - 0s 17ms/step
This image is Normal
AM222R.jpg
1/1 [==============================] - 0s 18ms/step
This image is Effusion
AM208R.jpg
1/1 [==============================] - 0s 33ms/step
This image is Normal
AM199L.jpg
1/1 [==============================] - 0s 28ms/step
This image is Normal
AM185L.jpg
1/1 [==============================] - 0s 30ms/step
This image is Normal
AM186L.jpg
1/1 [==============================] - 0s 26ms/step
This image is Effusion
AM156L.jpg
1/1 [==============================] - 0s 23ms/step
This image is Normal
AM225L.jpg
1/1 [==============================] - 0s 25ms/step
This image is Effusion
```

10 normal test images were tested. 6/10 of the data was observed correctly.

```
AM11R.jpg
1/1 [==============================] - 0s 20ms/step
This image is Normal
AM17L.jpg
1/1 [==============================] - 0s 17ms/step
This image is Effusion
AM6L.jpg
1/1 [==============================] - 0s 19ms/step
This image is Effusion
AM1L.jpg
1/1 [==============================] - 0s 17ms/step
This image is Effusion
AM10L.jpg
1/1 [==============================] - 0s 19ms/step
This image is Normal
AM7R.jpg
1/1 [==============================] - 0s 15ms/step
This image is Normal
AM6R.jpg
1/1 [==============================] - 0s 15ms/step
This image is Normal
AM12L.jpg
1/1 [==============================] - 0s 17ms/step
This image is Effusion
AM3R.jpg
1/1 [==============================] - 0s 17ms/step
This image is Normal
AM7L.jpg
1/1 [==============================] - 0s 15ms/step
This image is Normal
```

Accuracy and loss





8. Statistical analysis

