A

Major Project

On

SELF- SUPERVISED LEARNING FOR MEDICAL IMAGING

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Lekkala Harika Chowdary (187R1A05F3)

Koneru Pragna(187R1A05G1)

Nalla Bhavani (187R1A05G7)

Under the Guidance of

Dr. B.LAXMAIAH
(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V),Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "SELF-SUPERVISED LEARNING FOR MEDICAL IMAGING" is being submitted by **L. Harika Chowdary(187R1A05F3), K. Pragna (187R1A05G1)** & **N. Bhavani(187R1A05G7)** in partial fulfillment ofthe requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out byhim/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. B. Laxmaiah
(AssociateProfessor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ______________________

ACKNOWLEDGMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Dr. B.LAXMAIAH,** Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given byhim shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Dr. Suwarna Gothane, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

Lekkala Harika Chowdary (187R1A05F3)

Koneru Pragna (187R1A05G1)

Nalla Bhavani (187R1A05G7)

`

ABSTRACT

Deep learning role in medical imaging is increasing quite effectively. As the models are providing promising accuracy, the early detection and risk mitigation of several diseases is becoming easy. Diabetic Retinopathy is one such disease, where early detection plays a severe role as it can lead to vision loss.

To implement a model in supervised manner, we need huge amount of labeled data set which can be very costly. So as to overcome these problems, in this paper we have implemented a self-supervised learning model for detection of diabetic retinopathy, using very limited dataset. This model is implemented using one of the pretext/proxy task image rotations developed on Dense NET architecture. The model is fine-tuned with the various quantities of subsets of the original dataset and compared internally.

`

LIST OF FIGURES/TABLES

`

TABLE OF CONTENTS:

`

`

INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

Recognition is identifying or distinguishing a thing or an individual from the past experiences or learning. Similarly, Disease recognition is nothing but recognizing or identifying the disease. Disease recognition framework is simply the working of a machine to prepare itself or interpret the disease. There is an enormous amount of data that is being produced on a daily basis from different areas using different imaging modalities such as MRI, CT, microscopy, etc., leading to an unprecedented potential for machine learning algorithms.

The ophthalmology field has benefited from recent advances in deep learning, particularly in the case of deep convolutional neural networks (CNNs) when applied to large data sets, such as two-dimensional (2D) fundus photography, a low-key imaging technology that captures the back of the eye. These images can be taken using a smartphone and are available in a standardized fashion, often in very large quantities. Using data from diabetes screening programs and biobanks, cardiovascular risk factors, presence of diabetic retinopathy, and even gender, can be predicted with a high degree of accuracy Using Convolutional Neural Networks, we can predict the diabetic retinopathy and detect the different stages of it. The model also finds the accuracy of the model and finds the value by using “Kappa-kaggle score”.

1.2 PROJECT PURPOSE

A practitioner using Convolutional Neural Networks (CNN) for the task of medical imaging is faced with a plethora of options when it comes to the training methodology for the CNN. Several factors can influence the decision making process including, but not limited to the size, noise level and quality of the dataset at hand, computational resources available and robustness of the trained CNN. The task of Diabetic Retinopathy detection, using a Convolutional neural Networks and Self-Supervised Learning, has great importance and use.

1.3 PROJECT FEATURES

Diabetic Retinopathy disease recognition has been one of the active and challenging research areas in the field of image processing. Deep learning technique as well as hinders to work with disease recognition and find the accuracy of the model. Graphical representation is showed using the accuracy of the model. The model is compared with other models and the accuracy of the model is determined using a graph.

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.2 PROBLEM DEFINITION

The goal of this project is to create a model that will be able to recognize and determine diabetic retinopathy from its image by using the concepts of Convolution Neural Network. Though the goal is to create a model which can recognize the diabetic retinopathy, it can be extended to detect the different stages of the disease. The major goal of the proposed system is understanding Convolutional Neural Network, and applying it to the medical imaging of diabetic retinopathy.

2.3 EXISTING SYSTEM

Unsupervised learning in general can be formulated as learning and embedding space, where the data that is similar semantically are closer and vice versa. The self-supervised learning does the same by constructing such representation space with the help of proxy task from the data itself. The learning’s of model at the time of proxy task can also be used in various other downstream tasks. Recently, several methods in the line of research have been developed and found applications in numerous fields.

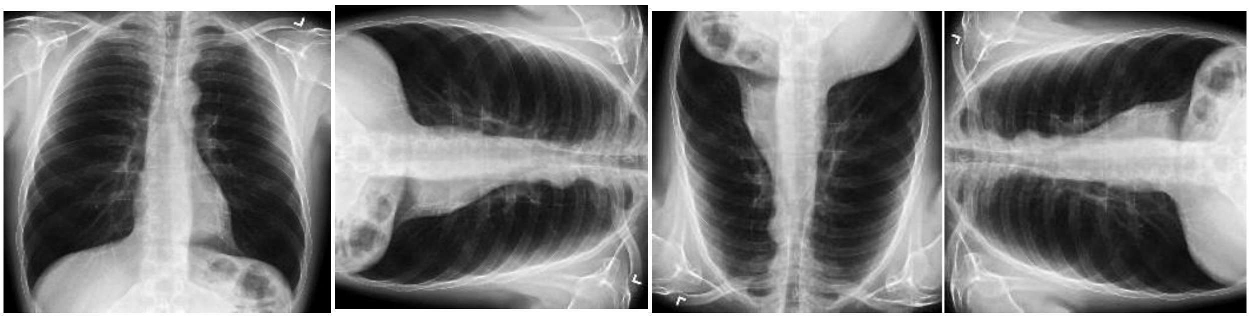
Self-supervised learning consists of two major parts of processing first is the proxy task and second is fine-tuning. There are different types of self-supervised learning methods that differ in their first building block i.e., proxy task. There are various types of proxy tasks developed in this line of research.

2.3.1 LIMITATIONS OF EXISTING SYSTEM

- You cannot get precise information regarding data sorting, and the output as data used in supervised learning is labelled and not known.
- Less accuracy of the results is because the input data is not known and not labelled by people in advance.

2.4 PROPOSED SYSTEM

The model follows a self-supervised paradigm and proposes to learn image representations by training to recognize the geometric transformation that is applied to the image that it gets as input. More specifically, we first define a small set of discrete geometric transformations, then each image on the dataset and the produced transformed images are fed to the model that is trained to recognize the transformation of each image



0 degrees 90 degrees 180 degrees 270 degrees

2.4.1 ADVANTAGES OF THE PROPOSED SYSTEM

- It requires much less labelled data
- It provides a significant improvement in accuracy
- We can reduce the need for expensive annotated data to build imageclassification models

2.4.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are:

1. Economic Feasibility
2. Technical Feasibility
3. Social Feasibility

2.4.3 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the formof reduced costs or fewer costly errors. Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.4.4 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.5 BEHAVIORAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system. This includes the following question:

- Will the proposed systemcause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• Processor	: INTEL CORE i5Processor
• Hard Disk	: 50 GB and Above.
• Input Devices	: Keyboard, Mouse.
• RAM	: 8GB and Above.

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

• Operating System	: WINDOWS XP
• Programming Language	: PYTHON
• Tools	: Anaconda, Google Colab.

ARCHITECTURE

3. ARCHITECTURE

3.1 CONVOLUTIONAL NEURAL NETWORKS

Deep artificial neural networks, such as convolutional neural networks, are a type of deep artificial neural networks. We use it to classify photographs (for example, naming what they see), cluster them by similarity (photo search), and recognise objects within scenes. It may also be used to recognise faces, people, street signs, cancers, platypuses, and a variety of other visual data. A CNN's basic building block is the convolutional layer. The layer's parameter is a set of learnable filters (or kernels) with a limited receptive field that spans the entire depth of the input volume. Each CNN filter is convolved over the width and height of the input volume during the forward pass, computing the dot product and providing a 2-dimensional activation map of the input volume.

As a result, the network learns when it sees a particular sort of feature at a particular spatial location in the input. The activation maps are then sent into a down sampling layer, which is applied one patch at a time, similar to convolutions. CNN also contains a fully connected layer that uses one label per node to classify output. Feature extraction and classification are the two fundamental components of the CNN architecture. Each layer of the network takes the output from the previous layer as input and provides the current output as input to the next layer in the feature extraction layers. Convolution, max-pooling, and classification are the three types of layers that make up the CNN architecture.

A. CONVOLUTION NETWORKS

The first layer to extract features from images is the convolutional layer. Convolution preserves the relationship between distinct regions of an image since pixels are only related to neighbouring and close pixels. Convolution is the process of reducing the size of a picture while maintaining the relationship between pixels by filtering it with a smaller pixel filter. We get a 3x3 output (64 percent reduction in complexity) when we apply convolution to a 5x5 image using a 3x3 filter with 1x1 stride (1-pixel shift at each step).

B. POOLING LAYER

Pooling layers are commonly inserted after each convolution layer in CNN to minimize the spatial size of the features maps. Overfitting can also be alleviated by pooling layers. By selecting the maximum, average, or total values within these pixels, we choose a pooling size to limit the number of parameters. One of the most prevalent pooling strategies is Max Pooling.

C. FULLY CONNECTED

Any design in which each parameter is linked to one another to determine the relationship and influence of each parameter on the labels is referred to as a fully connected network. We can build a fully connected network to identify the photos in the end since convolution and pooling layers reduce time-space complexity. We believe it is now appropriate to provide an overview of our proposed convolutional neural network. It is comparable to existing handwritten digit recognition architectures [1,6,8,10,11], but it has improved performance by changing a number of filters, neurons, and activation functions. It is made up of seven layers.

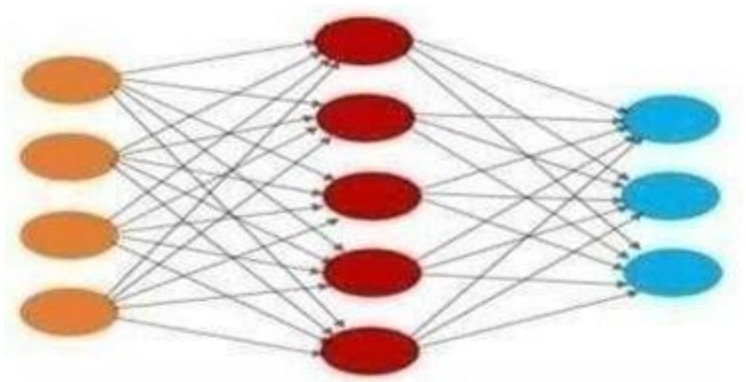


Fig 3.1 Neural networks

3.2 DENSENET 121

In short, Densenet -121 architecture of CNN is differed from the standards of CNN. The basic Operations and layers:

1 7*7 Convolution

58 3*3 Convolution

61 1*1 Convolution

4 Average Pool

1 Fully Connected layer

In a typical feed-forward Convolutional Neural Network (CNN), each convolutional layer receives the output of the previous convolutional layer and produces an output feature map that is then passed on to the next convolutional layer, with the exception of the first (which takes in the input). As a result, there are 'L' direct connections between each layer and the following layer for 'L' layers.

