Identify Signs Of Diabetic Retinopathy In Eye Images

Using : Convolutional Neural Networks

Developed by: Bonthula Bhanu Sahith, Venkat Anand Sai Duggirala, Manchikanti Vikas, Jayanta Mukherjee, Nerella Anirudh

Smart Bridge-Remote Summer Internship Program

1.Introduction

The Artificial Intelligence is a trending and booming industry in modern age known by it's name Industry 4.0. As AI is being used in evey field for automation, prediction, assistance or aiding etc., the two sub fields of AI i.e., Machine learning and Deep learning are also becoming increasingly popular and important.In DL the field of Neural Networks is increasingly more popular and important.

In this project we are going to deal with the classification of the different stages of the diabetic retinopahty. To detect this we are going to use classification algorithms of DL. Even though there are classification algorithms in ML like KNN, SVM, Random Forests, Decision Trees and Logistic Regression, we use DL algorithm of CNN i.e., Convolution Neural Network. This is because the data that we are going give as input is an image of any format like jpeg, jpg, png etc., and this is actually a Gaussian Filtered image.

A gaussian filter is a linear filter. It's usually used to blur the image or to reduce noise. The gaussian filter alone will blur edges and reduce contrast. It's more effective in smoothing the images.

1.1 Overview

In this project as stated above we are going to deal with the classification on gaussian images using CNN and predict whether the person is having diabetic retina problem to a certain degree or not.

We chose CNN algorithm because it's a neural network which is capable of object recognition and deals with computer vision. CNN's can derive minor details

from an image known as features and basing in that can classify other images into separate categories. This is a supervised learning as the images are already labeled in dataset. We chose the Gaussian-filtered images dataset for the training part from the kaggle data repository which is around 400MB and then we applied the CNN algorithm using Tensorflow and Keras libraries in python to classify the images. Meanwhile an application using flask API in python and HTML, CSS, Java-Script is developed where an image which is already gaussian filtered is given as an input. This input image is then predicted for it's degree of damage to the retina.

1.2 Purpose

The aim of the project is to make use of the packages like numpy, cv2, skimage, keras, tensorflow to use CNN and build a model that would classify the given image in either of the five categories i.e., no_DR, DR, mild, moderate, severe.

In this project we also made use of the web application and finally we have created an application a person can check whether he /she is having a diabetic retinal damage or not just by inputting the image scanned of his eye.

2.LITERATURE SURVEY

CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular.

These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition, etc.

2.1 Existing Problem

Diabetic Retinopathy is the leading cause of blindness in the working-age population of the developed world. It is estimated to affect over 93 million people. The US Center for Disease Control and Prevention estimates that 29.1 million people in the US have diabetes and the World Health Organization estimates that 347 million people have the disease worldwide. Diabetic Retinopathy (DR) is an eye disease associated with long-standing diabetes. Around 40% to 45% of Americans with diabetes have some stage of the disease. Progression to vision impairment can be slowed or averted if DR is detected in time, however this can be

difficult as the disease often shows few symptoms until it is too late to provide effective treatment.

Currently, detecting DR is a time-consuming and manual process that requires a trained clinician to examine and evaluate digital color fundus photographs of the retina. By the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment.

2.2 Proposed Solution

This project is capable of classifying images based on disease pathology from four severity levels. A convolutional neural network (CNN) convolves an input image with a defined weight matrix to extract specific image features without losing spatial arrangement information. There will be a web application through which user can give their input image then they can check the predicted output and this application is integrated with trained CNN model.

3.THEORETICAL ANALYSIS

There are three basic components to define a basic convolutional network.

1. The convolutional layer
2. The Pooling layer[optional]
3. The output layer

The convolutional layer:
->Image of size 6*6. We define a weight matrix which extracts certain features from the images. Weight as a 3*3 matrix. This weight shall now run across the image such that all the pixels are covered at least once, to give a convolved output.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

->The weight matrix behaves like a filter in an image extracting particular information from the original image matrix. A weight combination might be extracting edges.
->If stride is equal to 1,then weight matrix moves 1 step a head.It acts as a hyper-parameter.If stide is more then there will be a loss of information(Pixels present in the corner of the image are used only a few number of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss).
->To overcome this issue,Padding is is used over there. initial shape of the image is retained after we padded the image with a zero. This is known as **same padding**

0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0
0	0	0	0	0	0	0	0

->In most cases instead of a single filter(weight matrix), we have multiple filters of the same dimensions applied together.

