Technical Answers for Real World Problems (TARP)
CSE3999

TEAM MEMBERS:

Om Ashish Mishra	16BCE0789
Dhruv Goel	16BCE0686
Samyak Jain	16BCE0712

AIM:

This project aims to see the reduction in skin cancer in India and world at large. The research papers have found it to be difficult to do so with a very less accuracy. The project represents a good accuracy in finding the skin cancer occurrence and help more and more people.

OBJECTIVE:

The project is about making a Skin Cancer Detector. In India, many people are suffering through skin diseases. Many a times in village's people think it is just a mole and ignore it. Later when it is realised it is already too late to cure and the patients suffer a lot. Some even die. Thus keeping in thought the problem we are making an app which will help in detecting skin cancer through deep learning algorithms and using Generative Adversarial Networking for generating and checking more possible test cases. Thus, it will not only help to have a detection but also able to give a free chat with doctor for fixing an appointment for the skin cancer check-up.

Skin cancer constitutes a small but significant proportion of patients with cancer. Although the presence of eumelanin in dark skin is protective against the development of skin cancer, it is increasingly being diagnosed in the Indian population.

INTRODUCTION:

We are making a skin cancer classifier which classifies whether the user has cancer. The user can ask for online assistance to a doctor. More than half of people worry less about skin cancer than other cancers, while many do not realize it can be fatal, according to a poll. Some 53% are less worried about skin cancer than other cancers and 18% think it can be easily avoided, a survey for the British Skin Foundation found. A total of 38% do not realise that skin cancer can lead to death, while 56% do not know that malignant melanoma - the most deadly form of skin cancer - can spread to other parts of the body. There are some models that exist for this task, but in our opinion, they don't have enough accuracy. We plan on building a state of the art models using generative adversarial networks. We do not plan to completely eliminate the traditional methods for detection of skin cancer. We merely want to supplement them and make basic diagnosis more affordable to the common man.

Literature Survey

We went through many research papers before coming to a solution with the project. Here are they, listed below:-

Skin lesion classification from dermoscopic images using deep learning techniques

The recent emergence of deep learning methods for medical image analysis has enabled the development of intelligent medical imaging-based diagnosis systems that can assist the human expert in making better decisions about a patients health. The focus in on the problem of skin lesion classification, particularly early melanoma detection, and present a deep-learning based approach to solve the problem of classifying a dermoscopic image containing a skin lesion as malignant or benign.

Skin Lesion Classification Using Hybrid Deep Neural Networks

Skin cancer is one of the major types of cancers and its incidence has been increasing over the past decades. Skin lesions can arise from various dermatologic disorders and can be classified to various types according to their texture, structure, color and other morphological features. The accuracy of diagnosis of skin lesions, specifically the discrimination of benign and malignant lesions, is paramount to ensure appropriate patient treatment. Machine learning-based classification approaches are among popular automatic methods for skin lesion classification. While there are many existing methods, convolutional neural networks (CNN) have shown to be superior over other classical machine

learning methods for object detection and classification tasks. A fully automatic computerized method is present which employs well established pre-trained convolutional neural networks and ensembles learning to classify skin lesions.

Dermatologist-level classification of skin cancer with deep neural networks

Skin cancer is the most common human malignancy which is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Deep convolutional neural networks (CNNs) show potential for general and highly variable tasks across many fine-grained object categories. The demonstration shows the classification of skin lesions using a single CNN, trained end-to-end from images directly, using only pixels and disease labels as inputs. The sim is to train a CNN using a dataset of 129,450 clinical images—two orders of magnitude larger than previous datasets—consisting of 2,032 different diseases. The performance is tested against 21 board-certified dermatologists on biopsy-proven clinical images with two critical binary classification use cases: keratinocyte carcinomas versus benign seborrheic keratoses; and malignant melanomas versus benign nevi. The first case represents the identification of the most common cancers, the second represents the identification of the deadliest skin cancer. The CNN achieves performance on par with all tested experts across both tasks, demonstrating an artificial intelligence capable of classifying skin cancer with a level of competence comparable to dermatologists. Enabling deep neural networks in mobile devices, they can potentially extend the reach of dermatologists outside of the clinic. It is projected that 6.3 billion smartphone subscriptions will exist by the year 2021 and can therefore potentially provide low-cost universal access to vital diagnostic care.

The causes of skin cancer: a comprehensive review.

Skin cancer is the most common type of cancer in fair-skinned populations around the world. The incidence and mortality rates of skin cancers are dramatically increasing and thus pose a threat to public health. Understanding the pathogenesis of skin cancer remains a goal for healthcare systems. A clearer understanding of causative factors is an essential step in the prevention of skin cancer. This article comprehensively reviews the causative agents which play a role in the development of skin cancer. Ultraviolet radiation (UV) from sun exposure is the most important cause of skin cancer. Sunburns and excessive exposures cause cumulative damage which induces immunosuppression and skin

cancers. Ozone depletion, the level of UV light, elevation, latitude, altitude and weather conditions influence the emission of UV radiation reaching the earth's surface. Organ transplant recipients and AIDS patients have an increased incidence of skin cancers. Some treatment modalities, including radiation therapy, phototherapy and psoralen and long-wave ultraviolet radiation (PUVA) can also predispose to skin cancers. Individuals with familial genetic syndromes are susceptible to specific types of skin cancers. Ionizing radiation, environmental pollutants, chemical carcinogens and work-related exposures have been associated with skin cancers. Exposure to artificial UV radiation (tanning beds and lamps), aging, skin color, diet and smoking are attributable risks. Skin cancers have been found in dermatoses and various types of keratoses, chronically injured or nonhealing wounds, and scars. This article provides a comprehensive and thorough overview of skin cancer, with an emphasis on understanding its epidemiology, incidence, etiology and related risk factors.

