Deep Learning Fundus Image Analysis For Early Detection Of Diabetic Retinopathy

Diabetic Retinopathy (DR) is a common complication of diabetes mellitus, which causes lesions on the retina that affect vision. If it is not detected early, it can lead to blindness. Unfortunately, DR is not a reversible process, and treatment only sustains vision. DR early detection and treatment can significantly reduce the risk of vision loss. The manual diagnosis process of DR retina fundus images by ophthalmologists is time, effort and cost-consuming and prone to misdiagnosis unlike computer-aided diagnosis systems.

Transfer learning has become one of the most common techniques that has achieved better performance in many areas, especially in medical image analysis and classification. We used Transfer Learning techniques like Inception V3,Resnet50,Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

Project Objectives

- Know fundamental concepts and techniques of transfer learning like Xception.
- Gain a broad understanding of image data.
- Know how to pre-process/clean the data using different data pre-processing techniques.
- Know how to build a web application using the Flask framework.

Project Flow

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image analyzed by the model which is integrated with flask application.
- The Xception Model analyzes the image, then the prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
    - Create a Train and Test path.
- Data Pre-processing.
    - Import the required library
    - Configure ImageDataGenerator class
    - ApplyImageDataGenerator functionality to Trainset and Testset
- Model Building
    - Pre-trained CNN model as a Feature Extractor
    - Adding Dense Layer
    - Configure the Learning Process
    - Train the model
    - Save the Model
    - Test the model
- Cloudant DB
    - Register & Login to IBM Cloud
    - Create Service Instance
    - Creating Service Credentials

    - Launch Cloudant DB
    - Create Database
- Application Building
    - Create an HTML file
    - Build Python Code

Download The Dataset

In our project, we are not going to upload the dataset on colab. We are going to clone the kaggle dataset on colab. Kindly refer this - https://www.analyticsvidhya.com/blog/2021/06/how-to-load-kaggle-datasets-directly-into-google-colab/

Create Training And Testing Path

To build a DL model we have to split training and testing data into two separate folders. But In the project dataset folder training and testing folders are presented. So, in this case we just have to assign a variable and pass the folder path to it.

Four different transfer learning models are used in our project and the best model (Xception) is selected.

The image input size of xception model is 299, 299.

Configure ImageDataGenerator Class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

ImageDataGenerator:

Let us apply ImageDataGenerator functionality to the Train set and Test set by using the following code. For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories

Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 64.

- target_size: Size to resize images after they are read from disk.
- class_mode:
  - ‘int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
  - None (no labels).

Model Building

Now it's time to build our model. Let’s use the pre-trained model which is Xception, one of the convolution neural net (CNN) architecture which is considered as a very good model for Image classification.

Deep understanding on the Xception model – Link is referred to in the prior knowledge section. Kindly refer to it before starting the model building part.

Pre-Trained CNN Model As A Feature Extractor

For one of the models, we will use it as a simple feature extractor by freezing all the five convolution blocks to make sure their weights don’t get updated after each epoch as we train our own model.

Here, we have considered images of dimension (229,229,3).

Also, we have assigned include_top = False because we are using convolution layer for features extraction and wants to train fully connected layer for our images classification(since it is not the part of Imagenet dataset)

Flatten layer flattens the input. Does not affect the batch size.

Adding Dense Layers

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

Let us create a model object named model with inputs as xception.input and output as dense layer.

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

Configure The Learning Process

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

Train The Model

Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 10 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network

Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

- Epochs: an integer and number of epochs we want to train our model for.

- validation_data can be either:
  - an inputs and targets list
  - a generator
  - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

Save The Model

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Cloudant DB

Create Service Instance in IBM Cloud

Creating Service Credentials

reate Database

Activity 5:- Create Database

- In order to manage a connection from a local system you must first initialize the connection by constructing a Cloudant client.We need to import the cloudant library.

- IBM Cloud Identity & Access Management enables you to securely authenticate users and control access to all cloud resources consistently in the IBM Bluemix Cloud Platform.
- Once a connection is established you can then create a database, open an existing database.
- Create a database as my_database.

Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided to the user where he has uploaded the image.Based on the saved model, the uploaded image will be analyzed and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

Building Html Pages

For this project create three HTML files namely

- index.html
- register.html
- login.html
- prediction.html
- logout.html

and save them in the templates folder.

