ALZHEIMER’S DISEASE PREDICTION

INTRODUCTION

1.1. OVERVIEW

Alzheimer's disease is a progressive and irreversible neurological disorder that affects the brain, leading to memory loss, cognitive decline, and behavioral changes. It is the most common cause of dementia, accounting for 60-80% of all cases. Alzheimer's disease typically begins with mild memory problems and confusion, which gradually worsen over time. As the disease progresses, individuals may experience difficulty with language, problem-solving, and completing everyday tasks.

A treatment given at an early stage of AD is more effective, and it causes fewer minor damage than a treatment done at a later stage. Several techniques such as Decision Tree, Random Forest, Support Vector Machine, Gradient Boosting, and Voting classifiers have been employed to identify the best parameters for Alzheimer's disease prediction.
In Alzheimer's disease, the brain starts shrinking, and over time it converts into dementia. The diagnosis of dementia takes an ample amount of time, around 2.8 to 4.4 years after the first clinical symptoms arise. Alzheimer's disease cannot be cured by any pharmacologic therapies (drugs) now on the market. Alzheimer's disease can only be avoided by early detection and prompt treatment.

Predictions of Alzheimer's disease are based on Open Access Series of Imaging Studies (OASIS) data, and performance is measured with parameters like Precision, Recall, Accuracy, and F1-score for ML models. The proposed classification scheme can be used by clinicians to make diagnoses of these diseases. It is highly beneficial to lower annual mortality rates of Alzheimer's disease in early diagnosis with these ML algorithms. The proposed work shows better results with the best validation average accuracy of 83% on the test data of AD. This test accuracy score is significantly higher in comparison with existing works.

1.2 PURPOSE

Automated systems are more accurate than human assessment and can be used in medical decision support systems because they are not subject to human errors. Based on previous research on AD, researchers have applied images (MRI scans), biomarkers (chemicals, blood flow), and numerical data extracted from the MRI scans to study this Disease. As such, they were able to determine whether a person was demented or not. In addition to shortening diagnosis time, more human interaction will be reduced by automating Alzheimer's diagnosis. In addition, automation reduces overall costs and provides more accurate results. For example, we can predict whether a patient is demented by analyzing MRI scans and applying prediction techniques.

LITERATURE SURVEY

2.1 EXISTING PROBLEM

The absence of precise and early diagnostic techniques is one issue that currently exists in Alzheimer's disease prediction. Clinical signs and cognitive tests are currently the main methods used to diagnose Alzheimer's disease, although these methods may not be sensitive enough to detect the illness in its early stages. When symptoms start to show up, there may already be serious brain damage. The complexity and variability of Alzheimer's disease present another difficulty. There are numerous subtypes and phases of the disease, and various people's symptoms vary from one another. Because of this intricacy, it is challenging to create a universal prediction model that can consistently detect and forecast the course of the disease. Furthermore, there aren't many large-scale, diversified datasets available for training predictive models.