Surgical management of skin cancers: Experience from a regional cancer centre in North India

To review the disease profile and treatment outcome of patients with primary skin malignancies treated at a regional cancer centre. **Settings and Design:** Surgical oncology unit of a tertiary care regional cancer centre. Evaluation of treatment outcome of patients with skin cancer from Surgical Oncology database was done. **Materials and METHODS:** Retrospective analysis of records of 77 patients with skin cancers treated between 1995 and 2002 was conducted. Profile of patients with skin cancer, surgical details including the management of primary tumour, regional lymph nodes and reconstructive procedures performed and survivals were analysed. **Statistical analysis:** All computations were done using the Statistical Package for Social Sciences (SPSS-9). Descriptive statistics were calculated in a standard fashion and survival analysis was performed using Kaplan-Meier method. **Results:** Skin cancers constituted 2.4% (77/3154) of patients with cancer treated in the surgical oncology department. Squamous cell carcinoma (SCC) was the most common histological type (55.8%) followed by melanoma (26.1%) and basal cell carcinoma (BCC, 18.1%). Forty one percent of patients had undergone some form of intervention elsewhere before being referred. Reconstruction was required in 55.8% patients with large postresection defects. Regional lymph nodal dissection was required in 32.4% of total patients. Five-year median disease-free survival for the entire study population was

75%. **Conclusions:** Skin cancers constitute a small but significant proportion of patients with cancer. Unlike in the Western countries, SCC is the commonest histologic variety. Primary level inadequate intervention is very common. Optimal results can be obtained with radical surgery and optimal surgical margins along with a reconstructive procedure when needed.

Nonmelanoma skin cancer in India: Current Scenario

Incidence of skin cancers has been increasing since the last few decades worldwide. Nonmelanoma skin cancer (NMSC) is the commonest variety of cutaneous malignancy. Conventional wisdom has it that the incidence of all varieties of skin cancers is lower among Indians due to the protective effects of melanin. Though national surveys and cross-country data in India are unavailable, there are indirect indications from several smaller reports that NMSCs may be on the rise in India. Reports of quite a few atypical cases lead us to hypothesize that factors other than ultraviolet radiation may be important in the occurrences of these cancers, particularly in the skin types prevalent in India. The descriptive epidemiology and clinical characteristics of squamous and basal cell carcinoma in India, including their variants, are discussed here along with hypotheses on their etiopathogenesis. Novel management techniques currently available in India are also highlighted.

A Deep Learning Approach to Universal Skin Disease Classification

Haofu Liao

Skin diseases are very common in people’s daily life. Each year, millions of people in American are affected by all kinds of skin disorders. Diagnosis of skin diseases sometimes requires a high-level of expertise due to the variety of their visual aspects. As human judgment are often subjective and hardly reproducible, to achieve a more objective and reliable diagnosis, a computer aided diagnostic system should be considered. In this paper, we investigate the feasibility of constructing a universal skin disease diagnosis system using deep convolutional neural network (CNN). We train the CNN architecture using the 23,000 skin disease images from the Dermnet dataset and test its performance with both the Dermnet and OLE, another skin disease dataset, images. Our system can achieve as high as 73.1% Top-1 accuracy and 91.0% Top-5 accuracy when testing on the

Dermnet dataset. For the test on the OLE dataset, Top-1 and Top-5 accuracies are 31.1% and 69.5%. We show that these accuracies can be further improved if more training images are used.

Review:

Uses CNNs to get 73% accuracy. However, it observes that there accuracy can be improved by using more training images, for example, using data augumentation. There is also lack of variance in the dataset. The accuracy can also be improved using preprocessing like segmentation.

Novel Approaches for Diagnosing Melanoma Skin Lesions Through Supervised and Deep Learning Algorithms

Dermoscopy is a technique used to capture the images of skin, and these images are useful to analyze the different types of skin diseases. Malignant melanoma is a kind of skin cancer whose severity even leads to death. Earlier detection of melanoma prevents death and the clinicians can treat the patients to increase the chances of survival. Only few machine learning algorithms are developed to detect the melanoma using its features. This paper proposes a Computer Aided Diagnosis (CAD) system which equips efficient algorithms to classify and predict the melanoma. Enhancement of the images are done using *Contrast Limited Adaptive Histogram Equalization technique (CLAHE)* and *median filter*. A new segmentation algorithm called *Normalized Otsu's Segmentation (NOS)* is implemented to segment the affected skin lesion from the normal skin, which overcomes the problem of variable illumination. Fifteen features are derived and extracted from the segmented images are fed into the proposed classification techniques like *Deep Learning based Neural Networks* and *Hybrid Adaboost-Support Vector Machine (SVM)* algorithms. The proposed system is tested and validated with nearly 992 images (malignant & benign lesions) and it provides a high classification accuracy of 93 %. The proposed CAD system can assist the dermatologists to confirm the decision of the diagnosis and to avoid excisional biopsies.

Review:

They use various preprocessing techniques like CLAHE and median filter. They also use otsu thresholding. They use a deep neural network and a hybrid of a support vector machine and an adaboost classifier. It got 93% accuracy.

Deep learning ensembles for melanoma recognition in dermoscopy images

Melanoma is the deadliest form of skin cancer. While curable with early detection, only highly trained specialists are capable of accurately recognizing the disease. As expertise is in limited supply, automated systems capable of identifying disease could save lives, reduce unnecessary biopsies, and reduce costs. Toward this goal, we propose a system that combines recent developments in deep learning with established machine learning approaches, creating ensembles of methods that are capable of segmenting skin lesions, as well as analyzing the detected area and surrounding tissue for melanoma detection. The system is evaluated using the largest publicly available benchmark dataset of dermoscopic images, containing 900 training and 379 testing images. New state-of-the-art performance levels are demonstrated, leading to an improvement in the area under receiver operating characteristic curve of 7.5% (0.843 versus 0.783), in average precision of 4% (0.649 versus 0.624), and in specificity measured at the clinically relevant 95% sensitivity operating point 2.9 times higher than the previous state of the art (36.8% specificity compared to 12.5%). Compared to the average of eight expert dermatologists on a subset of 100 test images, the proposed system produces a higher accuracy (76% versus 70.5%), and specificity (62% versus 59%) evaluated at an equivalent sensitivity (82%).