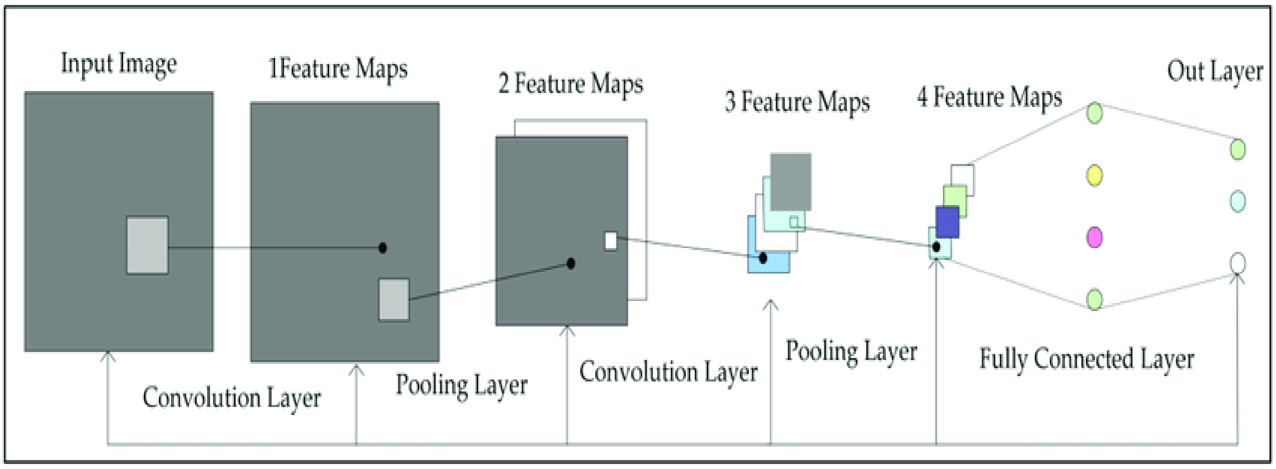


Fig 3.2 DenseNet 121

The 'vanishing gradient' problem emerges when the number of layers in the CNN increases, i.e. as they become deeper. This means that when the channel for information from the input to the output layers lengthens, certain data may 'disappear' or be lost, reducing the network's capacity to train effectively.

DenseNets solves this problem by streamlining the connectivity pattern between layers and altering the usual CNN design. Each layer in a DenseNet architecture is connected to every other layer directly, hence the name Densely Connected Convolutional Network. L(L+1)/2 direct connections exist for 'L' levels.

3.3 DENSENET COMPONENTS:

Components of DenseNet include:

- Connectivity
- DenseBlocks
- Growth Rate
- Bottleneck layers

A. Connectivity

The feature maps of all preceding layers are concatenated and used as inputs in each layer, rather than being summed. As a result, DenseNets require fewer parameters than a typical CNN, allowing for feature reuse by discarding redundant feature maps. As a result, the feature-maps of all preceding layers, x0,...,xl-1, are fed into the lth layer:

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]).$$

where $[x_0,x_1,...,x_{l-1}]$ is the concatenation of the feature-maps, i.e. the output produced in all the layers preceding l (0,...,l-1). The multiple inputs of H_l are concatenated into a single tensor to ease implementation.

B. Dense Blocks

When the size of feature maps varies, it is not possible to use the concatenation method. However, down-sampling of layers, which minimises the size of feature maps through dimensionality reduction to enable faster calculation speeds, is an important aspect of CNNs.

DenseNets are divided into DenseBlocks to do this, with the dimensions of the feature maps remaining constant inside a block but the number of filters between them changing. Transition Layers are the layers between the blocks that lower the number of channels to half of what they were before.

Hl is defined as a composite function that applies three sequential operations for each layer, as shown in the equation above: batch normalisation (BN), a rectified linear unit (ReLU), and a convolution (Conv). Three dense blocks are presented in a deep DenseNet. The transition layers are the layers that execute downsampling (i.e. changing the size of the feature maps) via convolution and pooling operations between two adjacent blocks, whereas the feature maps within the dense block are the same size to allow feature concatenation.

C. Growth Rate

The features can be thought of as the network's overall condition. After passing through each thick layer, the feature map expands in size, with each layer contributing 'K' features on top of the global state (existing features). This parameter, 'K,' is referred to as the network's growth rate, and it controls the quantity of data added to each layer of the network. The lth layer has input feature-maps if each function H l provides k feature maps, where k0 is the number of channels in the input layer. DenseNets, unlike other network architectures, can have very narrow layers.

$$k_l = k_0 + k * (l - 1)$$

D. Bottleneck layers

Despite the fact that each layer only produces k output feature-maps, the total number of inputs can be fairly large, especially when more levels are added. To improve the efficiency and speed of computations, a 1x1 convolution layer can be included as a bottleneck layer before each 3x3 convolution.

In comparison to their regular CNN or ResNet equivalents, DenseNets require fewer parameters and allow feature reuse, resulting in more compact models with state-of-the-art performances and superior outcomes across competing datasets.

3.4 PROJECT ARCHITECTURE

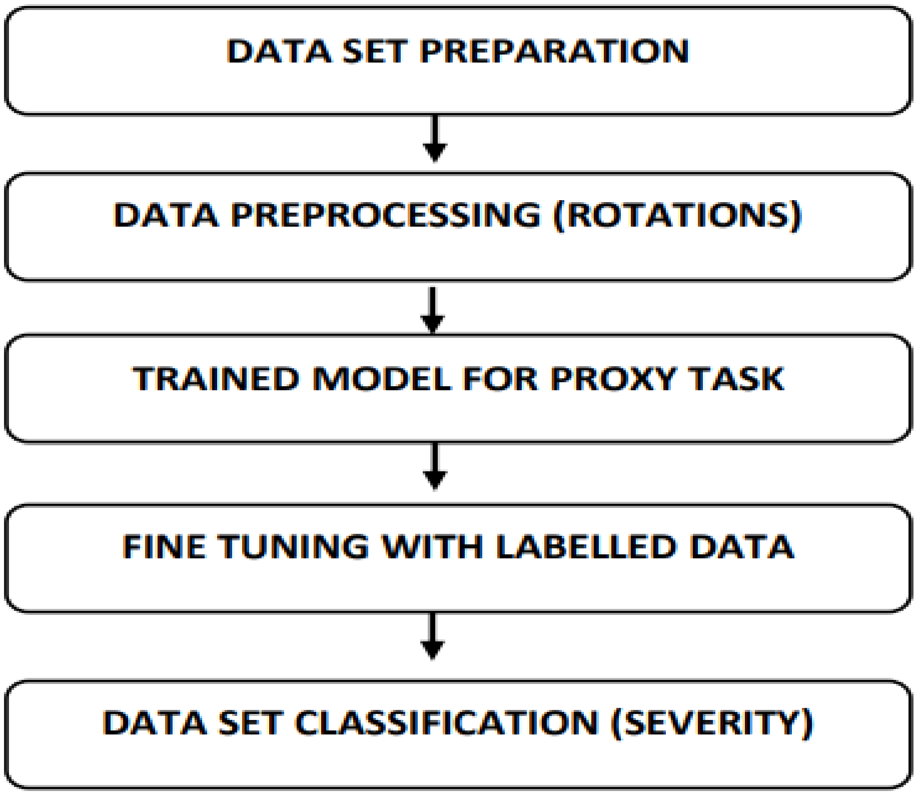

DATA SET PREPARATION
DATA PREPROCESSING (ROTATIONS)
TRAINED MODEL FOR PROXY TASK
FINE TUNING WITH LABELLED DATA
DATA SET CLASSIFICATION (SEVERITY)

Fig.3.4 Project Architecture

To develop a self-supervised learning model for medical imaging, specifically for Diabetic Retinopathy detection. We employ the Rotation approach as a proxy task, and we use unlabeled data with various geometric progressions to train a Dense ConvNet model to predict geometric progression probability. Then, as follows, fine-tune the model to determine the severity of the DR using some tagged data.

i. NO DR

ii. MILD DR

iii. MODERATE DR

iv. SEVERE DR

v. PROLIFERATE DR

The Figure basic deals with the architecture diagram of the proposed system. The proposed model is divided into four different stages in order to classify and detect the digits:

A. Dataset Preparation

B. Data Preprocessing

C. Fine-Tuning with Labelled Data

D. Classification

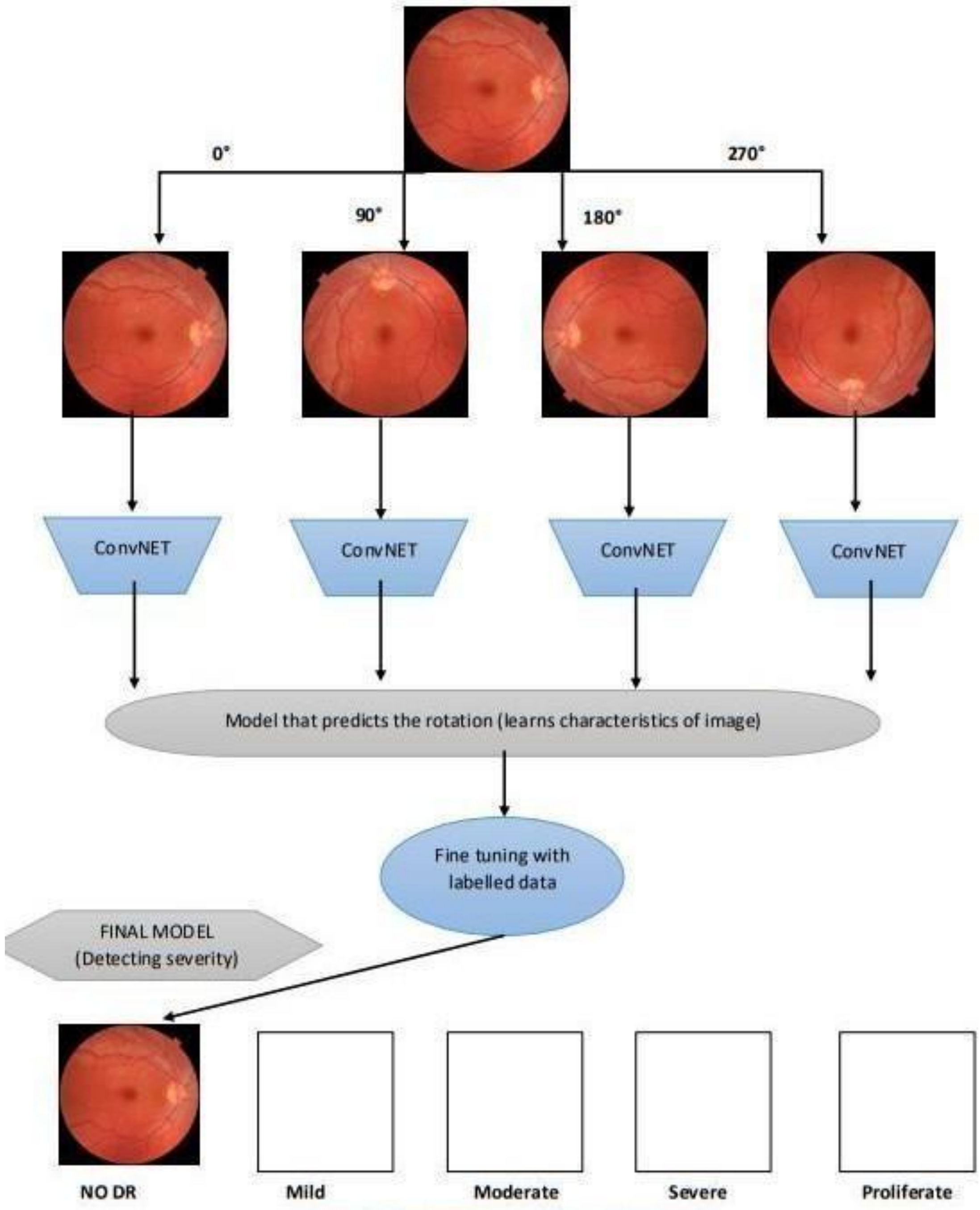

0°
270°
90°
180°
ConvNET
ConvNET
ConvNET
ConvNET
Model that predicts the rotation (learns characteristics of image)
Fine tuning with labelled data
FINAL MODEL (Detecting severity)
NO DR
Mild
Moderate
Severe
Proliferate

Figure 3: Representation of full working model.

Fig 3.4.1 Working model of Self-supervised learning for Medical Imaging

3.5 MODULE DESCRIPTION

A. DATASET PREPERATION

The information was gathered via Kaggle, which was mentioned in the Diabetic Retinopathy 2019 Kaggle challenge. It provides 224 x 224 photos of the retinal fundus, divided into five types: NO DR, mild, moderate, severe, and proliferate. We mix all of the types for the proxy task because it does not require any labelling. We employ 5%, 10%, 25%, and 50% of the data to finetune the efficiency for each given size of data.

The process of data preparation is an important aspect of data science. Data Cleaning and Feature Engineering are two topics included. These two are required for Machine Learning and Deep Learning projects to get higher accuracy and performance.

B. DATA PREPROCESSING

Data preprocessing is a method for converting unclean data into a clean data set. In other words, whenever data is gathered from various sources, it is collected in raw format, which makes analysis impossible. As a result, certain procedures are followed in order to convert the data into a tiny, clean data set. This strategy is used prior to performing the Iterative Analysis. Data Preprocessing is the term for the sequence of steps. It consists of –

- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

The data was downsized to 244 × 244 pixels and subjected to different geometric transformations, such as rotations into multiples of 90 degrees, and so on (0, 90, 180, 270 degrees). With the DenseNET121 encoder architecture, this pre-processed data is supplied to the ConvNET model. It was treated with a learning rate of 1e-5 and a batch size of 32 for 200 epochs.

C. FINE-TUNING WITH LABELLED DATA

The data was downsized to 244 × 244 pixels and subjected to different geometric transformations, such as rotations into multiples of 90 degrees, and so on (0, 90, 180, 270 degrees). With the DenseNET121 encoder architecture, this pre-processed data is supplied to the ConvNET model. It was treated with a learning rate of 1e-5 and a batch size of 32 for 200 epochs.

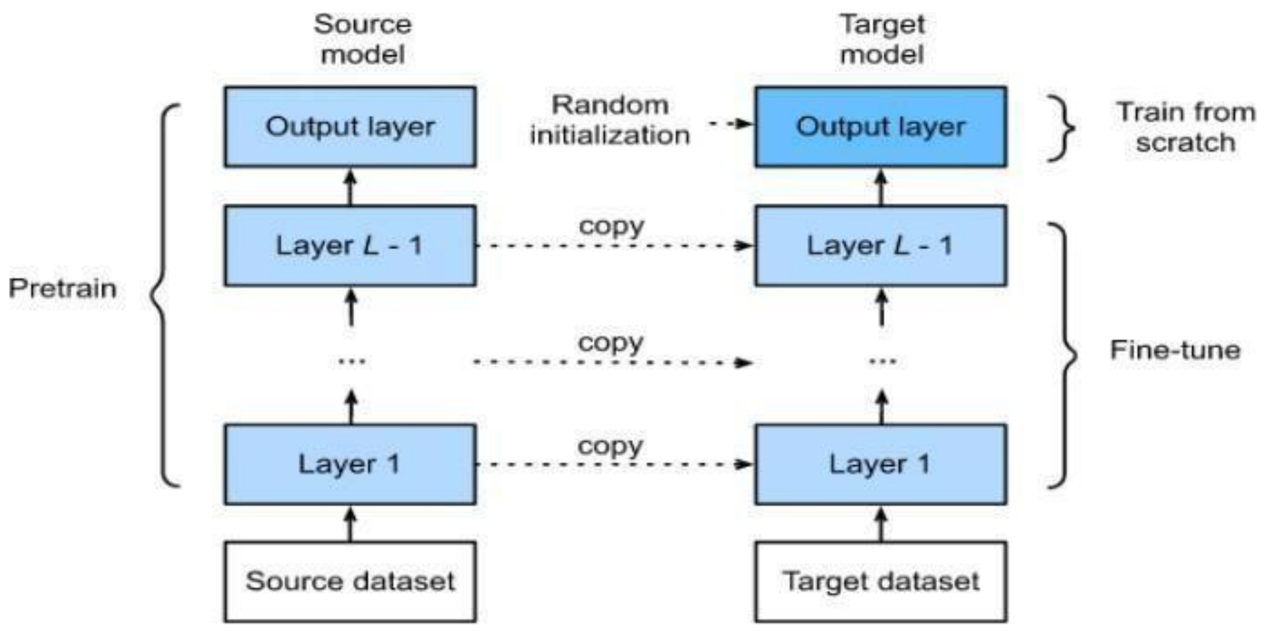


Fig 3.5 Fine-tuning with labeled data

D. CLASSIFICATION

After further tuning, the final model is created, which can classify or identify Diabetic retinopathy stages. It was put to the test using "qw kappa kaggle" scores based on accuracy, and the results were quite promising when compared to dataset size.

The quadratic weighted kappa, which evaluates the agreement between two ratings, is used to score submissions. This statistic normally ranges from 0 (random agreement between raters) to 1 (perfect agreement between raters) (complete agreement between raters). The metric may fall below 0 if there is less agreement amongst the raters than expected by chance. Between the expected/known scores and the anticipated scores, the quadratic weighted kappa is determined. There are five possible ratings for the results: 0,1,2,3,4. This is how you calculate the quadratic weighted kappa.

To begin, a N x N histogram matrix O is created, with Oi,j corresponding to the number of adoption records with an actual rating of I and a forecasted rating of j. Based on the difference between actual and anticipated rating scores, an N-by-N matrix of weights, w, is produced. Assuming that there is no association between rating scores, an N-by-N histogram matrix of expected ratings, E, is produced. This is determined as the outer product of the histogram vector of ratings in the actual rating and the histogram vector of ratings in the predicted rating, normalised so that E and O have the same sum.

Weighted Kappa Metric has four steps:

- ❖ To begin, make a multi-class confusion matrix O that compares expected and actual scores.

- Second, create a weight matrix w that calculates the difference in weight between actual and anticipated scores.
- Calculate value counts() for each rating in preds and actuals in the third step.
- Calculate E, which is the outer product of two value count vectors, in the fourth step.

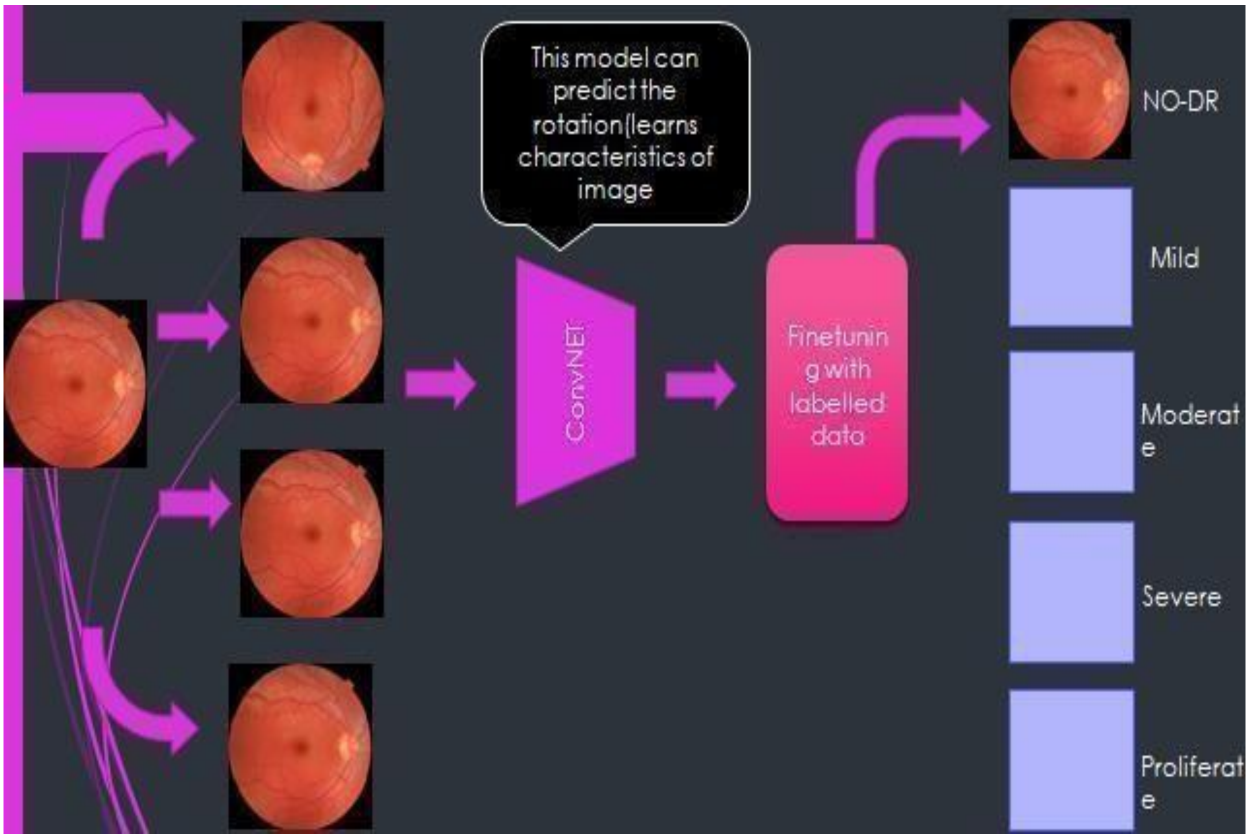

This model can predict the rotation(learns characteristics of image
ConvNET
Finetuning with labelled data
NO-DR
Mild
Moderate
Severe
Proliferate

Fig 3.5.2 Classification and Working model

3.6 DIABETIC RETINOPATHY

Diabetic retinopathy is an eye disease that affects people who have diabetes. When high blood sugar levels damage the blood vessels in the retina, this is known as diabetic retinopathy. These blood vessels have the potential to enlarge and leak. They can also close, preventing blood from flowing through. On the retina, aberrant new blood vessels can form. All of these changes have the potential to blur your eyesight.

