Diabetic Retinopathy using CNN – Source code

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Dropout
from tensorflow.keras.models import Model
import tensorflow as tf
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import tensorflow_datasets as tfds
import numpy as np
from sklearn.model_selection import train_test_split
import os
from glob import glob
import matplotlib.pyplot as plt
import cv2
import itertools
import random
from collections import Counter
from glob import iglob

IMAGE_SIZE = [512, 512]
train_path=r'C:\Users\Windows\Downloads\DR Notebook\val'
valid_path=r'C:\Users\Windows\Downloads\DR Notebook\dataset'

train_images = glob(os.path.join(train_path, '/.jpg'))
valid_images=  glob(os.path.join(valid_path, '/.jpg'))
```

```
total_images1 = len(train_images)
total_images2 = len(valid_images)
print('Train images:', total_images1)
print('Valid images:', total_images2)
image_count = []
class_names = []
print('Training images\n')
for folder in os.listdir(os.path.join(train_path)):
    folder_num = len(os.listdir(os.path.join(train_path, folder)))
    image_count.append(folder_num)
    class_names.append(folder)

    print('{:20s}'.format(folder), end=' ')
    print(folder_num)
print('\n')
print('Validating images\n')
for folder in os.listdir(os.path.join(valid_path)):
    folder_num = len(os.listdir(os.path.join(valid_path, folder)))

    print('{:20s}'.format(folder), end=' ')
    print(folder_num)

folders1 = glob(r'C:\Users\Windows\Downloads\DR Notebook\dataset\*')

folders2= glob(r'C:\Users\Windows\Downloads\DR Notebook\val\*')

train_datagen =ImageDataGenerator(rescale=1./512,
                        shear_range = 0.2,
                          zoom_range = 0.2,
```

```
                        horizontal_flip = True,
                        )
valid_datagen=ImageDataGenerator(rescale=1./255)

training_generator = train_datagen.flow_from_directory(
                        train_path,
                        target_size = (512, 512),
                        batch_size = 32,
                        class_mode = 'categorical'
                        )

validation_generator = valid_datagen.flow_from_directory(
                valid_path,
                target_size=(512,512),
                batch_size=32,
                class_mode='categorical'
                )

#input_tensor = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3))
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet',  include_top=False)

for layer in inception.layers:
   layer.trainable = False

x = Flatten()(inception.output)

#prediction = Dense(len(folders1), activation='softmax')(x)
prediction = Dense(5, activation='softmax')(x)
model = Model(inputs=inception.input, outputs=prediction)
```

```
model.summary()

model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

#r = model.fit_generator(
r = model.fit(
  training_generator,
  validation_data=validation_generator,
  epochs=50,
  steps_per_epoch=len(training_generator),
  validation_steps=len(validation_generator)
)

plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train accuracy')
plt.plot(r.history['val_accuracy'], label='val accuracy')
plt.legend()
plt.show()
```

```
plt.savefig('AccVal_acc')


from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model.save('DRmodel_inception.h5')


from PIL import Image
from skimage.io import imread
from skimage.transform import resize
import skimage
import os
def load_image(file):
    dimension=(512, 512)
    image = Image.open(file)
    flatten_data = []
    img = skimage.io.imread(file)
    img = resize(img, dimension)
    return img
def predict(image):
    probabilities = model.predict(np.asarray([img]))[0]
    class_idx = np.argmax(probabilities)
    return class_names[class_idx]


%matplotlib inline
img = load_image(r'C:\Users\User\Documents\Python Projects\Diabatic Retinopathy
RCNN\CodeDR\1170_left.jpeg')
prediction = predict(img)
plt.imshow(img)
plt.show()
```

```
test = class_names


print("Predicted Disease is", (prediction))
```