Review:

They use an ensemble of models and get accuracy of 76%. They conclude that this is higher accuracy with multiple experts trying to identify the disease. They use segmentation for preprocessing of images.

Machine-learning classification of non-melanoma skin cancers from image features obtained by optical coherence tomography

Background/purpose: A number of publications have suggested that optical coherence tomography (OCT) has the potential for non-invasive diagnosis of skin

cancer. Currently, individual diagnostic features do not appear sufficiently discriminatory. The combined use of several features may however be useful.

Methods: OCT is based on infrared light, photonics and fibre optics. The system used has an axial resolution of 10 μm, lateral 20 μm. We investigated the combined use of several OCT features from basal cell carcinomas (BCC) and actinic keratosis (AK). We studied BCC (41) and AK (37) lesions in 34 consecutive patients. The diagnostic accuracy of the combined features was assessed using a machine-learning tool.

Results: OCT images of normal skin typically exhibit a layered structure, not present in the lesions imaged. BCCs showed dark globules corresponding to basaloid islands and AKs showed white dots and streaks corresponding to hyperkeratosis. Differences in OCT morphology were not sufficient to differentiate BCC from AK by the naked eye. Machine-learning analysis suggests that when a multiplicity of features is used, correct classification accuracies of 73% (AK) and 81% (BCC) are achieved.

Conclusion: The data extracted from individual OCT scans included both quantitative and qualitative measures, and at the current level of resolution, these single factors appear insufficient for diagnosis. Our approach suggests that it may be possible to extract diagnostic data from the overall architecture of the OCT images with a reasonable diagnostic accuracy when used in combination.

Risk of Subsequent Basal Cell Carcinoma and Squamous Cell Carcinoma of the Skin Among Patients With Prior Skin Cancer

Objective. —The primary aims of this study were to assess risk of subsequent basal and squamous cell skin cancer among patients with a prior history of these tumors and to examine these risks in relation to patient characteristics and life-style factors.

Design. —Follow-up of participants in a randomized trial of betacarotene as a possible skin cancer preventive agent.

Setting. —Clinical centers in Los Angeles, Calif, San Francisco, Calif, Minneapolis, Minn, and Hanover, NH.

Participants. —Patients (n=1805) who were diagnosed as having a basal or squamous cell skin cancer between January 1980 and February 1986 and were free of skin cancer at study entry.

Main Outcome Measure. —Time from study entry to first new occurrence of basal and squamous cell skin cancer.

Results. —The estimated risk of developing one or more new skin cancers was 35% at 3 years and 50% at 5 years. New skin cancers tended to be of the same cell type as the previous skin cancers. For both basal and squamous cell skin cancer, risk was higher among patients who were male, were over the age of 60 years, had more prior skin cancers, had severe actinic skin damage, or who burned easily with sun exposure. Compared with those who had never smoked, the rate of subsequent squamous cell skin cancer was higher among current smokers (rate ratio, 2.01; 95% confidence interval,1.21 to 3.34) and former smokers (rate ratio,1.62; 95% confidence interval, 1.07 to 2.47) and increased with both duration and amount smoked. There was no clear relationship between smoking and basal cell skin cancer; the rate appeared lower among heavy smokers but was unrelated to duration of smoking.