Diabetics are estimated to affect roughly 600 million individuals worldwide by 2040. Diabetic retinopathy, the most common cause of vision loss in working-age adults around the world, is predicted to affect about one-third of them. Mild diabetic retinopathy (MDR) is an early stage of diabetic retinopathy that can be identified by the presence of microaneurysms. The later stages of the disease might result in serious eyesight loss. As a result, early detection is critical on a large scale and can be treated to reduce visual loss. However, the ratio of ophthalmologists to patients is large.

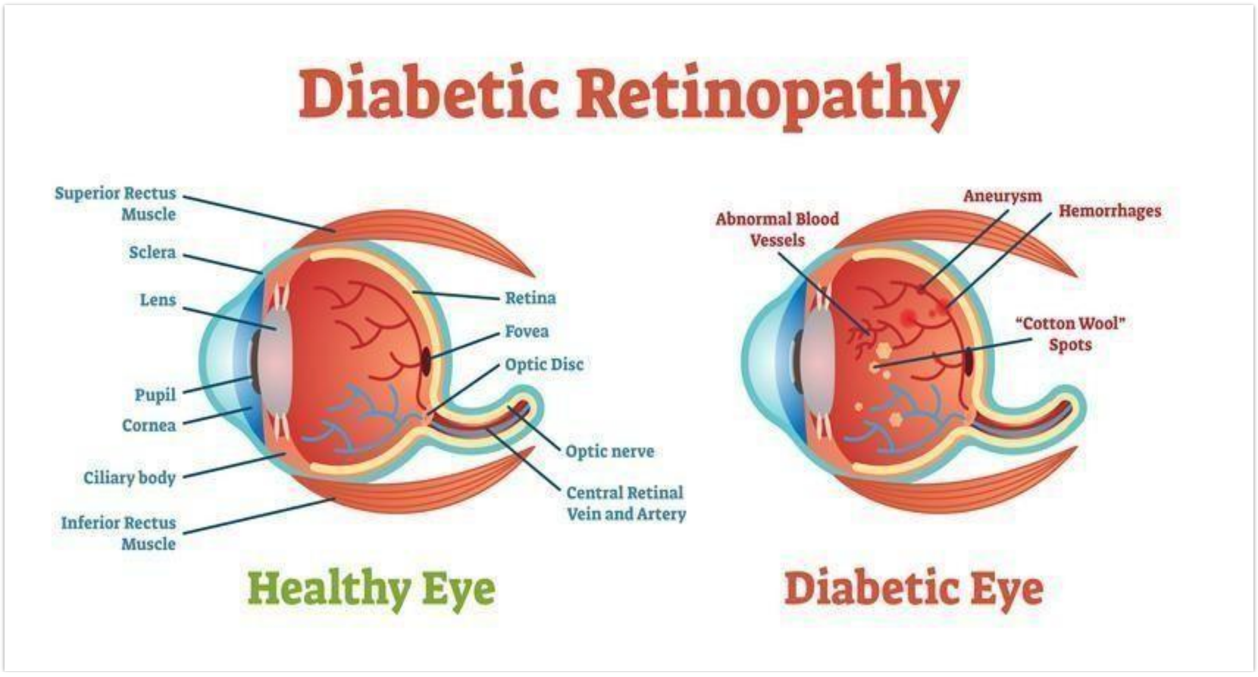


Fig 3.6 Diabetic Retinopathy

3.7 SELF SUPERVISED LEARNING

Supervised learning is a sort of machine learning in which models are trained with well-labeled data and the output is predicted by the model. Providing large amounts of labelled data isn't always cost or time efficient, because supervised learning requires a lot of labelled data. Unlabeled data, on the other hand, is readily available. This increases motivation for both unsupervised and self-supervised learning. The self-supervised model uses proxy tasks to learn usable data representations from an unlabeled pool of data, then fine-tunes the model with a few labels for the supervised downstream job. It can handle simple tasks like picture classification as well as more complicated tasks like semantic segmentation.

- ❖ Using proxy tasks, the self-supervised model learns usable representations of the data from an unlabeled pool of data, and then fine-tunes the model for the supervised downstream job with a few labels.
- ❖ It can handle simple tasks like picture classification as well as more complicated tasks like semantic segmentation.

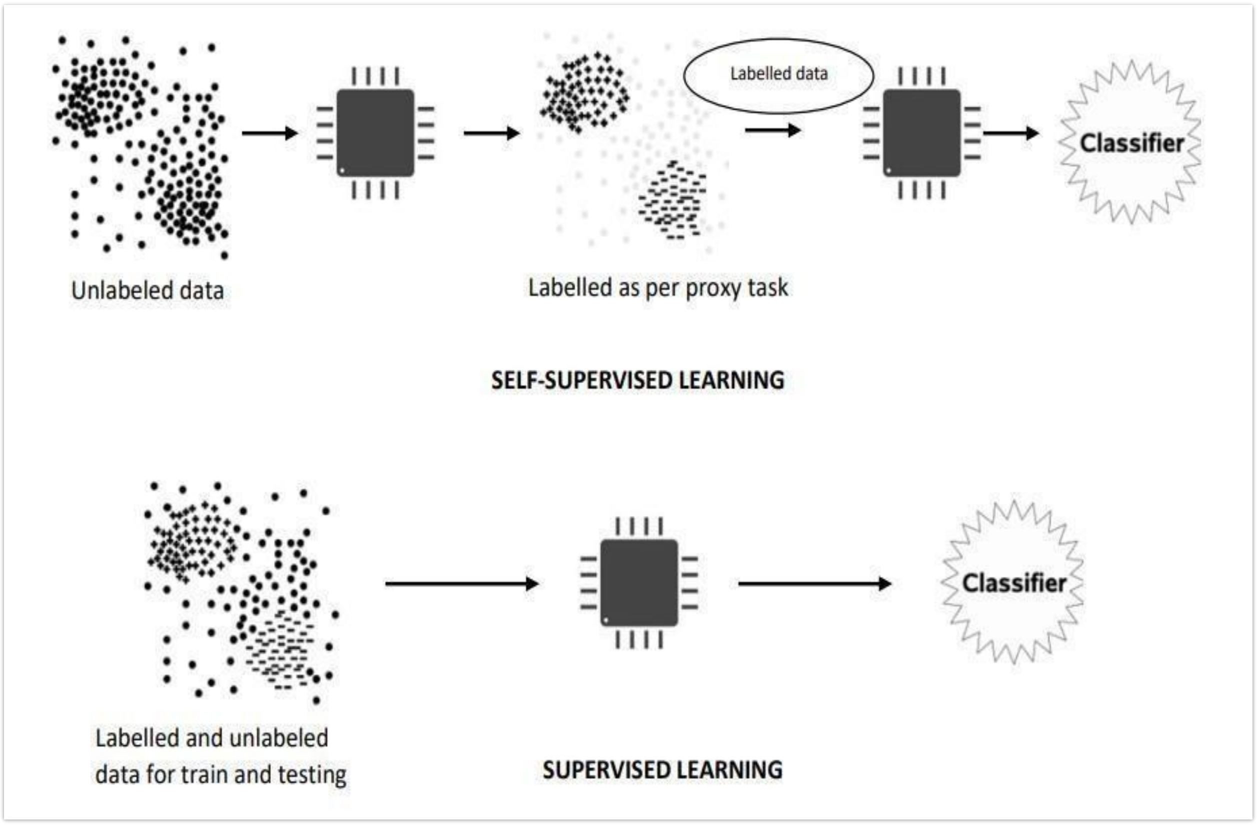


Fig 3.7 Self-Supervised Learning vs Supervised Learning

3.8 USECASE DIAGRAM

In the use case diagram we have basically two actors who are the user and the system. The user has the rights to login, access to resources and to view the details. Whereas the system has the login, access to resources of the users and also the right to update and remove the details, and he can also view the user files.

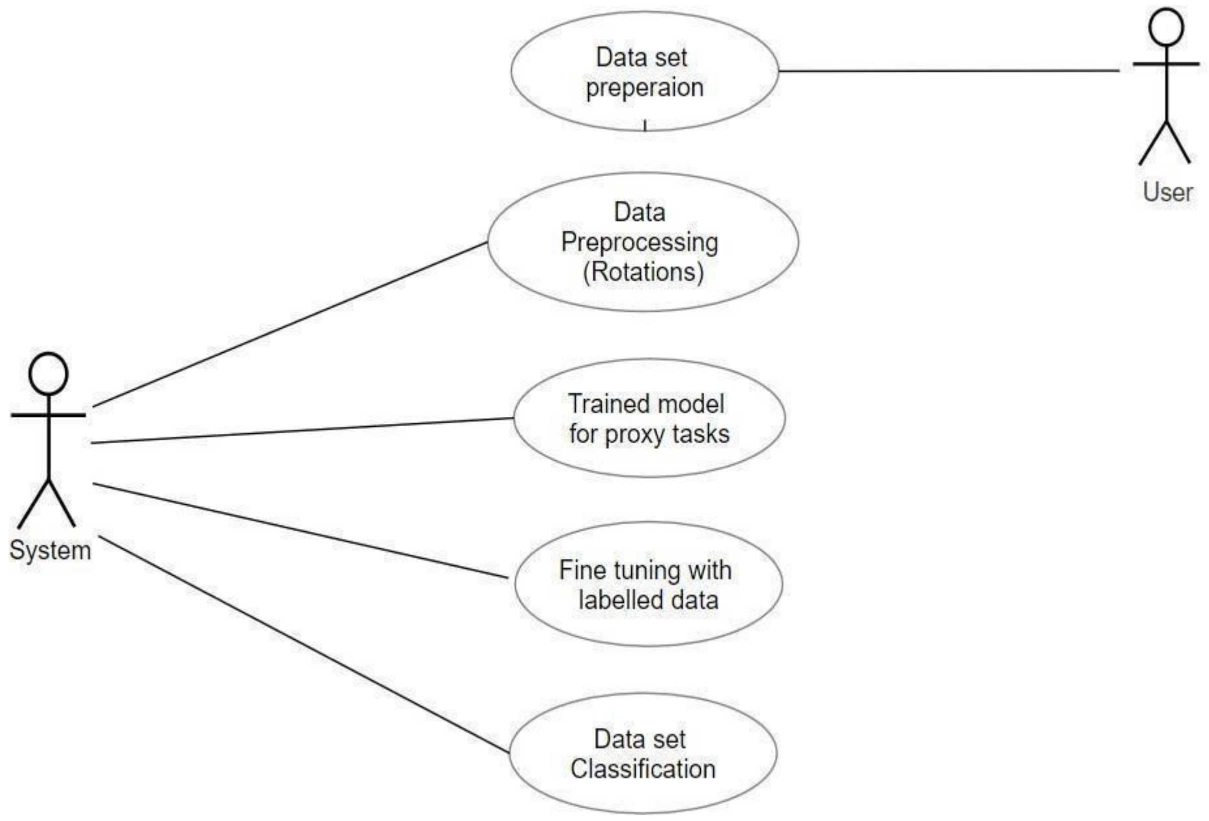

Data set preperaion
User
Data Preprocessing (Rotations)
System
Trained model for proxy tasks
Fine tuning with labelled data
Data set Classification