Let’s see how our index.html page looks like:

Build Python Code:

Import the libraries

Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

Create a database using an initiated client.

Render HTML page:

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Configure the registration page

Based on user input into the registration form we stored it on data dictionary then we can validate the data using _id parameter with user input that we can store it on query variable then we can validate by passing the query variable into the my_database.get_user_result() method.Then we can check the docs length by using len(docs.all()) function.If the length of docs is 0 then user will register successfully on the platform and user data will store on the database.Otherwise its shows the message as user already registered please login and use our web application for DR prediction.

Configure the login page

Based on user input into the login form we stored user id and password into the (user,passw) variables. Then we can validate the credentials using _id parameter with user input that we can store it on query variable then we can validate by passing the query variable into the my_database.get_user_result() method.Then we can check the docs length by using len(docs.all()) function.If the length of doc is 0 then it means username is not found.Otherwise its validate the data that is stored on the database and check the username & password. If it's matched then the user will be able to login and use our web application for DR prediction.Otherwise the user needs to provide correct credentials.

The image is selected from uploads folder. Image is loaded and resized with load_img() method. To convert image to an array, img_to_array() method is used and dimensions are increased with expand_dims() method. Input is processed for xception model and predict() method is used to predict the probability of classes. To find the max probability np.argmax is used.

Run The Application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

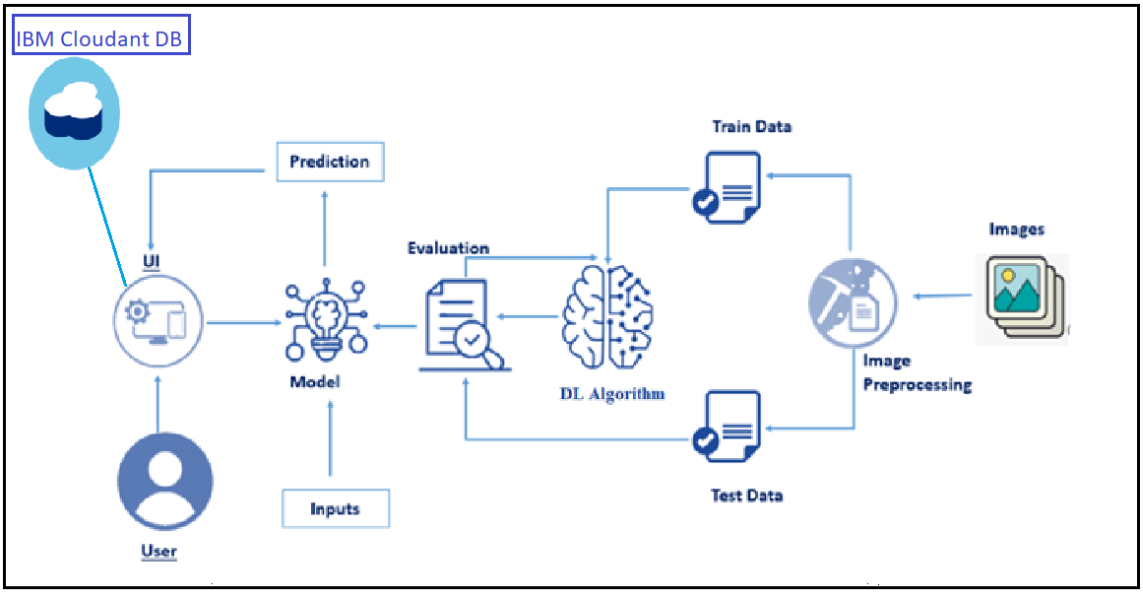
1: Run the application

In the anaconda prompt, navigate to the folder in which the flask app is present. When the python file is executed the localhost is activated on 5000 port and can be accessed through it.

2: Open the browser and navigate to localhost:5000 to check your application

After successfully login you will redirect to the prediction page where we have to upload the image to predict the outcomes.

After successfully login you will redirect to the prediction page where we have to upload the image to predict the outcomes


IBM Cloudant DB
Prediction
UI
Model
Evaluation
DL Algorithm
Train Data
Test Data
Images
Image
Preprocessing
Inputs
User