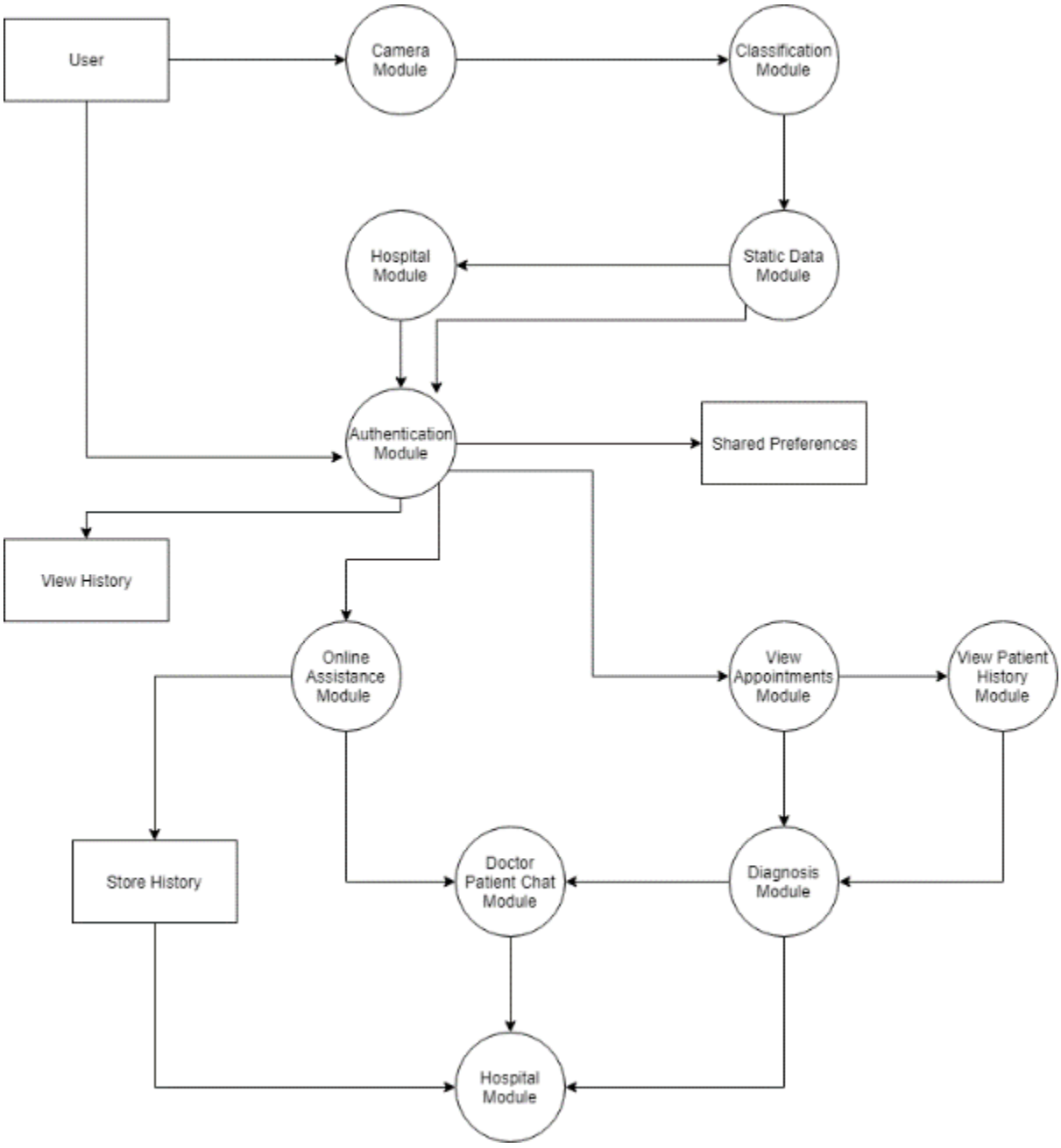
Conclusions. —Persons with a prior nonmelanoma skin cancer had a substantial 5-year risk of developing another tumor of the same histologic type. Number of previous skin cancers, solar damage, and skin sensitivity to sun exposure were particularly related to this risk. The increased risk of squamous cell skin cancer associated with cigarette smoking merits further study.(*JAMA.* 1992;267:3305-3310)

An Overview of Melanoma Detection in Dermoscopy Images Using Image Processing and Machine Learning

The incidence of malignant melanoma continues to increase worldwide. This cancer can strike at any age; it is one of the leading causes of loss of life in young persons. Since this cancer is visible on the skin, it is potentially detectable at a very early stage when it is curable. New developments have converged to make fully automatic early melanoma detection a real possibility. First, the advent of dermoscopy has enabled a dramatic boost in clinical diagnostic ability to the point that melanoma can be detected in the clinic at the very earliest stages. The global adoption of this technology has allowed accumulation of large collections of dermoscopy images of melanomas and benign lesions validated by histopathology. The development of advanced technologies in the areas of image processing and machine learning have given us the ability to allow distinction of malignant melanoma from the many benign mimics that require no biopsy. These new technologies should allow not only earlier detection of melanoma, but also reduction of the large number of needless and costly biopsy procedures. Although some of the new systems reported for these technologies have shown promise in

preliminary trials, widespread implementation must await further technical progress in accuracy and reproducibility. In this paper, we provide an overview of computerized detection of melanoma in dermoscopy images. First, we discuss the various aspects of lesion segmentation. Then, we provide a brief overview of clinical feature segmentation. Finally, we discuss the classification stage where machine learning algorithms are applied to the attributes generated from the segmented features to predict the existence of melanoma.