3.2 Software Designing

- Jupyter Notebook Environment
- Spyder Ide

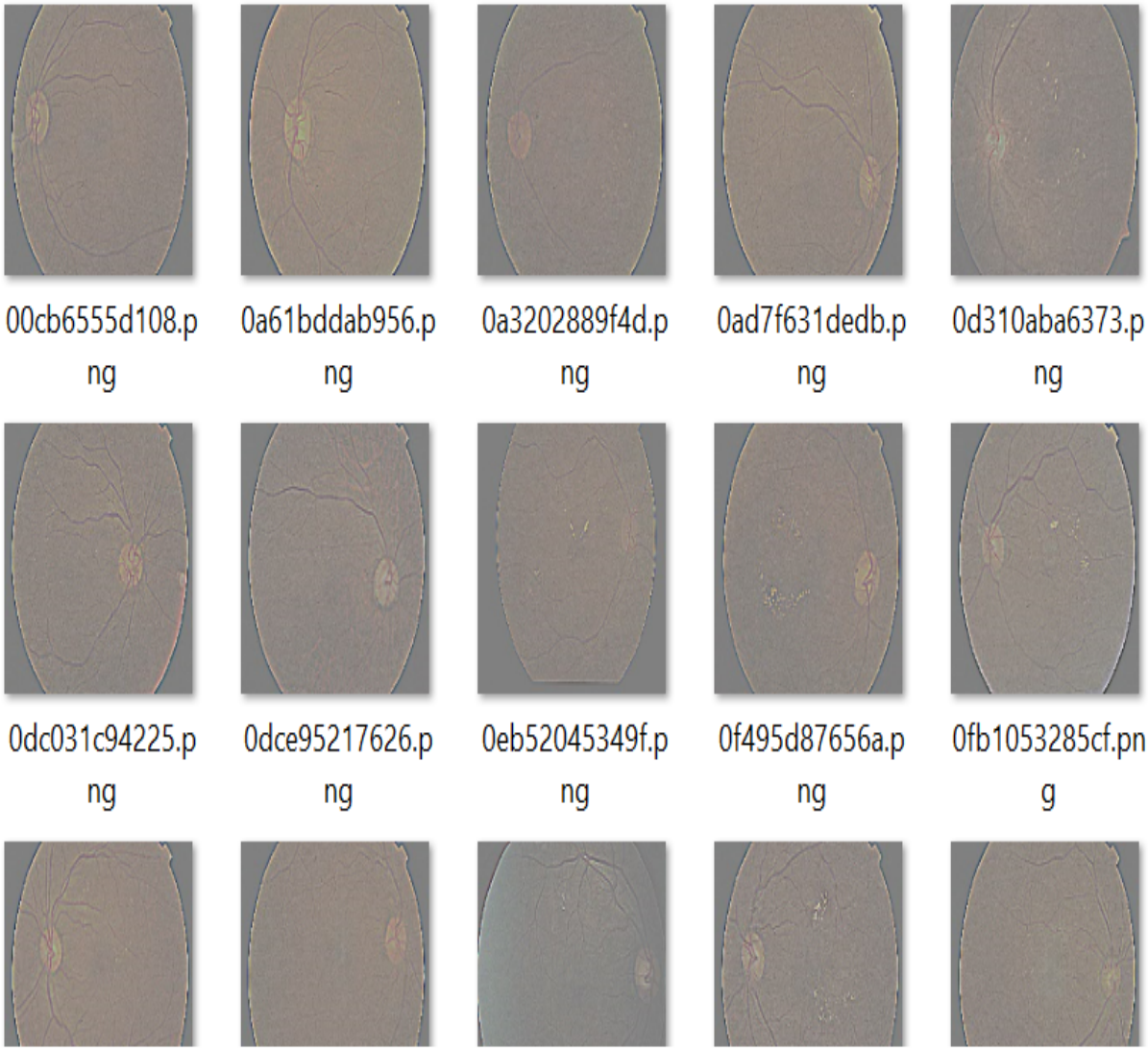
- Machine Learning Algorithms
- Python (numpy)
- HTML
- Flask
- openCV
- Tensorflow
- Keras

We developed this diabetic retinopathy classifier by using Jupyter notebook, flask application in python for which we used spyder and finally we used command prompt for running the application. The anaconda package was used.