2.2 PROPOSED SOLUTION

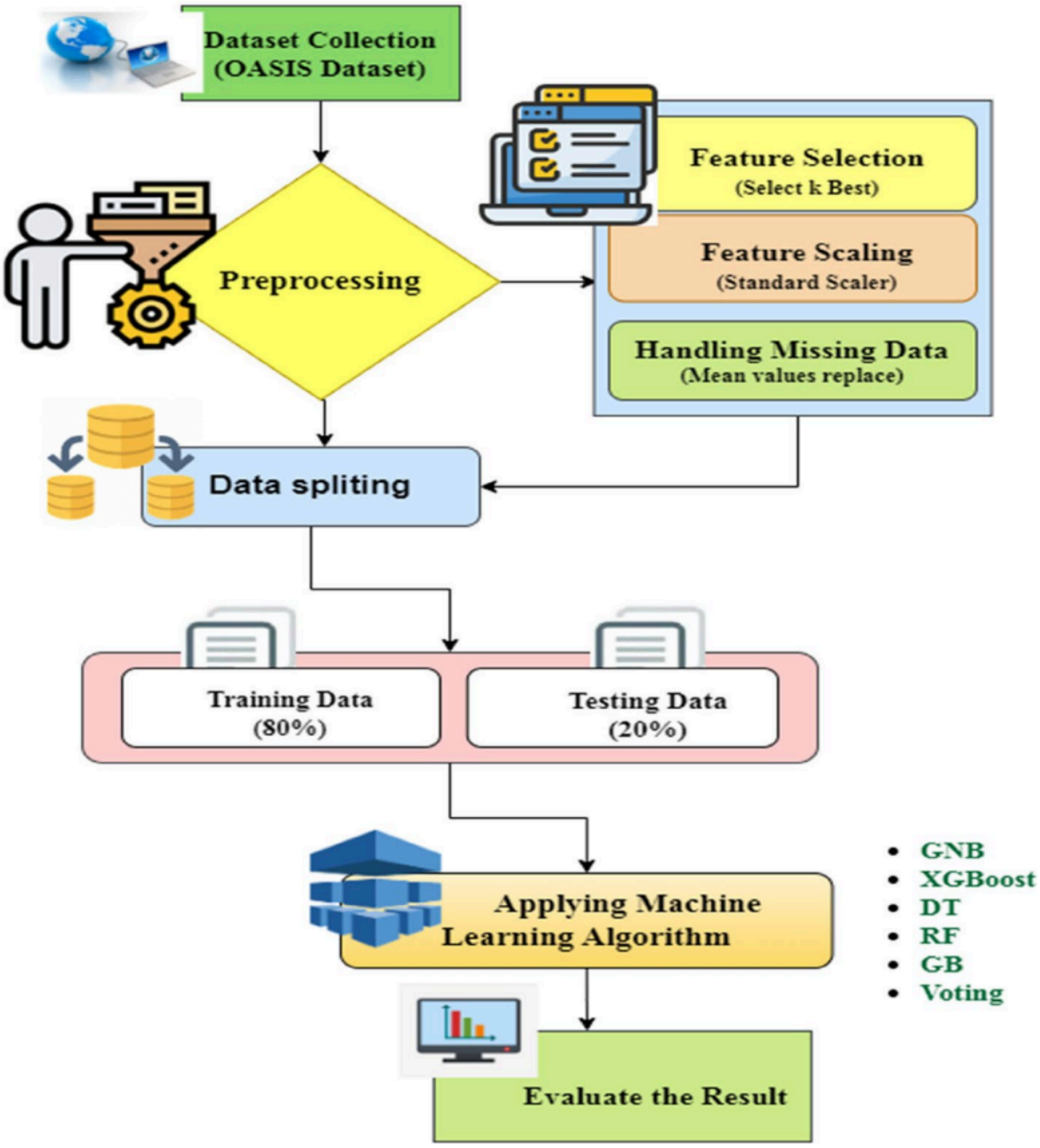
This paper proposes deep transfer learning models and MRI (Magnetic Resonance Imaging) images to detect the multiple stages of Alzheimer's disease such as "Very-Mild -Demented," "Mild-Demented," "Moderate-Demented," and "No-Demented." Data preprocessing and augmentation process are applied, enabling the model to detect the correct class of Alzheimer's disease. Then further deep transfer learning models are used to classify and predict the early stages of Alzheimer's disease

Efforts to establish large-scale and diverse longitudinal datasets are essential for training and validating predictive models. Collaborative initiatives that pool data from different research centers and include diverse populations can help overcome the challenge of data scarcity. CNNs are effective in processing image-based data, such as brain MRI scans. GoogLeNet, introduced in 2014, introduced the concept of inception modules, which employ multiple filter sizes within the same layer to capture features at different scales. It significantly reduced the number of parameters compared to previous architectures. GoogLeNet won the ILSVRC in 2014 and was notable for its efficiency.

EXPERIMENTAL INVESTIGATIONS

The performance of Xception model gives the accuracy of 71% and the model predicts results effectively. The Dataset provided in Kaggle has an unbalanced dataset that was oversampled to achieve a neutral dataset Using a web application, clinicians and patients may remotely screen for Alzheimer's disease. According to the AD spectrum, it also establishes the patient's AD stage. The VGG19 pre-trained model has been improved, and it now identifes AD stages with an accuracy of 97%. a model which utilized transfer learning on multiclass categorization using brain MRIs, to classifying the pictures into four categories: very mild dementia (VMD), mild dementia (MD), and moderate dementia (MOD) non-dementia (ND). The correctness of the proposed system model is 91.70% according to simulation fndings. It was also noted that the suggested technique provides fndings that are more accurate when compared to earlier methods

FLOWCHART


Dataset Collection
(OASIS Dataset)
Preprocessing
Feature Selection
(Select k Best)
Feature Scaling
(Standard Scaler)
Handling Missing Data
(Mean values replace)
Data spliting
Training Data
(80%)
Testing Data
(20%)
Applying Machine
Learning Algorithm
GNB
XGBoost
DT
RF
GB
Voting
Evaluate the Result

