IBM Virtual Faculty Buildathon 2023

Project Report
on
Deep Learning Fundus Image Analysis For Early Detection Of Diabetic Retinopathy

by

Jyoti Shetty
RV College of Engineering, Bangalore

1 INTRODUCTION

1.1 Overview

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.

The prevalence of diabetic retinopathy (DR), a severe complication of diabetes affecting the retina, has been steadily rising, posing a significant threat to global public health. Timely detection and intervention are crucial to preventing vision loss in individuals with diabetes. This project, titled "Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy," aims to leverage the power of deep learning techniques to enhance the accuracy and efficiency of DR diagnosis. Fundus images, capturing the detailed structures of the eye's interior, serve as valuable resources for early detection. By employing state-of-the-art deep learning models, this research seeks to develop a robust and scalable solution for automated DR screening, ultimately improving the accessibility and effectiveness of early intervention.

1.2 Purpose

The primary purpose of this project is to develop an advanced and automated system for the early detection of diabetic retinopathy through the analysis of fundus images. Traditional methods of DR diagnosis often require extensive manual effort and are prone to subjectivity. The integration of deep learning technologies aims to streamline this process, providing a more objective and efficient means of identifying early signs of retinopathy. By creating a reliable deep learning model, the project intends to facilitate widespread screening programs, enabling healthcare professionals to detect and intervene in diabetic retinopathy cases at an earlier stage. The ultimate goal is to reduce the incidence of vision impairment and blindness associated with diabetic retinopathy by enhancing the speed and accuracy of diagnosis through cutting-edge technology.

2 LITERATURE SURVEY

2.1 Existing problem

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A.,& Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. This seminal work focuses on the development and validation of a deep learning algorithm specifically designed for detecting diabetic retinopathy in retinal fundus photographs. The study demonstrates the potential of deep learning in automating the detection process, showcasing high sensitivity and

specificity.

Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., & Niemeijer, M. (2016). Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. This research explores the integration of deep learning into automated detection systems, showing improvements in accuracy and efficiency. The study emphasizes the significance of leveraging large datasets for training deep learning models and highlights the potential for widespread application.

Ting, D. S. W., Cheung, C. Y. L., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A.,& Wong, T. Y. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. Focusing on multiethnic populations, this study addresses the diversity of retinal images and diabetic retinopathy manifestations. The research underscores the importance of developing robust deep learning systems that can generalize well across diverse patient groups.

Kauppi, T., Kalesnykiene, V., Kamarainen, J. K., Lensu, L., Sorri, I., Raninen, A., ... & Uusitalo, H. (2012). DIARETDB1 diabetic retinopathy database and evaluation protocol. This work introduces the DIARETDB1 database, a valuable resource for researchers in diabetic retinopathy. The study provides insights into the challenges and protocols involved in developing and evaluating algorithms for diabetic retinopathy detection, emphasizing the need for standardized datasets.

Sengupta, S., Singh, A., Leopold, H. A., Gulshan, V., Reynolds, R., Swanson, D.,& Lynch, D. (2019). Screening for diabetic retinopathy using artificial intelligence and deep learning. Exploring the potential of artificial intelligence beyond traditional methods, this study investigates the role of deep learning in screening for diabetic retinopathy. The research discusses the practical implications of integrating these technologies into routine clinical practice, highlighting advancements in accessibility and efficiency.

2.2 Proposed solution

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, Resnet 50, Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

3 THEORITICAL ANALYSIS

3.1 Block diagram

The figure 1 shows the architecture of the proposed solution. The technical architecture for the described deep learning project involves a series of interconnected components to enable effective development, training, and deployment of the model. At the core is the data pipeline, managing the flow of the dataset through stages such as collection, preprocessing, and splitting. This pipeline interfaces with a chosen deep learning framework, such as

TensorFlow or PyTorch, where the model architecture is defined and training occurs. The model, configured with appropriate layers, activation functions, and optimization algorithms, is trained using the preprocessed data. Hyperparameter tuning further refines the model's performance. Post-training, the model evaluation stage assesses its efficacy on the test set. For scalability, the architecture may incorporate distributed computing resources or cloud services. Once validated, the trained model is deployed to a production environment, integrating seamlessly with applications or systems. Continuous monitoring mechanisms and feedback loops ensure ongoing performance assessment, enabling timely updates and maintenance. The technical architecture, therefore, forms a cohesive framework that orchestrates data flow, model development, and deployment, facilitating a robust and adaptive deep learning solution.