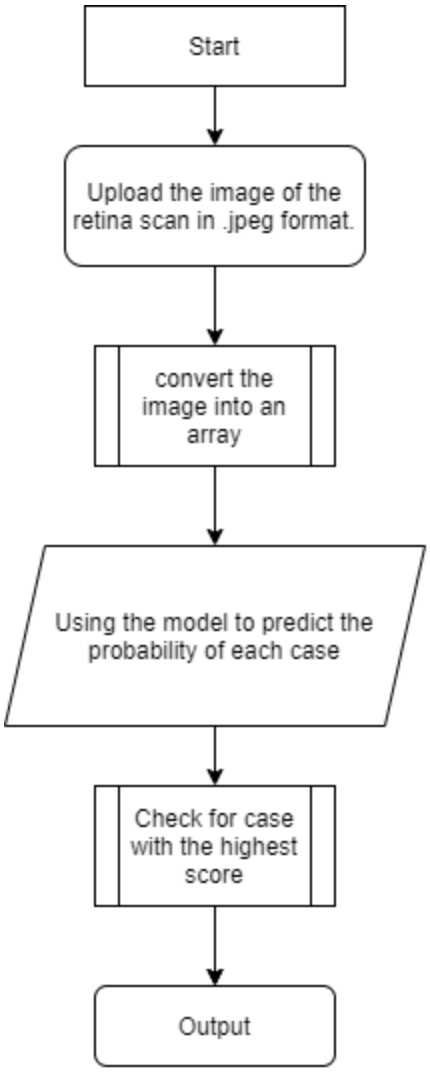
For creating an user interface for the prediction we used the Flask. It is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions, and a scripting language to create a webpage is HTML by creating the templates to use in th functions of the Flask and HTML.

4.Experimental Investigation

The dataset for this project has been taken from the Gaussian-filtered images for diabetic retinopathy in kaggle dataset repository. This dataste has all images already smoothened by applying gaussian filter. This dataset has two different folders i.e., train and test folders which inturn have five folders each namely mild, moderate, proliferate_DR, no_DR and severe. The ratio of train and test folders is 80:20. The train folder has 2929 images and test dataset has 733.


00cb6555d108.png
0a61bddab956.png
0a3202889f4d.png
0ad7f631dedb.png
0d310aba6373.png
0dc031c94225.png
0dce95217626.png
0eb52045349f.png
0f495d87656a.png
0fb1053285cf.png

5.Flowchart



6.Result

In this project, we use a Convolution Neural Network with a simple 2 layer design and cateragorical crossentropy to adjust the weights every training cycle. Each epoch as 92 training cycles/steps. The hidden layer uses Relu activation function while the out put layer uses softmax activation function. The model has been trained for exactly 100 epochs. We use categorical accuracy as a metric to determine the accuracy of the model. The final trained model has and estimated prediction accuracy of 89% which is very high. It is understood that this accuracy may be improved upon by training the model for more epochs and feeding it a larger dataset.

```
Epoch 24/24
92/92 [==============================] - 175s 2s/step - loss: 0.2967 - categorical_accuracy: 0.8928 -
accuracy: 0.8928 - val_loss: 0.2962 - val_categorical_accuracy: 0.8894 - val_accuracy: 0.8894

<tensorflow.python.keras.callbacks.History at 0x26d04088d48>
```

7.Advantages and Disadvantages

Advantages:

- Easy and simple to use User Interface for doctors and medical professionals.
- The model is trained over 100 epochs and provides an accuracy of 89% using categorical crossentropy loss for tweaking the weights every epoch.
- Provides quick and accurate results.
- Composed using html and Python for real time applications on web and wide compatibility.

Disadvantages:

- 11% margin for inaccurate predictions.
- External equipment needed for obtaining a retinal scan.
- The picture uploaded for prediction must be in .png format and at least 224*224 in dimension.