True /Positive – TP, actual effusion truly predicted

True/Negative – TN, actual effusion false predicted

False/Positive – FP, actual normal truly predicted

False/Negative – FN, actual normal false predicted

TP = 9, FN = 1 , FP=0, TN = 10

Accuracy $=\frac{TP + TN}{TP + TN+FP+FN} =\frac{9+10}{9+10+0+1} =\frac{19}{20}$=0.95=95%

Precision$=\frac{TP}{TP +FP}=\frac{9}{9 +0}$=1=100%

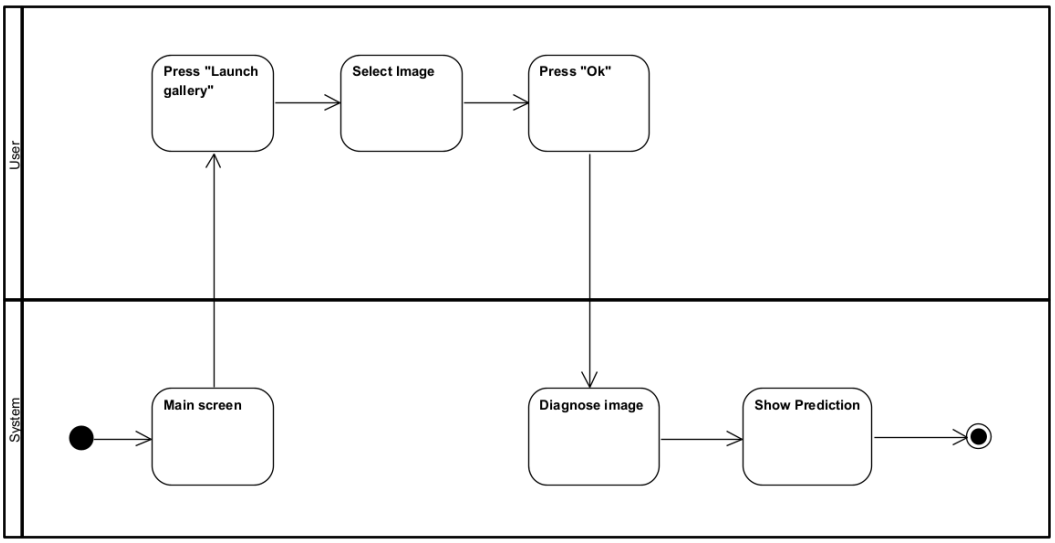
Sensitivity$=\frac{TP}{TP +FN}=\frac{9}{9 +10}=\frac{9}{19}$=0.4736=47.36%

F1 score$=\frac{2*Precision*Sensitivity}{Precision +Sensitivity}=\frac{2*1*0.47}{1 +0.47}=\frac{0.94}{1.47}$=0.639=63%

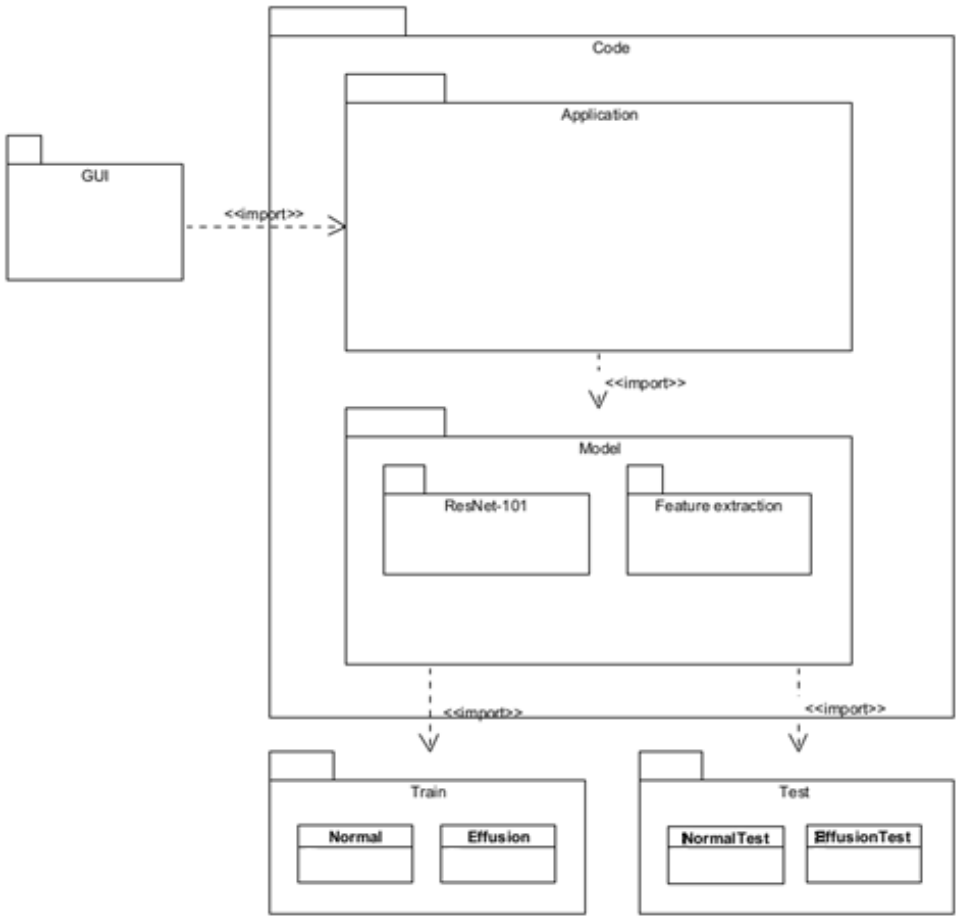
Specificity$=\frac{TN}{TN +FP}=\frac{10}{10 +0}$=1=100%

9. UML's

Activity diagram



Package diagram



10. Testing plan

Case number	Test Case	Description	Expected result	Actual result
1	Invalid image format	Few image formats other than JPG such as TIFF, JPEG have been loaded into the application.	The images will be still processed and predicted well.	No matter what was the image format that has been loaded, the model still was effective.
2	Expected image format	Images in JPG were loaded into the application.	Image prediction works well	All of the image formats will be accepted by the application.
3	Invalid image size	Images with other sizes than 224x224 have been loaded into the application.	The images will be rescaled during the preprocessing and will predict well.	The process has been done successfully.
4	Expected image size	The application was loaded with 224x224 images.	The application will analyze and predict the images well.	The process has been done successfully.