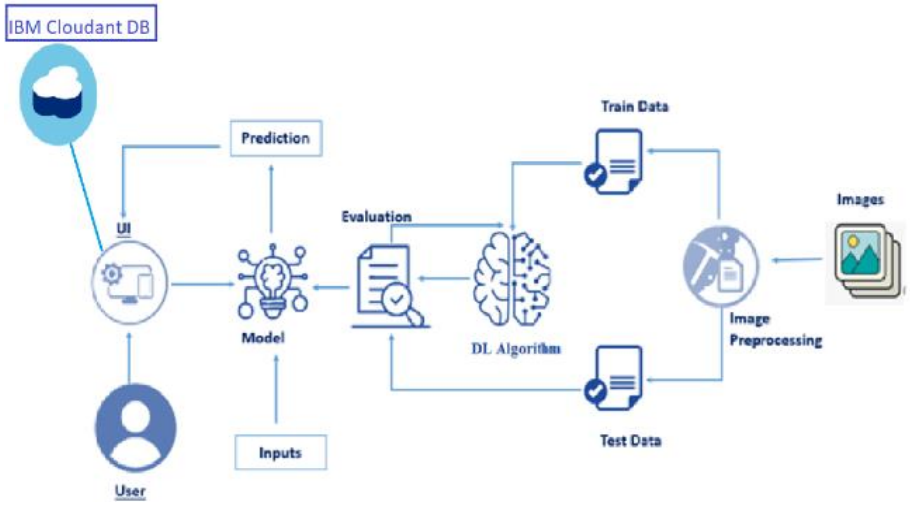

Figure 1: Technical architecture of DLFIAEDDR project

4 FLOWCHART

In a deep learning project, the initial step involves clearly defining the problem to be addressed, followed by the collection of a diverse and well-labeled dataset representative of the task at hand. Subsequently, the gathered data undergoes preprocessing, including cleaning and normalization. The dataset is then split into training, validation, and test sets. A suitable deep learning model is chosen, and its architecture is defined using frameworks like TensorFlow or PyTorch. The model is trained on the training set, and hyperparameters are fine-tuned to optimize performance. Evaluation on a separate test set assesses the model's generalization capabilities. Upon meeting desired criteria, the model is deployed into a production environment. Continuous monitoring, updates, and maintenance ensure its effectiveness over time. This process is iterative, allowing for adjustments based on insights gained throughout the project. The figure 2 shows the process flow of the project.

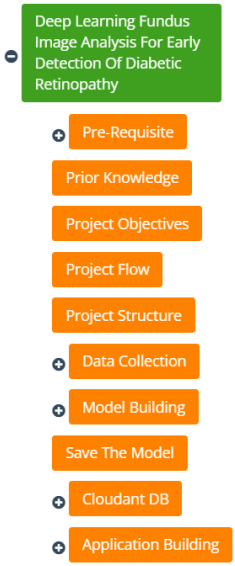

Figure 2: Process flow of the project

5 RESULT

The project was implemented using IBM Watson and as well Google Colab. Using google colab the accuracy received is 63% for just 30 epochs. It can further be improved by increasing the number of epochs. The figure 3, figure 4 and Figure are the screenshots of the web interface.