RESULT

```
[9]  model.summary()

...  Model: "sequential"
     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     conv2d (Conv2D)              (None, 148, 148, 32)      896
     _________________________________________________________________
     conv2d_1 (Conv2D)            (None, 146, 146, 64)      18496
     _________________________________________________________________
     max_pooling2d (MaxPooling2D) (None, 73, 73, 64)        0
     _________________________________________________________________
     dropout (Dropout)            (None, 73, 73, 64)        0
     _________________________________________________________________
     conv2d_2 (Conv2D)            (None, 71, 71, 64)        36928
     _________________________________________________________________
     conv2d_3 (Conv2D)            (None, 69, 69, 64)        36928
     _________________________________________________________________
     dropout_1 (Dropout)          (None, 69, 69, 64)        0
     _________________________________________________________________
     max_pooling2d_1 (MaxPooling2 (None, 34, 34, 64)        0
     _________________________________________________________________
     dropout_2 (Dropout)          (None, 34, 34, 64)        0
     _________________________________________________________________
     conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856
     _________________________________________________________________
     conv2d_5 (Conv2D)            (None, 30, 30, 128)       147584
     ...
     Total params: 4,447,044
     Trainable params: 4,447,044
     Non-trainable params: 0
```

Code File Edit Selection View Go Run Terminal Window Help 66% Tue 4 Jul 1:41 PM

Search

alzheimer-s-prediction.ipynb

Users > jagger > Downloads > alzheimer-s-prediction.ipynb > # This Python 3 environment comes with many helpful analytics libraries installed

+ Code + Markdown | Outline ··· Select Kernel

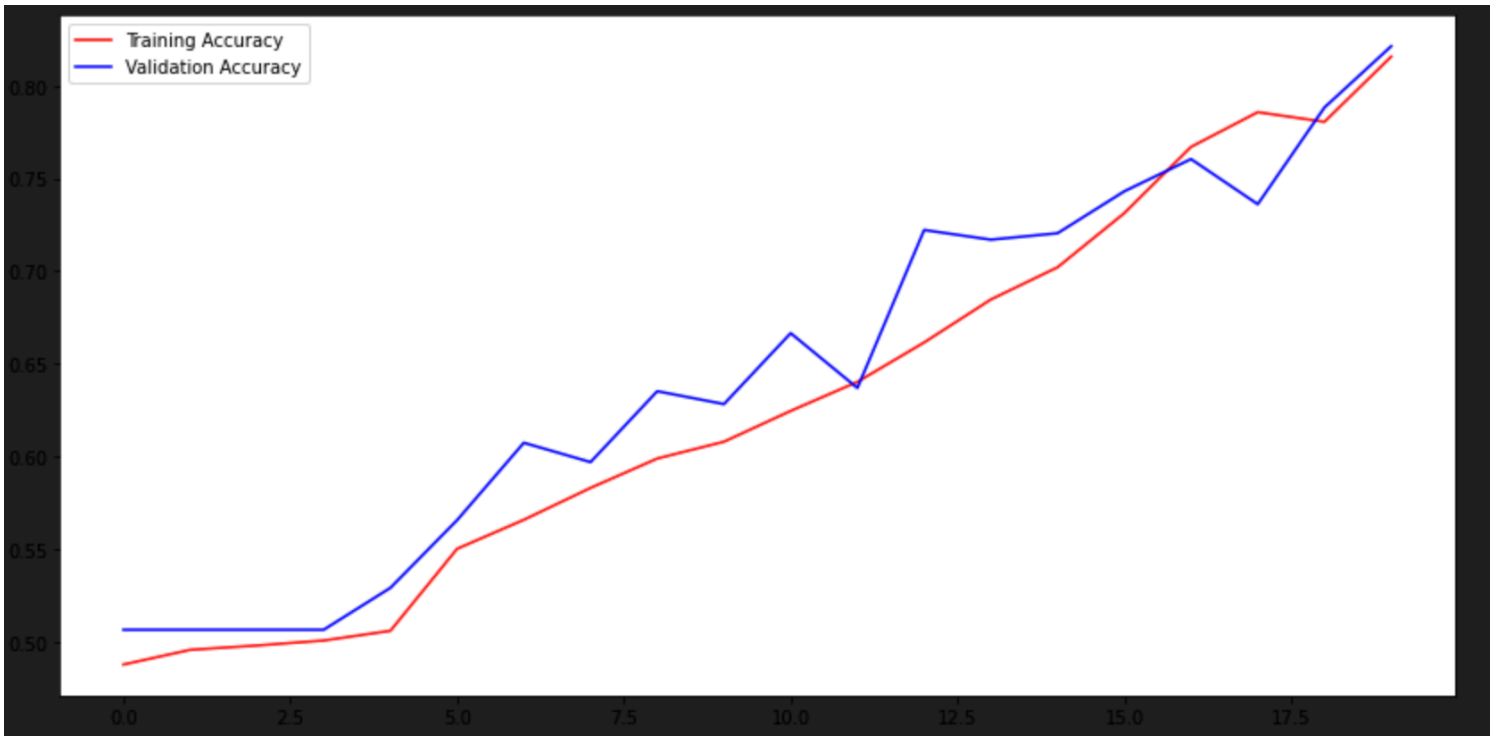
```
[11]  history = model.fit(X_train,y_train,epochs=20,validation_split=0.1)
```