Fig.3.8 Usecase Diagram for Self supervised learning for Medical Imaging

3.9 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

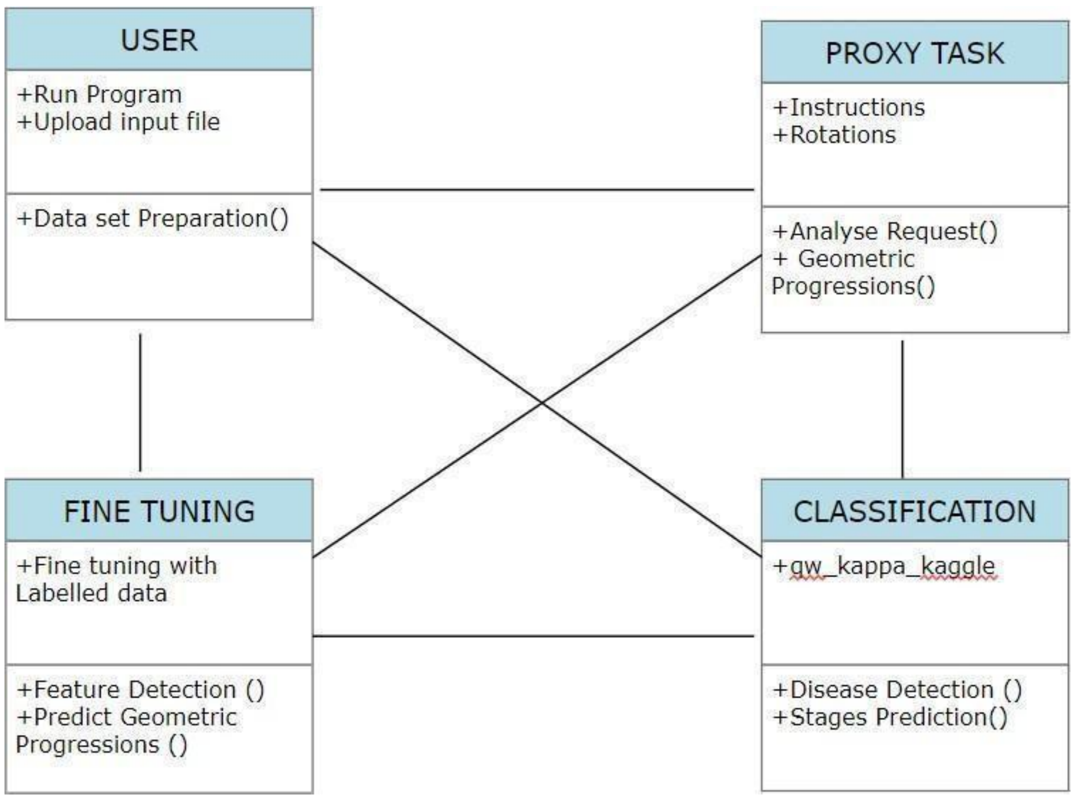


Fig.3.9 Class Diagram for Self-supervised learning for medical Imaging

3.10 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence.

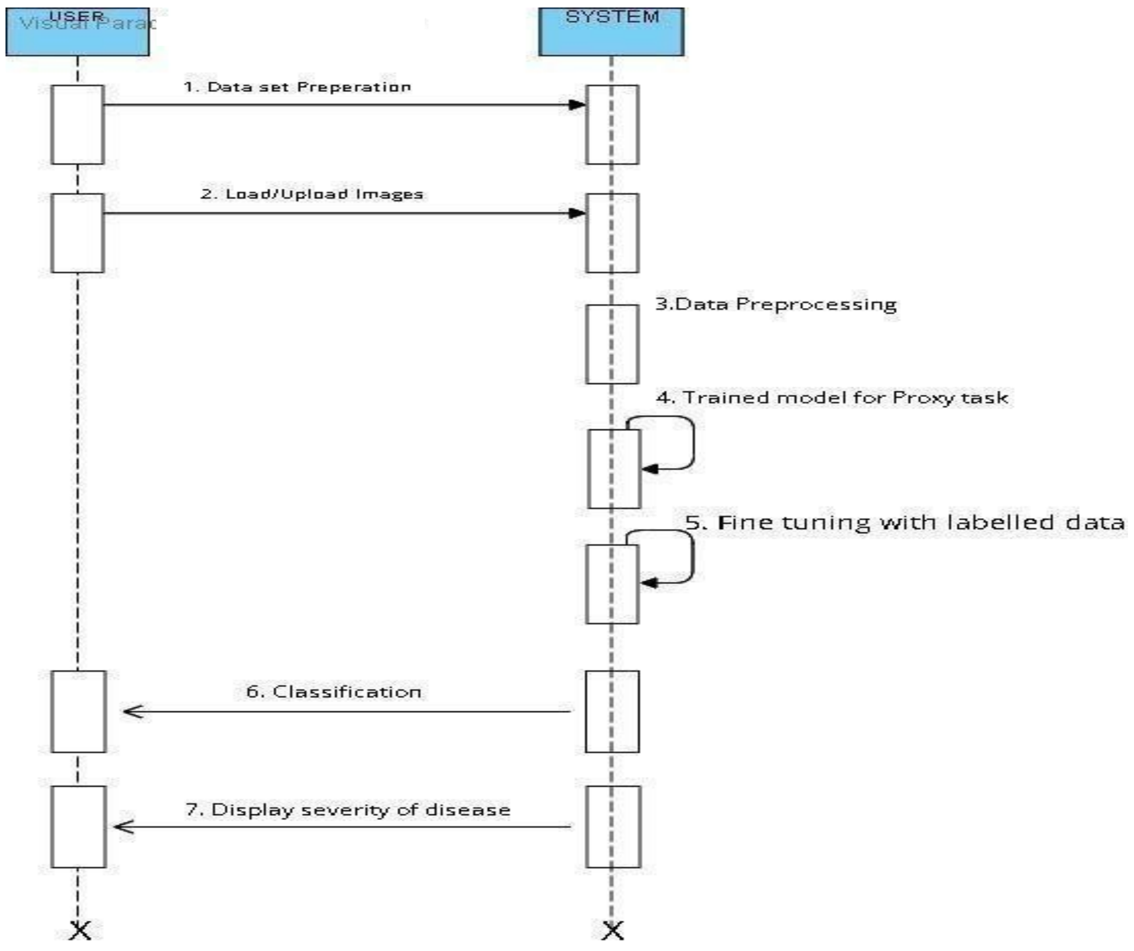


Fig.3.10 Sequence Diagram for Self-supervised learning for medical Imaging

3.11 ACTIVITY DIAGRAM

It describes about flow of activity states.