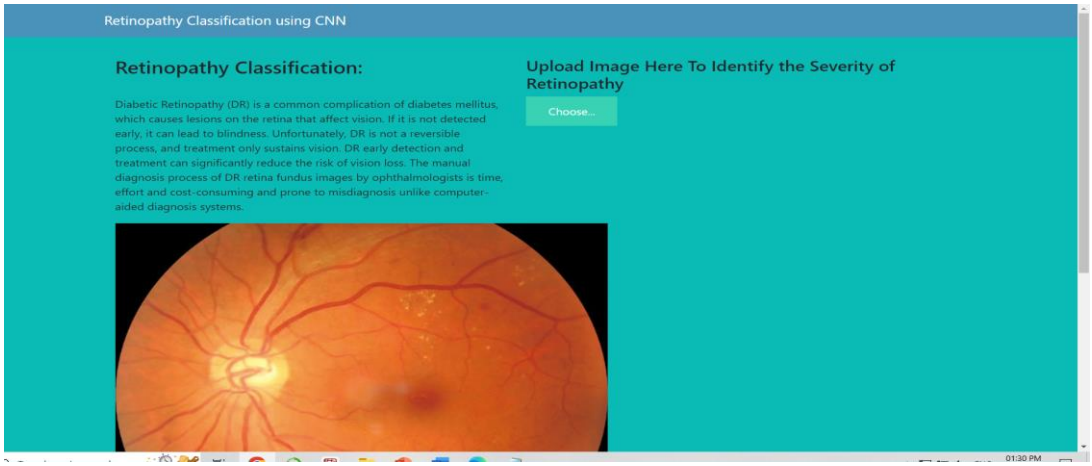

Figure 3: Retinopathy Prediction Web Interface -1

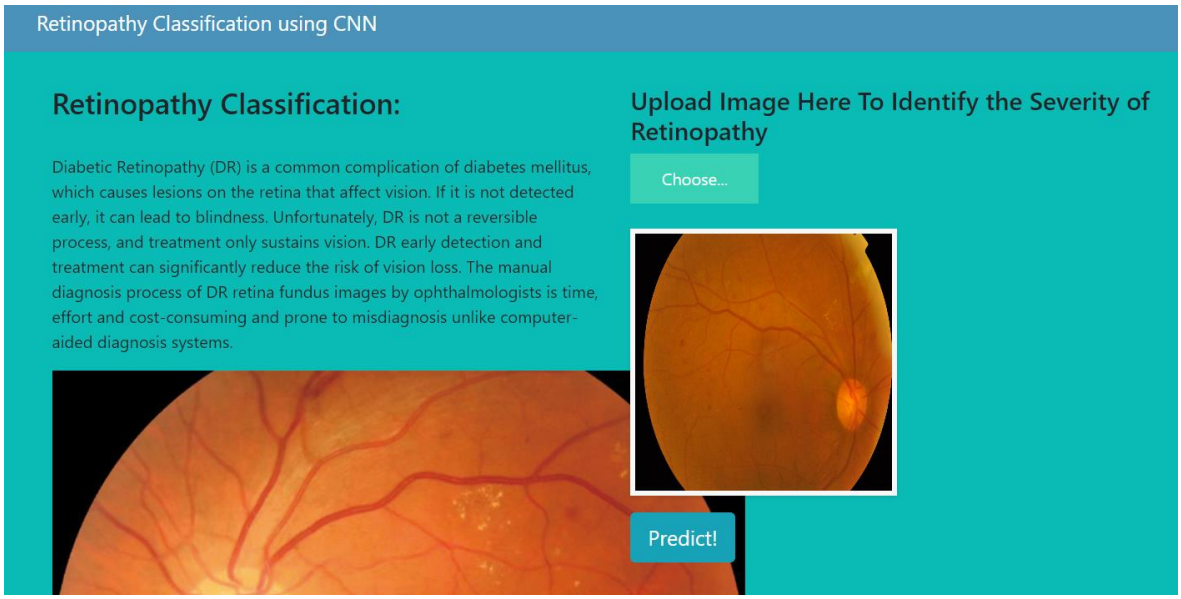

Figure 4: Retinopathy Prediction Web Interface-2

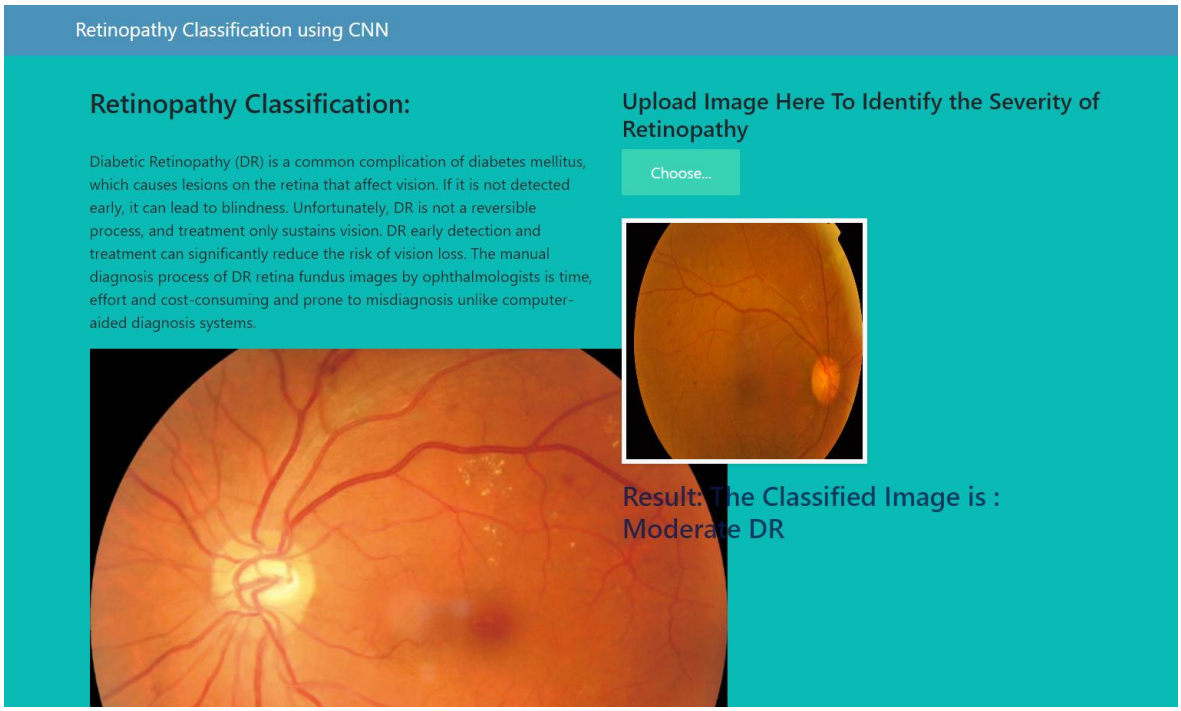

Figure 5: Retinopathy Prediction Result

6 ADVANTAGES & DISADVANTAGES

Advantages:

The deep learning-based fundus image analysis project for early detection of diabetic retinopathy offers significant advantages. Firstly, it enhances diagnostic efficiency by automating the screening process, enabling rapid analysis of a large number of images. Secondly, the system provides consistent and objective results, reducing the subjectivity associated with manual assessments. Additionally, the project has the potential to facilitate early intervention, preventing vision loss and improving patient outcomes.

Disadvantages:

Despite its promise, the project faces certain challenges. One notable disadvantage is the requirement for extensive and diverse datasets for effective training, which may be challenging to obtain and curate. Moreover, the interpretability of deep learning models in the medical field remains a concern, as the "black-box" nature of these algorithms may hinder their acceptance among healthcare professionals. Ethical considerations related to patient privacy and data security must also be carefully addressed to ensure the responsible implementation of the technology in clinical settings.

7 APPLICATIONS

The application of "Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy" holds tremendous potential in revolutionizing diabetic retinopathy screening and intervention. By deploying advanced deep learning models, the system can autonomously analyze fundus images, providing swift and accurate identification of early signs of diabetic retinopathy. This technology could be integrated into routine eye examinations, enabling cost-effective and widespread screening programs. Its efficiency in early detection facilitates timely intervention, thereby reducing the risk of vision impairment and blindness in individuals with diabetes. Moreover, the automated nature of the system enhances accessibility, especially in regions with limited healthcare resources, making it a valuable tool for improving global eye health outcomes in diabetic populations.