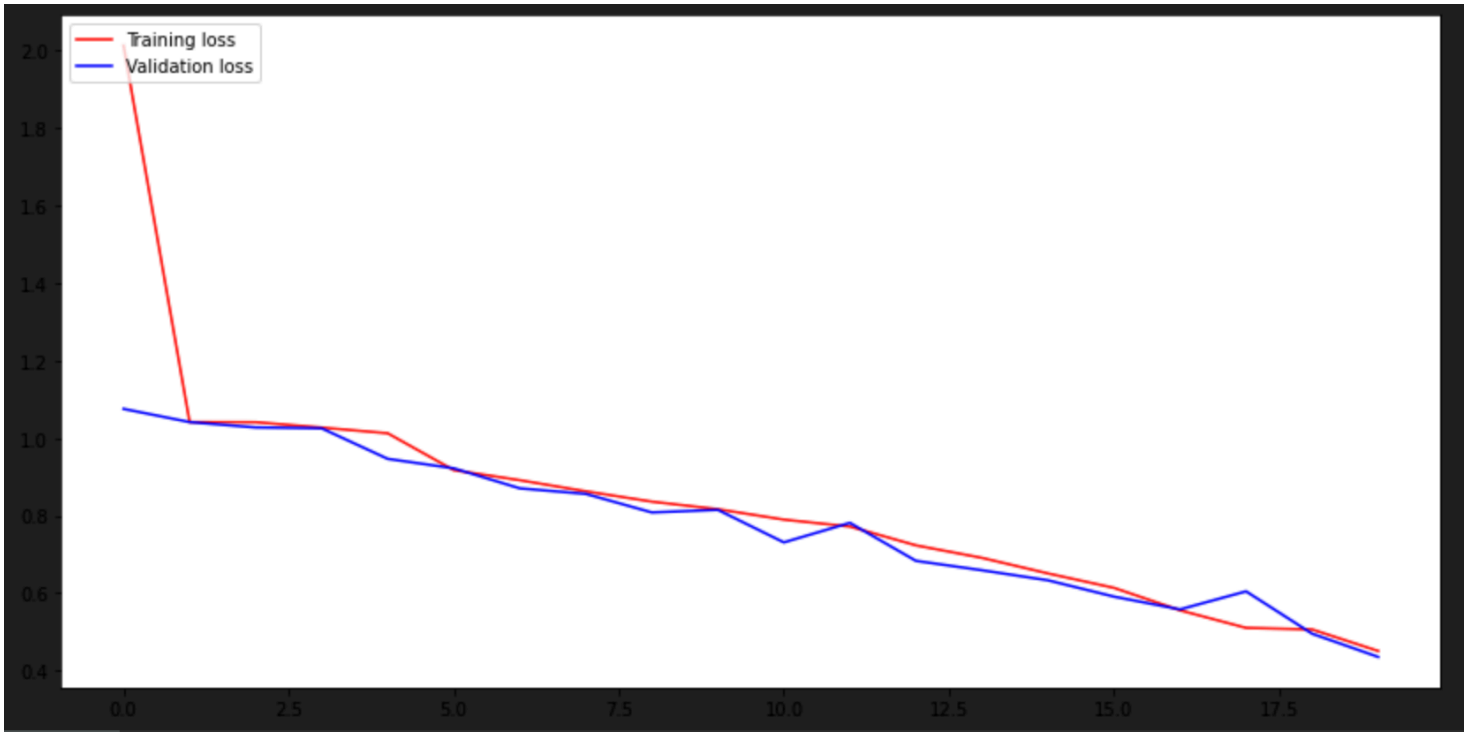
Python

```
Epoch 1/20
162/162 [==============================] - 416s 3s/step - loss: 2.0120 - accuracy: 0.4882 - val_loss: 1.0752 - val_accuracy: 0.5069
Epoch 2/20
162/162 [==============================] - 407s 3s/step - loss: 1.0413 - accuracy: 0.4961 - val_loss: 1.0412 - val_accuracy: 0.5069
Epoch 3/20
162/162 [==============================] - 400s 2s/step - loss: 1.0409 - accuracy: 0.4985 - val_loss: 1.0274 - val_accuracy: 0.5069
Epoch 4/20
162/162 [==============================] - 399s 2s/step - loss: 1.0270 - accuracy: 0.5012 - val_loss: 1.0251 - val_accuracy: 0.5069
Epoch 5/20
162/162 [==============================] - 403s 2s/step - loss: 1.0124 - accuracy: 0.5064 - val_loss: 0.9464 - val_accuracy: 0.5295
Epoch 6/20
162/162 [==============================] - 409s 3s/step - loss: 0.9173 - accuracy: 0.5505 - val_loss: 0.9220 - val_accuracy: 0.5660
Epoch 7/20
162/162 [==============================] - 405s 3s/step - loss: 0.8913 - accuracy: 0.5662 - val_loss: 0.8703 - val_accuracy: 0.6076
Epoch 8/20