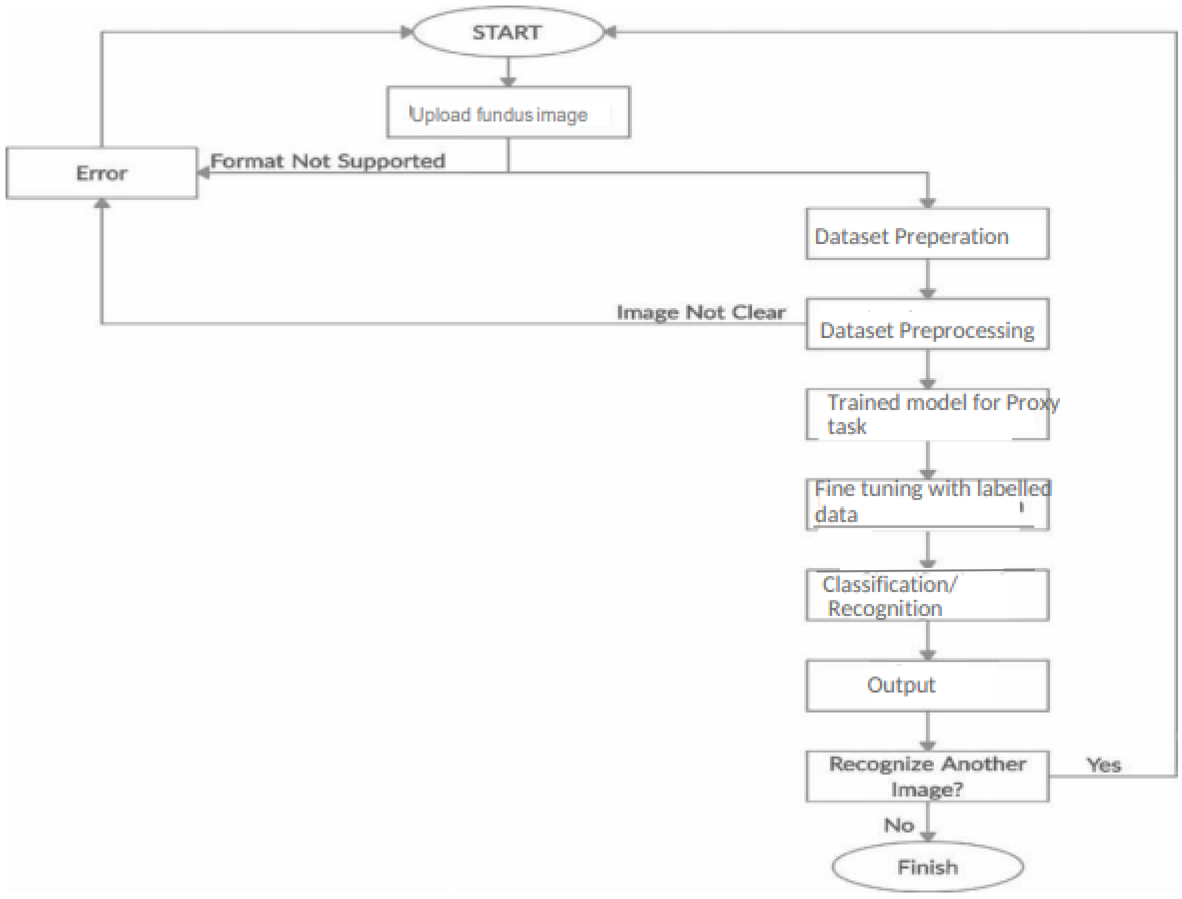
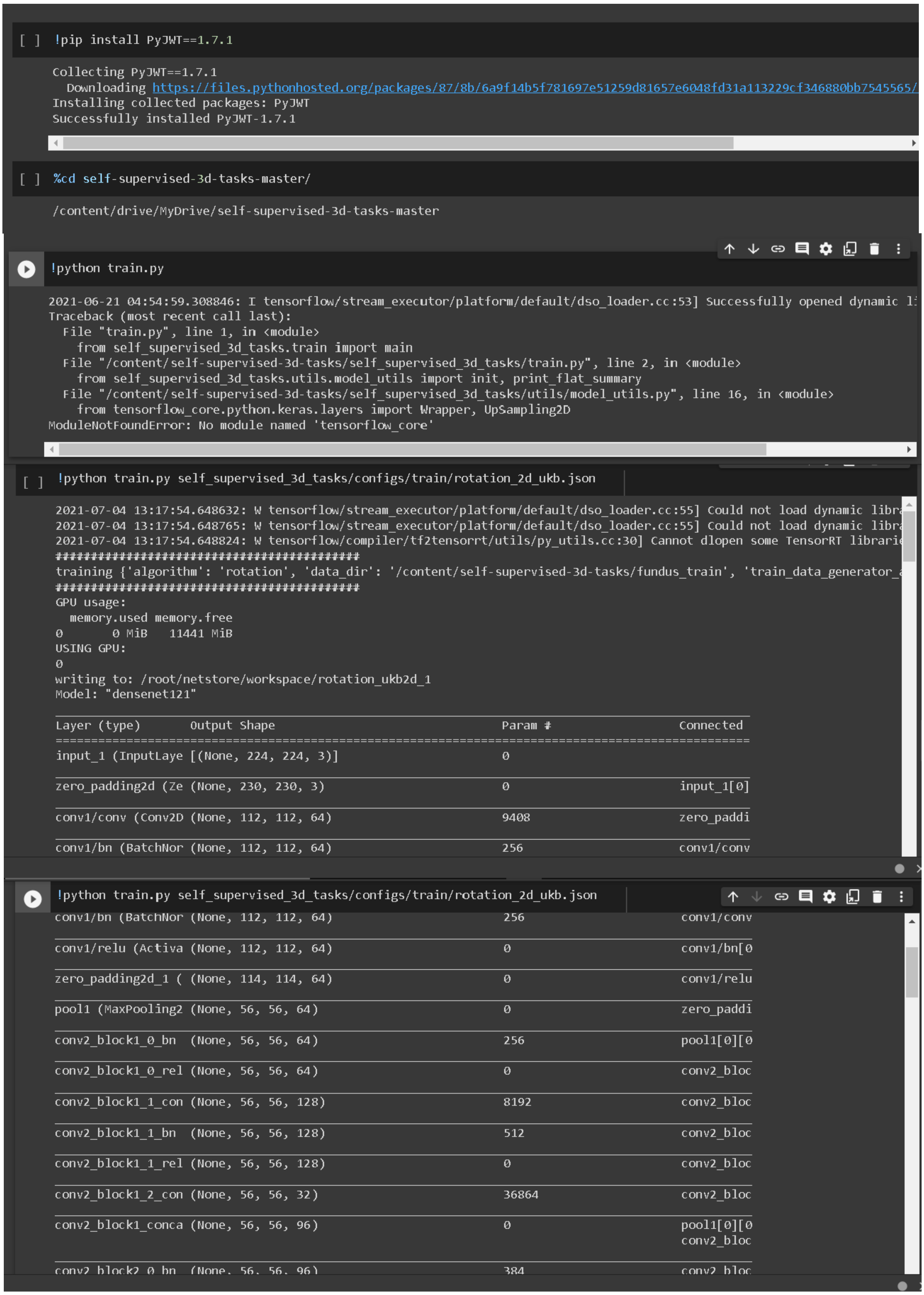


Fig.3.11 Activity Diagram User for Self-supervised learning for medical imaging

IMPLEMENTATION

4.1 SAMPLE CODE



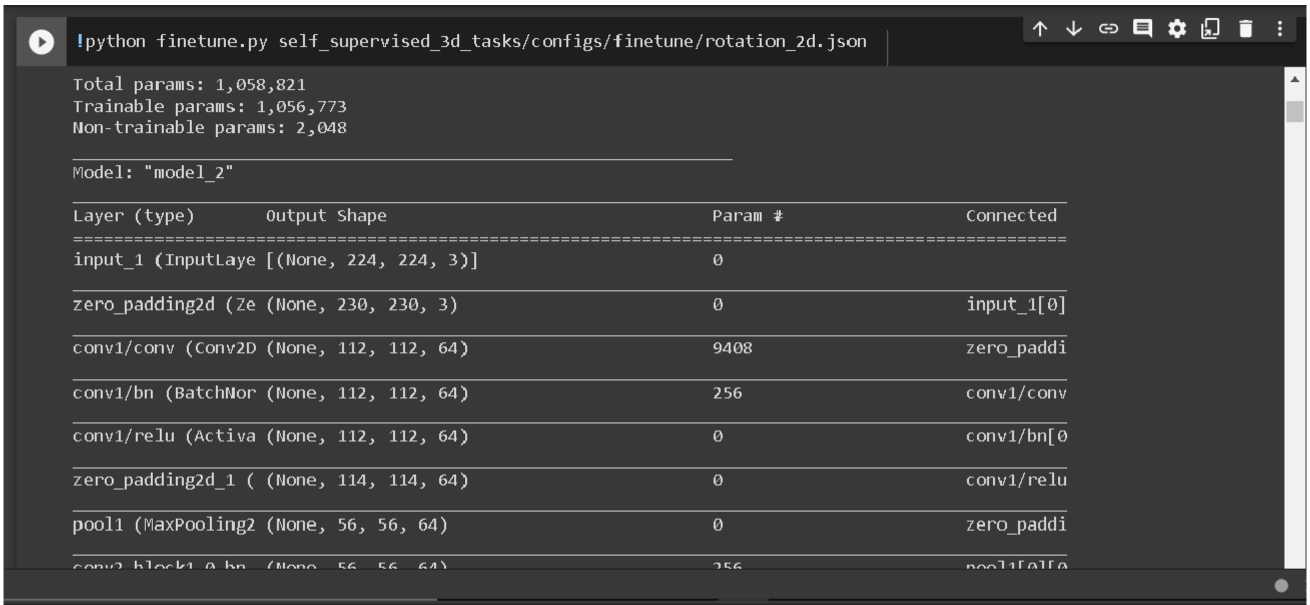
```
[ ] !pip install PyJWT==1.7.1

    Collecting PyJWT==1.7.1
      Downloading https://files.pythonhosted.org/packages/87/8b/6a9f14b5f781697e51259d81657e6048fd31a113229cf346880bb7545565/
    Installing collected packages: PyJWT
    Successfully installed PyJWT-1.7.1
```

```
[ ] %cd self-supervised-3d-tasks-master/

    /content/drive/MyDrive/self-supervised-3d-tasks-master
```

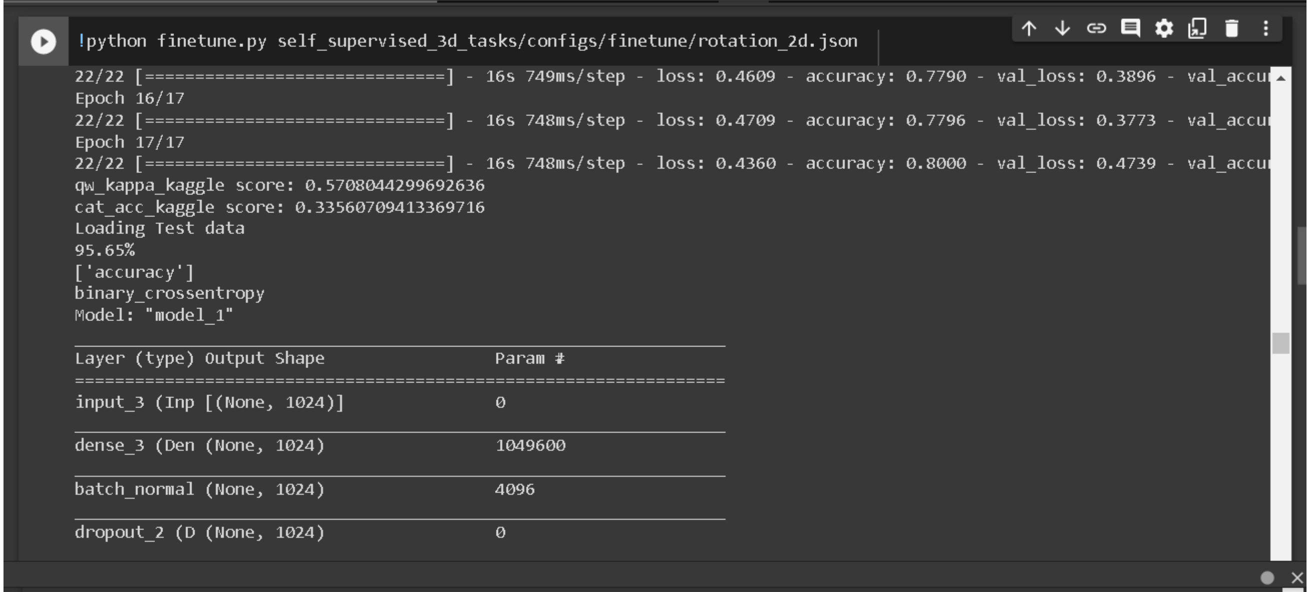
```
!python train.py

2021-06-21 04:54:59.308846: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic li
Traceback (most recent call last):
  File "train.py", line 1, in <module>
    from self_supervised_3d_tasks.train import main
  File "/content/self-supervised-3d-tasks/self_supervised_3d_tasks/train.py", line 2, in <module>
    from self_supervised_3d_tasks.utils.model_utils import init, print_flat_summary
  File "/content/self-supervised-3d-tasks/self_supervised_3d_tasks/utils/model_utils.py", line 16, in <module>
    from tensorflow_core.python.keras.layers import Wrapper, UpSampling2D
ModuleNotFoundError: No module named 'tensorflow_core'
```

```
[ ] !python train.py self_supervised_3d_tasks/configs/train/rotation_2d_ukb.json

    2021-07-04 13:17:54.648632: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic libra
    2021-07-04 13:17:54.648765: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic libra
    2021-07-04 13:17:54.648824: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:30] Cannot dlopen some TensorRT librarie
    ###########################################
    training {'algorithm': 'rotation', 'data_dir': '/content/self-supervised-3d-tasks/fundus_train', 'train_data_generator_a
    ###########################################
    GPU usage:
      memory.used memory.free
    0       0 MiB   11441 MiB
    USING GPU:
    0
    writing to: /root/netstore/workspace/rotation_ukb2d_1
    Model: "densenet121"
    _______________________________________________________________________________________________
    Layer (type)         Output Shape                                  Param #                Connected
    ===============================================================================================
    input_1 (InputLaye [(None, 224, 224, 3)]                           0
    _______________________________________________________________________________________________
    zero_padding2d (Ze (None, 230, 230, 3)                             0                      input_1[0]
    _______________________________________________________________________________________________
    conv1/conv (Conv2D (None, 112, 112, 64)                            9408                   zero_paddi
    _______________________________________________________________________________________________
    conv1/bn (BatchNor (None, 112, 112, 64)                            256                    conv1/conv
```