Architecture Layout

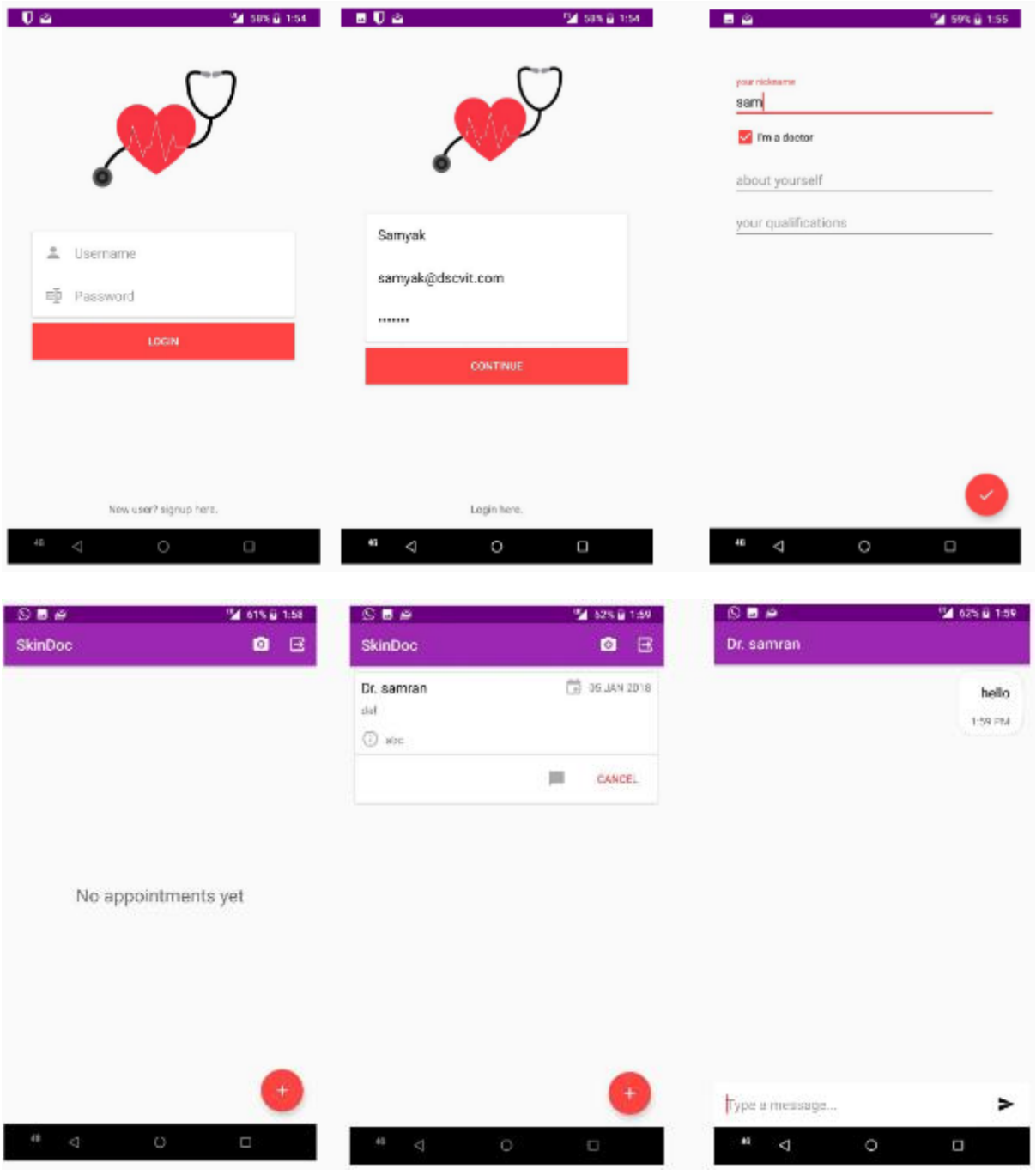


Dataset Images

A	B	C	D	E	F	G	H
id	name	meta.clinical.age_approx	meta.clinical.anatom_site_genera	meta.clinical.benign_malignant	meta.clinical.clin_size_long_diam_	meta.clinical.diagnosis	meta.clinical.diagnosis_confirm_type
5436e3abbae478396759f0cf	ISIC_0000000	55	anterior torso	benign		nevus	
5436e3acbae478396759f0d1	ISIC_0000001	30	anterior torso	benign		nevus	
5436e3acbae478396759f0d3	ISIC_0000002	60	upper extremity	malignant		melanoma	histopathology
5436e3acbae478396759f0d5	ISIC_0000003	30	upper extremity	benign		nevus	
5436e3acbae478396759f0d7	ISIC_0000004	80	posterior torso	malignant		melanoma	histopathology
5436e3acbae478396759f0d9	ISIC_0000005	40	anterior torso	benign		nevus	
5436e3acbae478396759f0db	ISIC_0000006	25	posterior torso	benign		nevus	
5436e3acbae478396759f0dd	ISIC_0000007	25	posterior torso	benign		nevus	
5436e3acbae478396759f0df	ISIC_0000008	30	anterior torso	benign		nevus	
5436e3acbae478396759f0e1	ISIC_0000009	30	anterior torso	benign		nevus	
5436e3acbae478396759f0e3	ISIC_0000010	35	posterior torso	benign		nevus	
5436e3acbae478396759f0e5	ISIC_0000011	35	lower extremity	benign		nevus	
5436e3acbae478396759f0e7	ISIC_0000012	30	posterior torso	benign		nevus	
5436e3adbae478396759f0e9	ISIC_0000013	30	posterior torso	malignant		melanoma	histopathology
5436e3adbae478396759f0eb	ISIC_0000014	35	posterior torso	benign		nevus	
5436e3adbae478396759f0ed	ISIC_0000015	35	posterior torso	benign		nevus	
5436e3adbae478396759f0ef	ISIC_0000016	55	anterior torso	benign		nevus	
5436e3adbae478396759f0f1	ISIC_0000017	50	posterior torso	benign		nevus	
5436e3adbae478396759f0f3	ISIC_0000018	30	posterior torso	benign		nevus	
5436e3adbae478396759f0f5	ISIC_0000019	30	posterior torso	benign		nevus	
5436e3adbae478396759f0f7	ISIC_0000020	25	anterior torso	benign		nevus	
5436e3adbae478396759f0f9	ISIC_0000021	55	posterior torso	benign		nevus	
5436e3adbae478396759f0fb	ISIC_0000022	55	lower extremity	malignant		melanoma	histopathology

Individual Contribution:

Dhruv(The Frontend Development):