162/162 [==============================] - 400s 2s/step - loss: 0.8627 - accuracy: 0.5833 - val_loss: 0.8561 - val_accuracy: 0.5972
Epoch 9/20
162/162 [==============================] - 398s 2s/step - loss: 0.8361 - accuracy: 0.5992 - val_loss: 0.8082 - val_accuracy: 0.6354
Epoch 10/20
162/162 [==============================] - 404s 2s/step - loss: 0.8165 - accuracy: 0.6082 - val_loss: 0.8149 - val_accuracy: 0.6285
Epoch 11/20
162/162 [==============================] - 413s 3s/step - loss: 0.7897 - accuracy: 0.6248 - val_loss: 0.7310 - val_accuracy: 0.6667
Epoch 12/20
162/162 [==============================] - 402s 2s/step - loss: 0.7717 - accuracy: 0.6404 - val_loss: 0.7813 - val_accuracy: 0.6372
Epoch 13/20
...
Epoch 19/20
162/162 [==============================] - 411s 3s/step - loss: 0.5064 - accuracy: 0.7805 - val_loss: 0.4954 - val_accuracy: 0.7882
Epoch 20/20
162/162 [==============================] - 406s 3s/step - loss: 0.4509 - accuracy: 0.8156 - val_loss: 0.4356 - val_accuracy: 0.8212
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Restricted Mode ⊗ 0 ⚠ 0