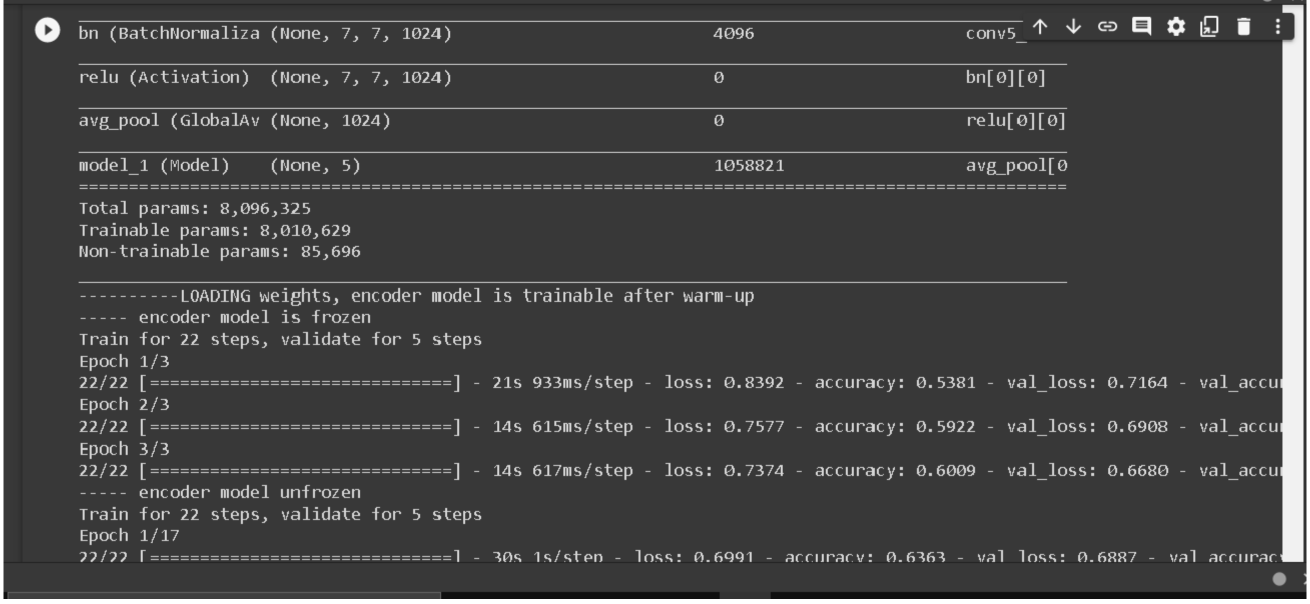
```
!python train.py self_supervised_3d_tasks/configs/train/rotation_2d_ukb.json
    conv1/bn (BatchNor (None, 112, 112, 64)                            256                    conv1/conv
    _______________________________________________________________________________________________
    conv1/relu (Activa (None, 112, 112, 64)                            0                      conv1/bn[0
    _______________________________________________________________________________________________
    zero_padding2d_1 ( (None, 114, 114, 64)                            0                      conv1/relu
    _______________________________________________________________________________________________
    pool1 (MaxPooling2 (None, 56, 56, 64)                              0                      zero_paddi
    _______________________________________________________________________________________________
    conv2_block1_0_bn  (None, 56, 56, 64)                              256                    pool1[0][0
    _______________________________________________________________________________________________
    conv2_block1_0_rel (None, 56, 56, 64)                              0                      conv2_bloc
    _______________________________________________________________________________________________
    conv2_block1_1_con (None, 56, 56, 128)                             8192                   conv2_bloc
    _______________________________________________________________________________________________
    conv2_block1_1_bn  (None, 56, 56, 128)                             512                    conv2_bloc
    _______________________________________________________________________________________________
    conv2_block1_1_rel (None, 56, 56, 128)                             0                      conv2_bloc
    _______________________________________________________________________________________________
    conv2_block1_2_con (None, 56, 56, 32)                              36864                  conv2_bloc
    _______________________________________________________________________________________________
    conv2_block1_conca (None, 56, 56, 96)                              0                      pool1[0][0
                                                                                              conv2_bloc
    _______________________________________________________________________________________________
    conv2_block2_0_bn  (None, 56, 56, 96)                              384                    conv2_bloc
```



```
!python finetune.py self_supervised_3d_tasks/configs/finetune/rotation_2d.json

Total params: 1,058,821
Trainable params: 1,056,773
Non-trainable params: 2,048
__________________________________________________________________
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)       Output Shape                             Param #          Connected
==================================================================================================
input_1 (InputLaye [(None, 224, 224, 3)]                    0

zero_padding2d (Ze (None, 230, 230, 3)                      0                input_1[0]

conv1/conv (Conv2D (None, 112, 112, 64)                     9408             zero_paddi

conv1/bn (BatchNor (None, 112, 112, 64)                     256              conv1/conv

conv1/relu (Activa (None, 112, 112, 64)                     0                conv1/bn[0

zero_padding2d_1 ( (None, 114, 114, 64)                     0                conv1/relu

pool1 (MaxPooling2 (None, 56, 56, 64)                       0                zero_paddi

conv2_block1_0_bn  (None, 56, 56, 64)                       256              pool1[0][0
```



```
!python finetune.py self_supervised_3d_tasks/configs/finetune/rotation_2d.json
22/22 [==============================] - 16s 749ms/step - loss: 0.4609 - accuracy: 0.7790 - val_loss: 0.3896 - val_accu
Epoch 16/17
22/22 [==============================] - 16s 748ms/step - loss: 0.4709 - accuracy: 0.7796 - val_loss: 0.3773 - val_accu
Epoch 17/17
22/22 [==============================] - 16s 748ms/step - loss: 0.4360 - accuracy: 0.8000 - val_loss: 0.4739 - val_accu
qw_kappa_kaggle score: 0.5708044299692636
cat_acc_kaggle score: 0.33560709413369716
Loading Test data
95.65%
['accuracy']
binary_crossentropy
Model: "model_1"
_________________________________________________________________
Layer (type) Output Shape                    Param #
=================================================================
input_3 (Inp [(None, 1024)]                  0
_________________________________________________________________
dense_3 (Den (None, 1024)                    1049600
_________________________________________________________________
batch_normal (None, 1024)                    4096
_________________________________________________________________
dropout_2 (D (None, 1024)                    0
```



```
bn (BatchNormaliza (None, 7, 7, 1024)                       4096             conv5_
__________________________________________________________________________________________________
relu (Activation)  (None, 7, 7, 1024)                       0                bn[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAv (None, 1024)                             0                relu[0][0]
__________________________________________________________________________________________________
model_1 (Model)    (None, 5)                                1058821          avg_pool[0
==================================================================================================
Total params: 8,096,325
Trainable params: 8,010,629
Non-trainable params: 85,696
__________________________________________________________________________________________________
----------LOADING weights, encoder model is trainable after warm-up
----- encoder model is frozen
Train for 22 steps, validate for 5 steps
Epoch 1/3
22/22 [==============================] - 21s 933ms/step - loss: 0.8392 - accuracy: 0.5381 - val_loss: 0.7164 - val_accu
Epoch 2/3
22/22 [==============================] - 14s 615ms/step - loss: 0.7577 - accuracy: 0.5922 - val_loss: 0.6908 - val_accu
Epoch 3/3
22/22 [==============================] - 14s 617ms/step - loss: 0.7374 - accuracy: 0.6009 - val_loss: 0.6680 - val_accu
----- encoder model unfrozen
Train for 22 steps, validate for 5 steps
Epoch 1/17
22/22 [==============================] - 30s 1s/step - loss: 0.6991 - accuracy: 0.6363 - val_loss: 0.6887 - val_accurac
```

RESULTS

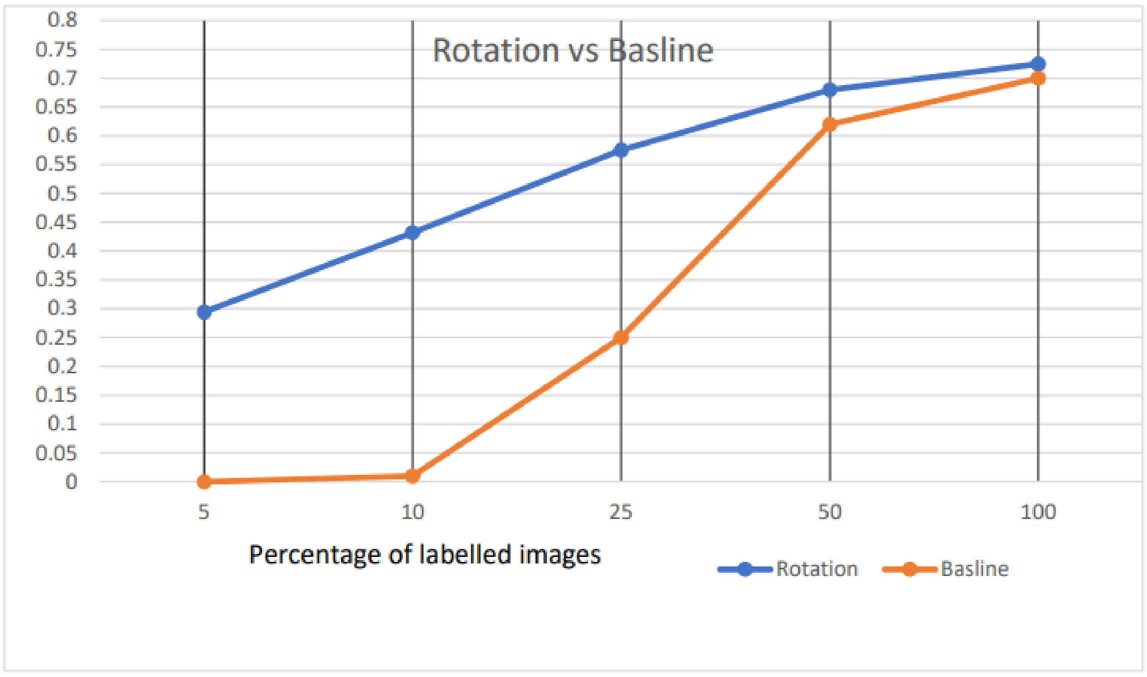
5. RESULTS

The final model recognises and categorises the retinal fundus data into one of the five kinds discussed before. The Kaggle dataset contains approximately 3600 photos, each of which has been scored on a scale of 0 to 4 by a clinician (NO DR, mild, moderate, severe, proliferate). To assess our performance on this benchmark, we pre-trained the model using all of the dataset's photos. Then they were fine-tuned on the same Kaggle data but with varied subset sizes, resulting in a data-efficient evaluation. When compared to other transfer learning methods that use a big corpus, the outcomes due to data efficient evaluation are not up to par. The dataset is being tested using 5-fold cross validation. The task's statistic is quadratic weighted kappa, which determines how well two ratings agree. Its values range from random (0) to total (1) agreement, and it can become negative if there is less agreement than chance.