Om (Backend Connector):

```
import random
import json
import os
import traceback
import tornado.httpserver
import tornado.ioloop
import tornado.web
from motor import motor_tornado
```

```
from tornado.gen import coroutine
from tornado.options import define, options
from models import users, patient
from passlib.hash import pbkdf2_sha256
import tornado.escape
import jwt
from functools import wraps
from oauth2client import client as auth_client
from oauth2client import crypt
from io import BytesIO
from PIL import Image
import base64
import requests

define("port", default=8080, help="runs on the given port", type=int)


class MyAppException(tornado.web.HTTPError):
    pass


def protected(f):
    @wraps(f)
    def wrapper(*args, **kwargs):
        token = tornado.escape.json_decode(args[0].request.body)
        decoded_token = jwt.decode(token['token'], "abc")
        if decoded_token['valid']:
            return f(*args, **kwargs)
        else:
            raise MyAppException(reason="Invalid Token",
status_code=401)
    return wrapper


def Authenticated(f):
    @wraps(f)
    def wrapper(*args, **kwargs):
        # try:
        #     token = args[0].get_argument('token')
        # except tornado.web.MissingArgumentError:
        token = tornado.escape.json_decode(args[0].request.body)
        decoded_token = jwt.decode(token['token'], "abc")
        if decoded_token['user'] in args[0].settings['logged_in']:
            return f(*args, **kwargs)
        else:
            raise MyAppException(reason='User is not logged in.',
status_code=301)
    return wrapper


class BaseHandler(tornado.web.RequestHandler):

    def db(self):
        clientz = self.settings['db_client']
        db = clientz.tornado
        return db

    def write_error(self, status_code, **kwargs):
```

```
        self.set_header('Content-Type', 'application/json')
        if self.settings.get("serve_traceback") and "exc_info" in
kwargs:
            # in debug mode, try to send a traceback
            lines = []
            for line in
traceback.format_exception(*kwargs["exc_info"]):
                lines.append(line)
            self.write(json.dumps({
                        'status_code': status_code,
                        'message': self._reason,
                        'traceback': lines,
                }))
        else:
            self.write(json.dumps({
                    'status_code': status_code,
                    'message': self._reason,
                }))


class AuthHandler(BaseHandler):
    @coroutine
    def post(self):
        self.set_header('Content-Type', 'application/json')
        login_data = tornado.escape.json_decode(self.request.body)
        if not login_data:
            raise MyAppException(reason="Invalid Credentials",
status_code=400)
        db_client = self.db()
        hash = yield db_client.auth.find_one({"user":
login_data['user']})
        if not hash:
            raise MyAppException(reason="Invalid Credentials",
status_code=400)
        if 'google' in hash:
            self.write(json.dumps({
                'message': 'Invalid Credentials',
                'status_code': 400,
            }))
        if pbkdf2_sha256.verify(login_data['pass'], hash["pass"]):
            flag = {'status_code': 200, 'message': 'valid', 'portal':
hash['portal']}
        else:
            raise MyAppException(reason="Invalid Credentials",
status_code=400)
        if str(flag['portal']) == "1":
            response = yield db_client.patient.find_one({"user":
login_data['user']})
        else:
            response = yield db_client.doctor.find_one({"user":
login_data['user']})
        response.pop('_id', None)
        response['valid'] = True
        token = jwt.encode(response, "abc")
        self.write(json.dumps({
            'message': 'Logged in Successfully.',
            'status_code': 200,
            'token': token.decode('utf-8')
```

```
        }))


class SignUpHandler(BaseHandler):
    @coroutine
    def post(self):
        self.set_header('Content-Type', 'application/json')
        signup_data = tornado.escape.json_decode(self.request.body)
        if signup_data is None:
            raise MyAppException(reason='Invalid Data Given.',
status_code=400)
        db_client = self.db()
        database_auth = db_client["auth"]
        database_details = None
        if str(signup_data['portal']) == "1":
            database_details = db_client["patient"]
            signup_data['ap_details'] = list()

        elif str(signup_data['portal']) == "0":
            database_details = db_client["doctor"]
            signup_data['plist'] = list()
            signup_data['availability'] = True
        signup_data['ap_inactive'] = list()

        if database_details is None:
            raise MyAppException(reason='Portal Not Set.',
status_code=400)
        find_email = yield database_auth.find_one({"user":
signup_data['user']})
        find_user = yield database_details.find_one({'email':
signup_data['email']})
        if find_email:
            raise MyAppException(reason='User Exists', status_code=400)
        if find_user:
            raise MyAppException(reason='Email Exists',
status_code=400)
        if 'google' in signup_data:
            hash_pass = 'google'
        else:
            hash_pass = pbkdf2_sha256.hash(signup_data['pass'])
        signup_data.pop('pass', None)
        signup_data['valid'] = True
        token = jwt.encode(signup_data, "abc")
        signup_data.pop('valid', None)
        if 'google' in signup_data:
            database_auth.insert_one({"user": signup_data['user'],
"pass": hash_pass, "portal": signup_data['portal'],
                                      "email": signup_data['email'],
"google": True})
        else:
            database_auth.insert_one({"user": signup_data['user'],
"pass": hash_pass, "portal": signup_data['portal'],
                                      "email": signup_data['email']})
        database_details.insert_one(signup_data)
        self.write(json.dumps({
            'status_code': 200,
            'message': 'Sign-up successful.',
            'token': token.decode('utf-8')
```

```
        }))


class my404handler(BaseHandler):
    def get(self):
        self.set_header('Content-Type', 'application/json')
        self.write(json.dumps({
                'status_code': 404,
                'message': 'illegal call.'
        }))


class OauthLogin(BaseHandler):
    @coroutine
    def post(self):
        self.set_header('Content-Type','application/json')
        token = tornado.escape.json_decode(self.request.body)
        try:
            idinfo = auth_client.verify_id_token(token['token'],
self.settings['client_id'])
        except crypt.AppIdentityError:
            idinfo = None
        if idinfo is None:
            raise MyAppException(reason="Invalid Token",
status_code=400)
        elif idinfo['iss'] not in ['accounts.google.com',
'https://accounts.google.com']:
            raise MyAppException(reason="wrong Issuer",
status_code=400)
        db = self.db()
        find_email = yield db.auth.find_one({"email": idinfo['email']})
        if find_email:
            if str(find_email['portal']) == "1":
                det = yield db.patient.find_one({"email":
idinfo['email']})
            else:
                det = yield db.doctor.find_one({"email":
idinfo['email']})
            det.pop('_id', None)
            det['valid'] = True
            tokenz = jwt.encode(det, "abc")
            self.write(json.dumps({
                "message": "Logged In",
                "token": tokenz.decode('utf-8'),
                "status_code": 200
            }))
        else:
            details = dict(email=idinfo['email'], name=idinfo['name'],
google=True)
            tokenz = jwt.encode(details, "abc")
            self.write(json.dumps({
                "message": "Signup required",
                "status_code": 301,
                "token": tokenz.decode('utf-8')
            }))


class LogoutHandler(BaseHandler):
```

```
    @coroutine
    @protected
    def post(self):
        self.set_header('Content-type', 'application/json')
        token = tornado.escape.json_decode(self.request.body)
        decoded_token = jwt.decode(token['token'], "abc")
        self.write(json.dumps({
            'message': 'Logged Out.',
            'status_code': 200
        }))


class NewTokenGenerator(BaseHandler):
    @coroutine
    @protected
    def post(self):
        self.set_header('Content-type', 'application/json')
        token = tornado.escape.json_decode(self.request.body)
        decoded_token = jwt.decode(token['token'], "abc")
        db = self.db()
        if str(decoded_token['portal']) == "1":
            response = yield db.patient.find_one({"user":
decoded_token['user']})
        else:
            response = yield db.doctor.find_one({"user":
decoded_token['user']})
        response.pop('_id', None)
        response['valid'] = True
        refreshed_token = jwt.encode(response, "abc")
        self.write(json.dumps({
            'status_code': 200,
            'message': 'Token Generated.',
            'token': refreshed_token.decode('utf-8')
        }))


class AppointmentHandler(BaseHandler):
    @coroutine
    @protected
    def post(self):
        self.set_header('Content-type', 'application/json')
        token = tornado.escape.json_decode(self.request.body)
        data = jwt.decode(token['token'], "abc")
        data['description'] = token['description']
        flag = yield patient.make_appointment(self.db(), data)
        if flag['status_code'] == 200:
            self.write(json.dumps(flag))
        else:
            raise MyAppException(reason=flag['message'],
status_code=flag['status_code'])


class ResolveAppointment(BaseHandler):
    @coroutine
    @protected
    def post(self):
        """
        removes appointment for doctor
```

```
        makes the status of the appointment done for patient
        """
        self.set_header('Content-type','application/json')
        token = tornado.escape.json_decode(self.request.body)
        db = self.db()
        user = token['user']
        doctor = token['doctor']
        doc = yield db.doctor.find_one({"user": doctor})
        pat = yield db.patient.find_one({"user": user})

        for record in doc['plist']:
            if pat['user'] == record['user']:
                doc['plist'].remove(record)
                doc['ap_inactive'].append(record)
        plist = doc['plist']
        ap_inactive1 = doc['ap_inactive']

        for record in pat['ap_details']:
            if doc['user'] == record['user']:
                pat['ap_details'].remove(record)
                pat['ap_inactive'].append(record)
        ap_details = pat['ap_details']
        ap_inactive2 = pat['ap_inactive']

        db.patient.update({'_id': pat['_id']}, {'$set': {'ap_details':
ap_details, 'ap_inactive': ap_inactive2}}, upsert=False)
        db.doctor.update({'user': doc['user']}, {'$set': {'plist':
plist, 'ap_inactive': ap_inactive1}}, upsert=False)
        self.write(json.dumps({
            'status_code': 200,
            'message': 'Updated Successfully.'
        }))


class MLHandler(tornado.web.RequestHandler):
    def set_default_headers(self):
        self.set_header("Access-Control-Allow-Origin", "*")
        self.set_header("Access-Control-Allow-Headers", "x-requested-
with")
        self.set_header('Access-Control-Allow-Methods', 'POST, GET,
OPTIONS')
        self.set_header('Content-type','application/json')

    @coroutine
    def post(self):
        self.write(json.dumps({"confidence": random.randint(1,48),
"result": "benign"}))
#         file_body = self.request.files['filefieldname'][0]['body']
#         img = Image.open(BytesIO(file_body))
#         img.save("current.jpg")
#         f = open("current.jpg", "rb")
#         images = [{'content': base64.b64encode(f.read()).decode('UTF-
8')}]
#         x = requests.post("https://skindoc-
10ef5.appspot.com/melanoma/predict", json=images)
#         f.close()
#         self.write(json.dumps(x.json()[0]))
```

```
class AdminFeature(tornado.web.RequestHandler):
    def get(self):
        self.write(json.dumps({'message': self.settings['logged_in']}))


if __name__ == "__main__":
    options.parse_command_line()
    client =
motor_tornado.MotorClient("mongodb://samyak:samyak1@ds247944.mlab.com:4
7944/tornado")
    app = tornado.web.Application(
        handlers=[
            (r"/", AuthHandler),
            (r"/login", AuthHandler),
            (r"/Signup", SignUpHandler),
            (r"/signup", SignUpHandler),
            (r"/logout", LogoutHandler),
            (r"/appoint", AppointmentHandler),
            (r"/admin", AdminFeature),
            (r"/new", NewTokenGenerator),
            (r"/oauth",OauthLogin),
            (r"/resolve", ResolveAppointment),
            (r"/mlpredict", MLHandler)
        ],
        default_handler_class = my404handler,
        debug = True,
        cookie_secret = "abc",
        login_url = "/login",
        db_client = client,
        ap_details = dict(),
        client_id = "def",
        template_path=os.path.join(os.path.dirname(__file__),
"template"),
        static_path=os.path.join(os.path.dirname(__file__), "static"),
    )
    http_server = tornado.httpserver.HTTPServer(app)
    http_server.listen(os.environ.get("PORT", options.port))
    tornado.ioloop.IOLoop.instance().start()


from tornado.gen import coroutine
from random import choice
from datetime import datetime
class users:
    def __init__(self, email, user, name):
        self.email = email
        self.user = user
        self.name = name

class patient(users):
    @staticmethod
    @coroutine
    def make_appointment(db, user):
        resp = db.doctor.find({"$where": "this.plist.length<3"})
        listOfDoc = []
        while (yield resp.fetch_next):
            ele = resp.next_object()
```

```
            if 'qualifications' not in ele:
                ele['qualifications'] = None
            if 'description' not in ele:
                ele['description'] = None
            if 'name' not in ele:
                ele['name'] = ele['user']
            listOfDoc.append(
                dict(email=ele['email'], user=ele['user'],
plist=ele['plist'], availability=ele['availability'],
                     qualifications=ele['qualifications'],
description=ele['description'], name=ele['name']))
        print(user)
        if len(user['ap_details']) >= 3:
            print(user)
            return {'message': 'Maximum Appointments reached.',
'status_code': 400}
        if not listOfDoc:
            return {'message': 'Doctors Not Available.',
                    'status_code': 400}
        dbdoc = choice(listOfDoc)
        plist = [i['user'] for i in dbdoc['plist']]
        print(plist)
        while user['user'] in plist:
            listOfDoc.remove(dbdoc)
            if not listOfDoc:
                return {'message': 'Doctors Not Available.',
                        'status_code': 400}
            dbdoc = choice(listOfDoc)
            if not dbdoc['availability']:
                listOfDoc.remove(dbdoc)
                if not listOfDoc:
                    return {'message': 'Doctors Not Available.',
                            'status_code': 400}
                dbdoc = choice(listOfDoc)
            plist = [i['user'] for i in dbdoc['plist']]
        print(dbdoc)
        if 'name' not in user:
            user['name'] = user['user']

        user_details = dict(user=user['user'], name=user['name'],
description=user['description'],
                            datetime=datetime.now().strftime("%d-%m-%Y
%H:%M"))
        doctor_details = dict(user=dbdoc['user'], name=dbdoc['name'],
qualifications=dbdoc['qualifications'],
                              datetime=datetime.now().strftime("%d-%m-
%Y %H:%M"),description=dbdoc['description'])
        dbdoc['plist'].append(user_details)
        plist = dbdoc['plist']
        ap_list = user['ap_details']
        ap_list.append(doctor_details)
        db.patient.update({'user': user['user']}, {'$set':
{'ap_details': ap_list}}, upsert=False)
        db.doctor.update({'user': dbdoc['user']}, {'$set': {'plist':
plist}}, upsert=False)
        return {'status_code': 200, 'message': 'Updated Successfully.'}
```