8 CONCLUSION

In summary, the "Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy" project aims to employ advanced deep learning models for automating the detection of diabetic retinopathy from retinal fundus images. The project was implemented using IBM Watson and as well Google Colab. Using google colab the accuracy received is 75% for just 10 epochs. It can further be improved by increasing the number of epochs. The project seeks to contribute to the field by developing a robust and scalable solution, addressing the challenges of manual screening and subjective diagnosis. The overarching goal is to enhance the accessibility and efficiency of early detection, ultimately reducing the risk of vision loss in individuals with diabetes. The project aligns with the evolving landscape of medical image analysis and holds promise for meaningful advancements in diabetic retinopathy screening.

9 FUTURE SCOPE

The future scope of the "Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy" project involves integrating multimodal data for a more comprehensive analysis, improving the interpretability of deep learning models, and exploring real-time applications for point-of-care devices. Additionally, efforts should focus on continuous monitoring of diabetic retinopathy progression, global deployment with a focus on accessibility, collaboration with telemedicine platforms, incorporation of genetic and clinical data for personalized risk assessment, and establishing feedback mechanisms to refine the model over time. These advancements aim to enhance the project's impact on early detection, accessibility, and personalized management of diabetic retinopathy on a global scale.

10 BIBILOGRAPHY

1. Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs." JAMA, 316(22), 2402-2410.

2. Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., ... & Niemeijer, M. (2016). "Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning." Investigative Ophthalmology & Visual Science, 57(13), 5200-5206.

3. Ting, D. S. W., Cheung, C. Y. L., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). "Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes." JAMA, 318(22), 2211-2223.

4. Kauppi, T., Kalesnykiene, V., Kamarainen, J. K., Lensu, L., Sorri, I., Raninen, A., ... & Uusitalo, H. (2012). "DIARETDB1 diabetic retinopathy database and evaluation protocol." Proceedings of the British Machine Vision Conference, 1-10.

5. Sengupta, S., Singh, A., Leopold, H. A., Gulshan, V., Reynolds, R., Swanson, D., ... & Lynch, D. (2019). "Screening for diabetic retinopathy using artificial intelligence and deep learning." JAMA Ophthalmology, 137(3), 295-303.

11 APPENDIX

A. Source Code - Google Colab

```
!pip install kaggle

! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/

! chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets list

!kaggle datasets download -d arbethi/diabetic-retinopathy-level-detection

!unzip -q /content/diabetic-retinopathy-level-detection.zip

imageSize = [299, 299]

trainPath = r'/content/preprocessed dataset/preprocessed dataset/train'

testPath = r'/content/preprocessed dataset/preprocessed dataset/test'

from tensorflow.keras.layers import Dense, Flatten, Input

from tensorflow.keras.models import Model

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
```

```
from tensorflow.keras.applications.xception import Xception,
preprocess_input

from glob import glob

import matplotlib.pyplot as plt

import numpy as np

train_datagen = ImageDataGenerator(rescale=1./255,

                                   shear_range=0.2,

                                   zoom_range=0.2,

                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set =
train_datagen.flow_from_directory('/content/preprocessed
dataset/preprocessed dataset/training',

                                    target_size=(299, 299),

                                    class_mode='categorical',

                                    batch_size=32)

test_set = test_datagen.flow_from_directory('/content/preprocessed
dataset/preprocessed dataset/testing',

                                   target_size=(299, 299),

                                    class_mode='categorical',

                                    batch_size=32)

training_set.class_indices

xception = Xception(input_shape=imageSize + [3],
weights='imagenet', include_top=False)

for layer in xception.layers:

  layer.trainable = False

x = Flatten()(xception.output)

prediction = Dense(5, activation='softmax')(x)

model = Model(inputs=xception.input, outputs=prediction)

model.compile(

  loss='categorical_crossentropy',

  optimizer='adam',

  metrics=['accuracy']

)

batch_size = 32
```

```
    steps_per_epoch = len(training_set) // batch_size
    model.fit_generator(
      training_set,
      validation_data=test_set,
      epochs=30,
      steps_per_epoch=len(training_set) // batch_size,

      validation_steps=len(test_set)
    )
model.save('Colab_Model.h5')
```