5	Unclear image	Blurry and unclear images were inserted into the application.	The prediction will not be accurate.	Some were well-predicted. In the case of the dark fields and low light, it was hard to.
6	Clear image	Clear images with bright light have been loaded.	The prediction will be well accurate.	The process has been done successfully.
7	Diagnosis not found	Images that are not ‘Normal’ or ‘Effusion’ were loaded.	There is not enough features to give an accurate diagnosis. The application will not predict well.	Since the model can deal with two classes, everything else will be classified to the closest of them.
8	Diagnosis found successfully	‘Normal’ or ‘Effusion’ images were loaded into the application.	The diagnosis will be accurate.	The process has been done successfully.

11. Developer manual

The process of making the application usable is divided into two main parts

1. Creating the model

2. Implant the model in the application

*__Attention__ – The explanations are based on the fact that there is no problem with resources.

1. Model creation

1.1 Go to Google colab and load the dataset.

1.2 Once loaded unzip it.

1.3 import the relevant libraries

1.4 insert the input image size, batch size, and seed.

1.5 locate the train and validation/test paths. (Note that for each DataSet, there should be a sub-folder for each class one for normal and the other for effusion **.**

1.6 Importing the ResNet-101 model from TensorFlow Keras.

1.7 Compiling the adjusted model with the relevant parameters.

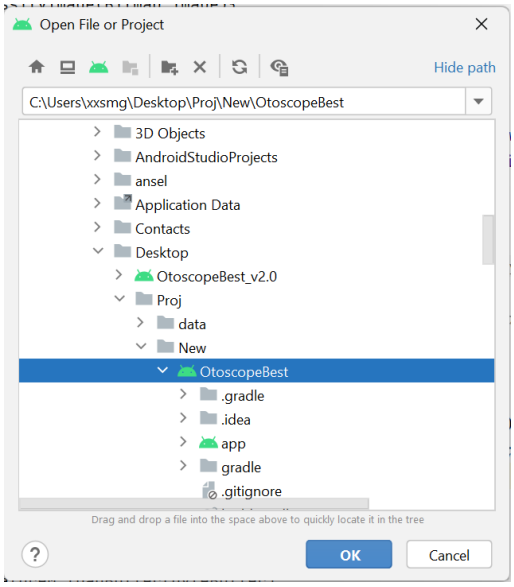
1.8 Training the model.

1.1.1 Prediction session.

1.1.2 Model convert for TensorFlow lite to be dynamic quantized. As this kind, it can be loaded to the application (Under 200MB). If its name is not model.tflite rename it to this name.

2. The application should be loaded in android studio.

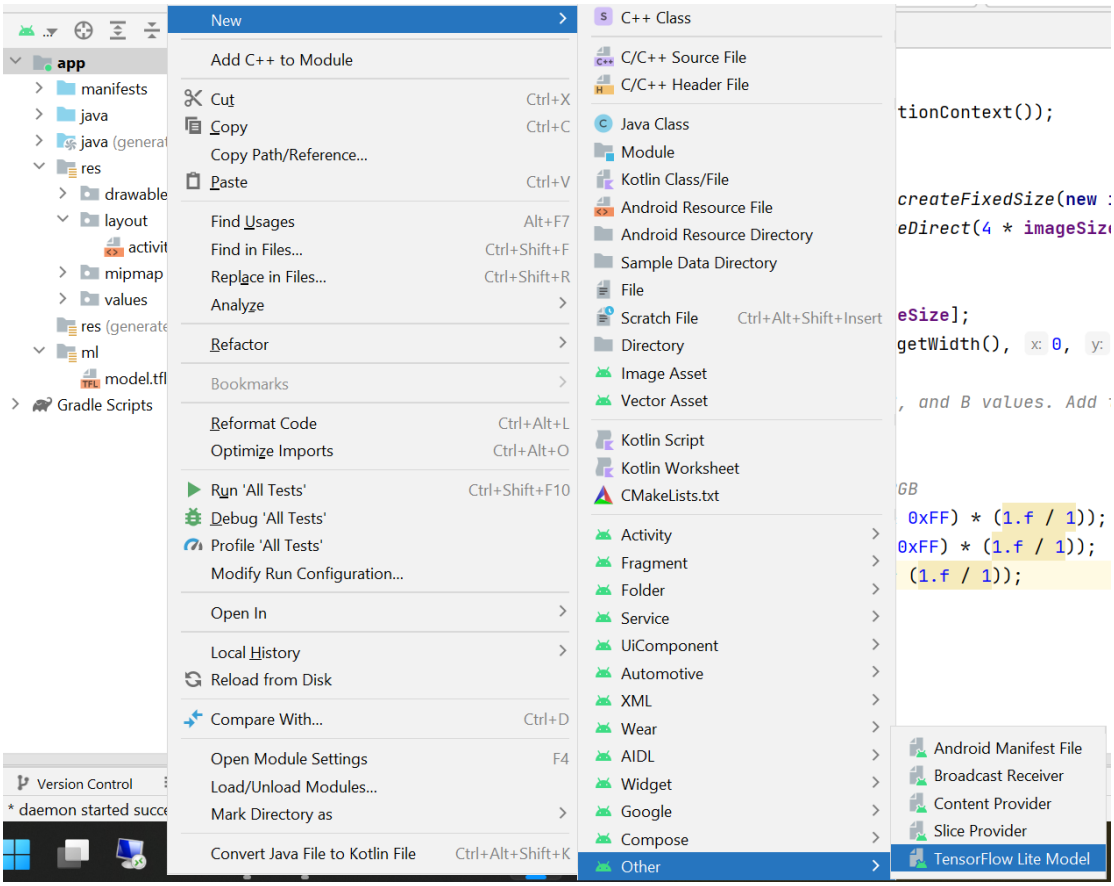
Enter Android studio. File -> Open -> Locate the application path.



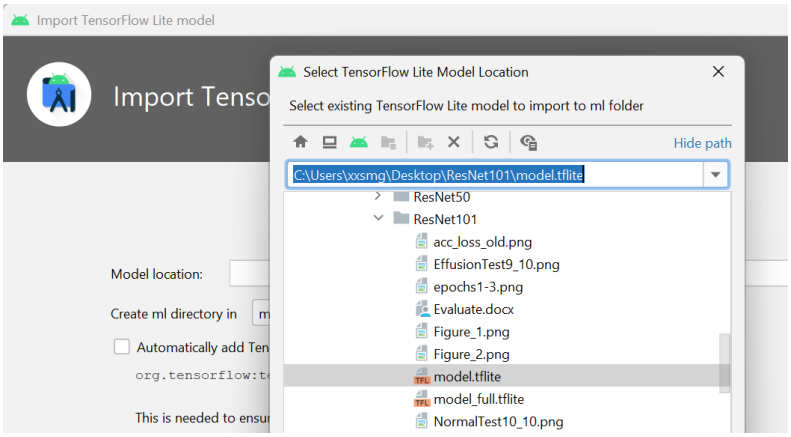
Click 'Ok'.

2.1 The model should be loaded through as follows

Right-click on app -> New -> Other ->TensorFlow Lite Model



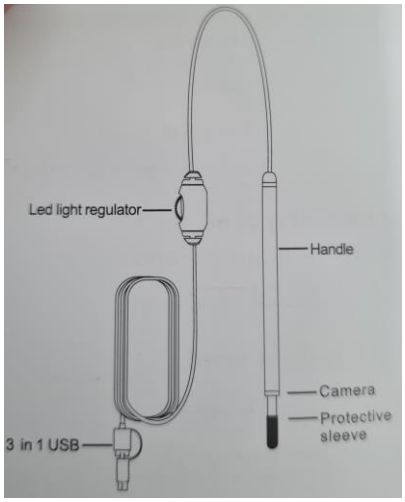
2.2 Next step is to locate the location of the model.



2.3 The model now is loaded into the application.

12. User manual

1. The external otoscope device can be plugged into the device. My external endoscope is used by CameraFi software. For existing image diagnoses please to go step 4 below.