Samyak (Machine Learning Developer):

```
import pytest
import base64
import os

from src.common.Image import Image

test_image_filename = os.path.dirname(__file__) +
'/resources/test_image.jpg'


def test_create_image_from_content():
    with open(test_image_filename, 'rb') as image_file:
        test_image = image_file.read()

    image_b64 = base64.b64encode(test_image).decode('UTF-8')

    image = Image(content=image_b64)

    assert image.content == base64.b64decode(image_b64)
    assert image.content_b64 == image_b64


def test_create_image_from_source():
    image = Image(source='some link')

    assert image.content == ""


def test_create_image_from_content_source():
    with open(test_image_filename, 'rb') as image_file:
        test_image = image_file.read()

    image_b64 = base64.b64encode(test_image).decode('UTF-8')

    image = Image(content=image_b64,
                  source="some link")

    assert image.content == base64.b64decode(image_b64)
    assert image.content_b64 == image_b64


def test_from_dict():
    with open(test_image_filename, 'rb') as image_file:
        test_image = image_file.read()

    image_b64 = base64.b64encode(test_image).decode('UTF-8')

    test_dict = {
        'content': image_b64,
        'source': "some link"
    }

    image = Image.from_dict(test_dict)

    assert image.content == base64.b64decode(image_b64)
    assert image.content_b64 == image_b64
```

```
def test_invalid_content():
    with pytest.raises(ValueError) as exception:
        Image(content='')

    assert 'image content is empty' == str(exception.value)
```