8.Applications

- It can be use at clinics and hospitals to get a quick estimate of the severity of the Diabetic retinopathy of the patient before the Doctor can proceed with further detailed checks or provide immediate treatment.
- It can be used by any small clinic with an retina scanning equipment and if cases are estimated to be severe, the can be recommended to be treated at a better equipped Hospital or treated locally if the severity is estimated to be less.

9.Conclusion

In this project, the CNN algorithm is adopted to build a UI model for predicting the severity of Diabetic retinopathy, by predicting based on the images of retinal scans. The convolution neural network has a simple 2-layer design and uses categorical cross entropy for tweaking its wights every epoch. The model has been trained for a total of 100 epochs over a large dataset. The model achieves a prediction accuracy of 89% which is very high and delivers the results immediately. Therefore, it can be concluded that training a large dataset over 100 epochs in CNN can yield highly accurate models.

10.Future Scope

In the future, as the application is used more by different users, the application simultanes is collection data in a specified folder, this data can later be verified and labelled by the same medical professionals and can be collectively be used to tain the model again and again over a period of time to improve the accuracy of the model. Subsequently this data will also be used for experimental and research purposes.

11.Bibliography

- https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function
- https://keras.io/api/layers/activations
- https://www.kaggle.com/c/diabetic-retinopathy-detection
- http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

12.Appendix

HTML:

```
1 {% extends "base.html" %} {% block content %}
2
```

```
<h2>Diabetic Retinopathy Prediction</h2>

<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <label for="imageUpload" class="upload-label">
            Choose...
        </label>
        <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
    </form>

    <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div>
        </div>
        <div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
        </div>
    </div>

    <div class="loader" style="display:none;"></div>

    <h3 id="result">
        <span> </span>
    </h3>

</div>

{% endblock %}
```

```
<html lang="en">
```

```
2
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Artificial Intelligence</title>
8      <link
   href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap
   .min.css" rel="stylesheet">
9      <script
   src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.mi
   n.js"></script>
10     <script
   src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></
   script>
11     <script
   src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.m
   in.js"></script>
12     <link href="{{ url_for('static',
   filename='css/main.css') }}" rel="stylesheet">
13 </head>
14
15 <body>
16     <nav class="navbar navbar-dark bg-dark">
17         <div class="container">
18             <a class="navbar-brand" href="#">Artificial
   Intelligence</a>
19             <button class="btn btn-outline-secondary my-2
   my-sm-0" type="submit">Help</button>
20         </div>
21     </nav>
22     <div class="container">
23         <div id="content" style="margin-top:2em">{% block
   content %}{% endblock %}</div>
24     </div>
```

```
</body>

<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>

</footer>

</html>
```

App.py:

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import numpy as np
from keras.preprocessing import image


from keras.applications.imagenet_utils import preprocess_input, decode_predictions

from keras.models import load_model
from keras import backend
from tensorflow.keras import backend

import tensorflow as tf

#global graph
#graph=tf.compat.v1.get_default_graph()
```

```
20
21 #global graph
22 #graph = tf.get_default_graph()
23
24
25 from skimage.transform import resize
26
27 # Flask utils
28 from flask import Flask, redirect, url_for, request,
   render_template
29 from werkzeug.utils import secure_filename
30 from gevent.pywsgi import WSGIServer
31
32 # Define a flask app
33 app = Flask(__name__)
34
35 # Model saved with Keras model.save()
36 MODEL_PATH = 'model.h5'
37
38 # Load your trained model
39 model = load_model(MODEL_PATH)
40         # Necessary
41 # print('Model loaded. Start serving...')
42
43 # You can also use pretrained model from Keras
44 # Check https://keras.io/applications/
45 #from keras.applications.resnet50 import ResNet50
46 #model = ResNet50(weights='imagenet')
47 #model.save('')
48 print('Model loaded. Check http://127.0.0.1:5000/')
49
50
51
52
53 @app.route('/', methods=['GET'])
54 def index():
```

```
    # Main page
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        #with graph.as_default():
        preds = model.predict_classes(x)
        index = ['Mild','Moderate','No_DR','Proliferate_DR','Severe']
        text = index[preds[0]]

                # ImageNet Decode



        return text


if __name__ == '__main__':
    app.run(debug=False,threaded = False)
```

```
88
89
90
```