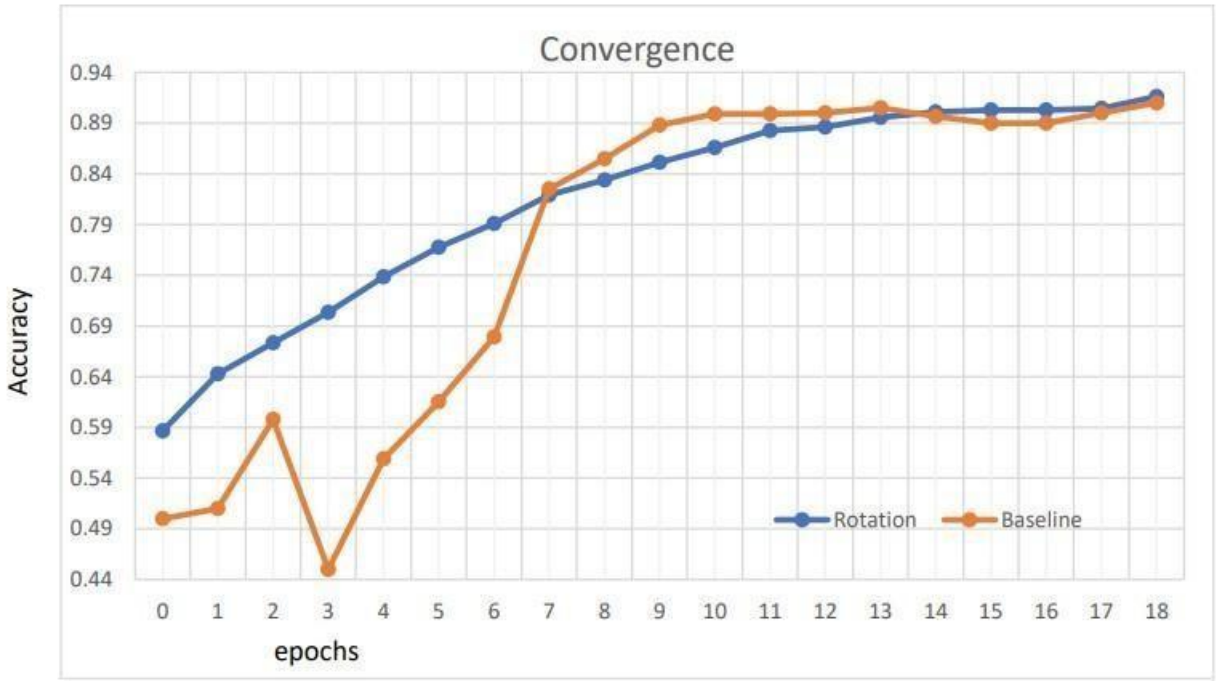
Train Split	**Qw_kappa_kaggle MIN**	**Qw_kappa_kaggle AVG**	**Qw_kappa_kaggle MAX**
10%	0.2888881102	0.4321430821	0.5084661884
5%	0.1751079345	0.2944362057	0.4669641719
50%	0.6116147969	0.6798179485	0.7365178391
25%	0.4738074393	0.5753686169	0.6937023326
100%	0.6955872731	0.7247486635	0.7555889924

Results obtained from the model, showing minimum, average and maximum quadratic weighted kappa scores for different subsets of data used in fine-tuning.

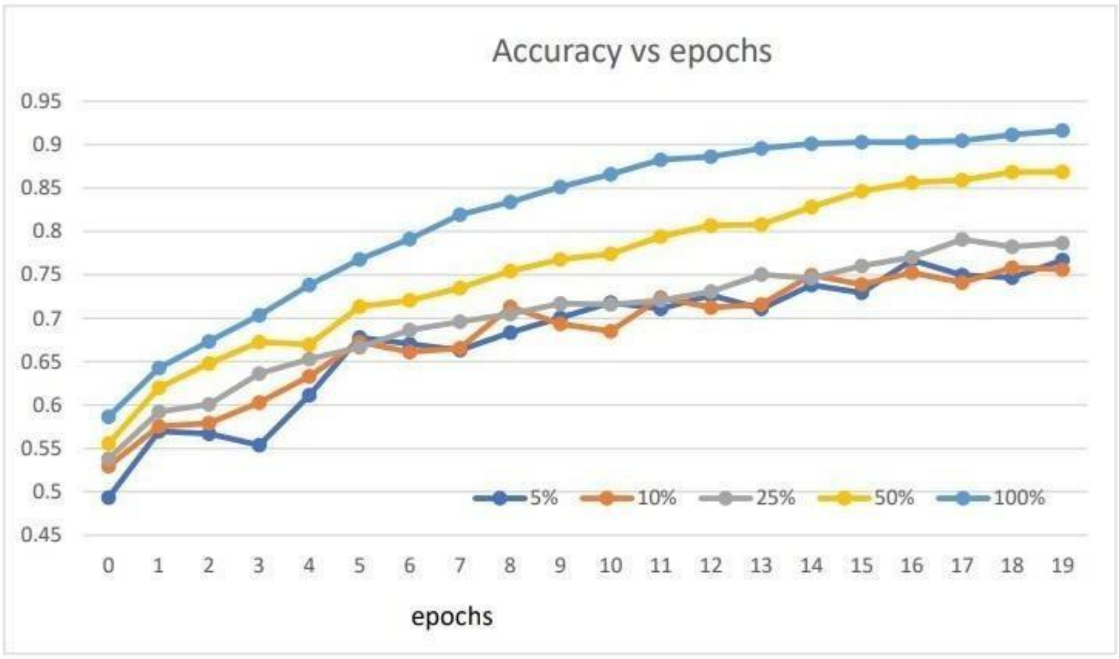
Avg QW kappa scores vs percentage of labelled images comparing Rotation technique and baseline values:



Convergence Rates comparision:



Accuracy vs epochs for various percentages of labelled data, showing convergence rates and differences in ranges of accuracy. (fifth repetition is used for every percentage.



TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNITTESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted
Invalid Input	: identified classes of invalid input must berejected.
Functions	: identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

6.3.1 ENTER INPUT

Test case ID	**Test case Name**	**Purpose**	**Test case**	**Output**
1	Retinal fundus image is given	Use it for Identification	Identifies the severity of disease	Kappa kaggle score is given
2	Another Retinal fundus image is given	Use it for Identification	Identifies the severity of disease	Kappa kaggle score is given

6.3.2 CLASSIFICATION

Test case ID	**Test case Name**	**Purpose**	**Input**	**Output**
1	Classification Test 1	To check if the classifier performs its task	Retinal fundus image is given	Kappa kaggle score is predicted
2	Classification Test 2	To check if the classifier performs its task	Retinal fundus image is given	Kappa kaggle score is predicted
3	Classification Test 3	To check if the classifier performs its task	Retinal fundus image is given	Kappa kaggle score is predicted

CONCLUSION AND FUTURE SCOPE

7. CONCLUSION

7.1 CONCLUSION

We developed a ConvNet model that outperforms the supervised baseline findings in this study. Furthermore, we employed data-efficient evaluation and implemented with very little data. Our findings, particularly in the low data regime, show that in the medical imaging sector, where data and annotation scarcity is a problem, it is possible to reduce the manual annotation labour necessary. Furthermore, we find that pretraining our algorithms on a large unlabeled corpus then fine-tuning them on a smaller downstream-specific dataset yields comparable results. We believe there is room for progress in this area, such as creating new proxy tasks, assessing other architectural solutions, and combining images/scans with other data modalities (e.g., text). We believe that utilising deep learning to diagnose Diabetic Retinography improves and mitigates the risk of vision loss for many patients, as well as making it cost-effective for regular check-ups.

7.2 FUTURE SCOPE

These results, however, do not appear to be very promising for real-world applications. We believe there is room for progress in this area, such as creating new proxy tasks, assessing other architectural solutions, and combining images/scans with other data modalities (e.g., text).

We believe that utilising deep learning to diagnose Diabetic Retinography improves and mitigates the risk of vision loss for many patients, as well as making it cost-effective for regular check-ups.

BIBILOGRAPHY

8. BIBILOGRAPHY

1. International Diabetes Federation. IDF Diabetes Atlas 7th edn (International Diabetes Federation, Brussels, Belgium, 2015).

2. https://youtu.be/SaJL4SLfrcY

3. C. Doersch, A. Gupta and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1422-1430, doi: 10.1109/ICCV.2015.167.

4. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles [1603.09246v3] Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles (arxiv.org)

5. Zhang R., Isola P., Efros A.A. (2016) Colorful Image Colorization. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science,vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_40

6. Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. CoRR, abs/1803.07728, 2018. URL http://arxiv.org/abs/1803.07728.

7. Jiawei Wang, Shuai Zhu, Jiao Xu, and Da Cao. The retrieval of the beautiful: Self supervised salient object detection for beauty product retrieval. In Proceedings of the 27th ACM International Conference on Multimedia, MM '19, page 2548–2552, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368896. doi: 10.1145/3343031.3356059.
URL: https://doi.org/10.1145/3343031. 3356059.

8. Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

9. Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. CoRR, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

10. https://www.kaggle.com/sovitrath/diabetic-retinopathy-224x224-2019-data XI 3D self-supervised for medical imaging. https://arxiv.org/abs/2006.03829