References:

1. Lopez, Adria Romero, et al. "Skin lesion classification from dermoscopic images using deep learning techniques." *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)*. IEEE, 2017.
2. Mahbod, Amirreza, Rupert Ecker, and Isabella Ellinger. "Skin lesion classification using hybrid deep neural networks." *arXiv preprint arXiv:1702.08434* (2017).
3. Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115.
4. Saladi, Rao N., and Andrea N. Persaud. "The causes of skin cancer: a comprehensive review." *Drugs of Today* 41.1 (2005): 37-54.
5. Deo, S. V., et al. "Surgical management of skin cancers: Experience from a regional cancer centre in North India." *Indian journal of cancer* 42.3 (2005): 145.
6. Panda, Saumya. "Nonmelanoma skin cancer in India: Current scenario." *Indian journal of dermatology* 55.4 (2010): 373.
7. Liao, Haofu. "A deep learning approach to universal skin disease classification." *University of Rochester Department of Computer Science, CSC* (2016).
8. Premaladha, J., and K. S. Ravichandran. "Novel approaches for diagnosing melanoma skin lesions through supervised and deep learning algorithms." *Journal of medical systems* 40.4 (2016): 96.
9. Codella, Noel CF, et al. "Deep learning ensembles for melanoma recognition in dermoscopy images." *IBM Journal of Research and Development* 61.4/5 (2017): 5-1.
10. Jørgensen, Thomas Martini, et al. "Machine-learning classification of non-melanoma skin cancers from image features obtained by optical coherence tomography." *Skin Research and Technology* 14.3 (2008): 364-369.
11. Karagas, Margaret R., et al. "Risk of subsequent basal cell carcinoma and squamous cell carcinoma of the skin among patients with prior skin cancer." *Jama* 267.24 (1992): 3305-3310.
12. Mishra, Nabin K., and M. Emre Celebi. "An overview of melanoma detection in dermoscopy images using image processing and machine learning." *arXiv preprint arXiv:1601.07843* (2016).