1.1 To use the device on the phone:

- Android 4.2 or later should be installed.
- Have an open OTG function (make sure the phone turns on the OTG function before use).
- Access to an external camera.

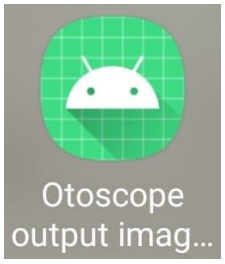
1.2 Download APP: Search for “the camera fi” app on the Google play store, or scan the QR below.

1.3 Install the app on the phone.

1.4 Connect the device to the phone, then you can use it.

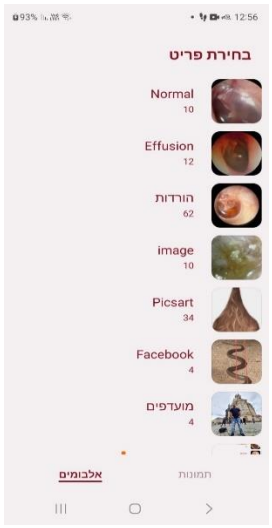
2. Launch the device application from an android device.

3. Take the required pictures and save them on the device.
4. Launch the application “Otoscope Output image analysis application”.



5. Click “Launch Gallery” for image import.

6. Locate an image for diagnosis: normal eardrum or inflamed eardrum.



7. The image is now classified as required.

13. Conclusions

My project was developed to assist parents of little children with ear disease diagnosis by retrieving the most similar images from a database of annotated cases. More importantly, this project aims to propose a generic model that can be used in conjunction with any convolutional neural network. I proposed a methodology to convert a deep learning model into an image retrieval system. For proof of concept, I showed how transfer learning can be used in the context of otoscopy imaging retrieval. I used deep learning-based approaches to extract the features to compare the similarity of query images with training images in multi-dimensional space. According to the experimental results, for the two categories with the highest numbers of images in my database, the maximum test accuracy of the system using transfer learning was 95%. I explored the variability of performance depending on some of the experimental design factors. In contrast, I observed that the performance would depend on the type of abnormality. Limitations of the study include a relatively small number of images in each category.

14. Environment

Software

Google Colab

PyCharm Community Edition 2021.2.2

Android Studio Dolphin 2021.3.1

Visual Paradigm 16.3

Devices

Visual auditory endoscope Model A98

Samsung Galaxy S20 FE 5G SM-G781B/DS

FX705GD_FX705GD CPU i7-8750H @2.20Ghz 2.21Ghz 64bit based processor

*Attention

The application requires a target API level of 31 or higher

The endoscope needs android 4.2 or higher

15. References

[1] Seda CamalanID1 , Muhammad Khalid Khan Niazi1 , Aaron C. Moberly2 , Theodoros Teknos3 , Garth Essig2 , Charles Elmaraghy2 , Nazhat Taj-Schaal4 , Metin N. Gurcan1 *"OtoMatch"* Content-based eardrum image retrieval using deep learning*"* May 15, 2020

[2] David P. McCormick, M.D. "*Acute Otisis Media*" Department of Pediatrics.

[3] UTHealth Houston "*Ear Anatomy Images*" Otorhinolaryngology – Head & Neck Surgery, 2008.

[4] Harmes KM, et al. (*American Family Physician) Otitis media: diagnosis and treatment*, Dec 1, 2007.

[5] Demir E, Topal S, Atsal G, Erdil M, Coskun ZO, Dursun E. *Otologic Findings Based on no Complaints in a Pediatric Examination*. Int Arch Otorhinolaryngol. 2019.

[6] Chang P, Pedler K. *Ear examination - a practical guide*. Aust Fam Physician. 2005.

[7] Szymanski A, Geiger Z. StatPearls "*StatPearls Publishing*" Treasure Island (FL): Jul 26, 2021. Anatomy, Head and Neck, Ear.

[8] Eavey RD, Stool SE, Peckham GJ, Mitchell LR. How to examine the ear of the neonate. "*Careful monitoring can recognize problems early*". Clin Pediatr (Phila). Nov 9, 2012.