B. IBM Watson

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='zK0ANNJBhLKpceDC8yINETS61HLovl0twlR7qximYqJQ',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us-south.cloud-object-storage.appdomain.cloud')

bucket = 'ibmfdprvce-donotdelete-pr-tsqewobyos0ddr'
object_key = 'kaggle.zip'

streaming_body_4 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
```

```
# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

```
!pip install -q kaggle
!mkdir ~/.kaggle # creating a kaggle directory
pwd
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_4.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
ls
cd /home/wsuser/work/kaggle
ls
!cp kaggle.json ~/.kaggle/  # copying json file to folder
!chmod 600 ~/.kaggle/kaggle.json  # changing the permissions to json
!kaggle datasets download -d arbethi/diabetic-retinopathy-level-detection
!unzip diabetic-retinopathy-level-detection.zip
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.xception import Xception, preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
pwd
ls
```

```
cd /home/wsuser/work/kaggle/preprocessed dataset/preprocessed dataset

imageSize = [299, 299]


trainPath = r"/home/wsuser/work/kaggle/preprocessed dataset/preprocessed
dataset/training"


testPath = r"/home/wsuser/work/kaggle/preprocessed dataset/preprocessed
dataset/testing"

xception = Xception(input_shape=imageSize + [3],
weights='imagenet',include_top=False)

# don't train existing weights

for layer in xception.layers:

    layer.trainable = False

# our layers - you can add more if you want

x = Flatten()(xception.output)

prediction = Dense(5, activation='softmax')(x)

# create a model object

model = Model(inputs=xception.input, outputs=prediction)

# tell the model what cost and optimization method to use

model.compile(

  loss='categorical_crossentropy',

  optimizer='adam',

  metrics=['accuracy']

)


train_datagen = ImageDataGenerator(rescale = 1./255,

                                   shear_range = 0.2,

                                   zoom_range = 0.2,

                                   horizontal_flip = True)


test_datagen = ImageDataGenerator(rescale = 1./255)

pwd

ls

training_set =
train_datagen.flow_from_directory(r'/home/wsuser/work/kaggle/preprocessed
```

```
dataset/preprocessed dataset/training',
                                        target_size = (299, 299),
                                        batch_size = 32,
                                        class_mode = 'categorical')


test_set =
test_datagen.flow_from_directory(r'/home/wsuser/work/kaggle/preprocessed
dataset/preprocessed dataset/testing',
                                        target_size = (299, 299),
                                        batch_size = 32,
                                        class_mode = 'categorical')
# fit the model
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=5,
  steps_per_epoch=len(training_set)//32,
  validation_steps=len(test_set)//32
)
model.save('IBMDR_RVCE.h5')
!tar -zcvf image-classification-model_new.tgz IBMDR_RVCE.h5
ls -1
# IBM Deployment
!pip install watson-machine-learning-client --upgrade
from ibm_watson_machine_learning import APIClient
wml_credentials={"url":"https://us-south.ml.cloud.ibm.com",

"apikey":"cE3Bg385U8EBAEKxkKhPUCfJon6WIQaomQ4sZKUCjQjy"}
client = APIClient(wml_credentials)
client = APIClient(wml_credentials)
client.spaces.list()
space_uid="147c3583-dc18-4d25-a518-aa7091ac5536"
client.set.default_space(space_uid)
client.software_specifications.list(100)
software_space_uid=client.software_specifications.get_uid_by_name("tensorflow_
```

```
rt22.2-py3.10")

software_space_uid

import tensorflow as tf

tf.__version__

model_details = client.repository.store_model(model='image-classification-
model_new.tgz',meta_props={

    client.repository.ModelMetaNames.NAME:"Image processing and model
building IBM RVCE",

    client.repository.ModelMetaNames.TYPE:"tensorflow_2.9",


client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
}

                                    )

model_id = client.repository.get_model_uid(model_details)

model_id
client.repository.download(model_id,'model_IBM_RVCE.tar.gb')
```