```
a=model.predict(img_array)
indices = a.argmax()
indices
```

[18]

```
1
```

ADVANTAGES

Xception achieves improved efficiency by replacing standard convolutions with depthwise separable convolutions. This separation of spatial and channel-wise filtering reduces the number of parameters and computations, resulting in faster training and inference times.
The depthwise separable convolutions in Xception act as a regularizer, reducing the risk of overfitting on small datasets. This property makes Xception particularly useful in scenarios with limited training data.
Xception can be used effectively for transfer learning. By pretraining the network on a large dataset (e.g., ImageNet), the learned features can be transferred to a different task with a smaller dataset, enabling faster convergence and potentially improving performance

DISADVANTAGES

Although Xception reduces the number of parameters compared to traditional convolutional networks, it introduces an additional computational cost due to the depthwise separable convolutions. This increased complexity can limit its usage on resource-constrained devices or in real-time applications.
Xception allows for more efficient modeling of spatial and channel-wise dependencies. However, it may struggle to capture long-range contextual information that could be beneficial for certain tasks, such as semantic segmentation or tasks requiring larger receptive fields.

APPLICATIONS

Xception, as a convolutional neural network (CNN) architecture, has been successfully applied to various computer vision tasks. Some of the notable applications of Xception include:

- Image Classification: Xception has been widely used for image classification tasks, where the goal is to assign a label or category to an input image. It has demonstrated strong performance on benchmark datasets such as ImageNet, achieving high accuracy rates in classifying images into predefined categories.
- Object Detection: Xception can be employed for object detection tasks, where the goal is to detect and localize objects of interest within an image. By combining Xception with additional layers, such as region proposal networks (RPN) or anchor-based methods like Faster R-CNN, it becomes possible to accurately identify objects and their locations in complex scenes.
- Semantic Segmentation: Xception has been used for semantic segmentation, which involves assigning a class label to each pixel in an image, thereby segmenting the image into meaningful regions. By

incorporating Xception into fully convolutional networks (FCNs) or similar architectures, it becomes feasible to generate pixel-level segmentation masks for various applications like medical imaging, autonomous driving, and scene understanding.

- Transfer Learning: Xception, pre-trained on large-scale datasets such as ImageNet, can be used as a feature extractor in transfer learning scenarios. By leveraging the learned features from Xception's earlier layers, it becomes possible to fine-tune the network on smaller, domain-specific datasets, achieving good performance with less training data.
- Image Enhancement and Restoration: Xception can be utilized for image enhancement and restoration tasks, such as denoising, deblurring, or super-resolution. By training Xception on paired input-output image examples, it can learn to restore or enhance image details, resulting in improved image quality.
- Biomedical Imaging: Xception has found applications in biomedical imaging tasks, such as histopathology analysis, medical image classification, and segmentation. Its ability to learn discriminative features from complex images makes it suitable for analyzing medical images and assisting in disease diagnosis and treatment planning.

CONCLUSION

Xception can be effectively used for transfer learning. By pretraining the network on a large dataset, such as ImageNet, and fine-tuning it on Alzheimer's disease-specific data, the learned features can be transferred. Although Alzheimer's disease prediction involves complex data and multiple modalities, Xception's ability to capture relevant features can be beneficial. Xception's architectural design can be extended to handle multiple modalities, allowing for the integration of diverse data sources. The complexity of the model makes it difficult to understand the learned features and decision-making process. Ensuring interpretability and explainability is an important consideration in Alzheimer's disease prediction.

BIBLIOGRAPHY

1. https://www.connectedpapers.com/main/4f28ec72db0792948f85f6cd294ec9594796af37/Early%20Stage-Alzheimer's-Disease-Prediction-Using-Machine-Learning-Models/graph

2. https://link.springer.com/article/10.1007/s11831-022-09870-0

3. https://link.springer.com/content/pdf/10.1007/s44174-023-00078-9.pdf

4. https://www.frontiersin.org/articles/10.3389/fpubh.2022.853294/full

APPENDIX
SOURCE CODE:

LOAD LIBRARIES

```
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout,
Flatten, Dense, concatenate
from tensorflow.keras.models import Sequential
from tensorflow.keras import Input
from tensorflow.keras.layers import Input, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import SeparableConv2D,
BatchNormalization, GlobalAveragePooling2D
```

LOAD TRAIN DATA

```
train_df = tf.keras.preprocessing.image_dataset_from_directory('/Users/
jagger/Downloads/Externship/Dataset/train',
                                                labels = 'inferred',
                                                label_mode = 'categorical',
                                                class_names = None,
                                                color_mode = 'grayscale',
                                                batch_size = 32,
                                                image_size = (224,224),
                                                shuffle = True,
                                                seed = 42,
                                                validation_split = 0.2,
                                                subset = 'training',
                                                interpolation = 'bilinear',
                                                follow_links = False
                                               )
print(train_df)
train_df.class_names
```

LOAD TEST DATA

```
test_df = tf.keras.preprocessing.image_dataset_from_directory('/Users/
jagger/Downloads/Externship/Dataset/test',
                                                labels="inferred",
                                                label_mode="categorical",
                                                class_names=None,
                                                color_mode="grayscale",
                                                batch_size=32,
                                                image_size=(224,224),
                                                shuffle=True,
                                                seed=None,
                                                validation_split=None,
                                                subset=None,
                                                interpolation="bilinear",
                                                follow_links=False)
test_df.class_names
train_df.class_names
```

BUILD THE MODEL

```
input_shape = (224,224,1)

input_layer = Input(shape=input_shape)
rgb_input = Concatenate()([input_layer, input_layer, input_layer])

xcep_model =Xception(input_tensor=rgb_input, weights='imagenet',
include_top=False)
for layer in xcep_model.layers:
    layer.trainable = False
cnn = Sequential([
    xcep_model,
    Dropout(0.5),
    GlobalAveragePooling2D(),
    Flatten(),
    BatchNormalization(),
    Dense(512, activation ='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(256, activation ='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(128, activation ='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(64, activation ='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(4,activation = 'softmax')
],name = ‘Xception_model')

cnn.compile(loss='categorical_crossentropy',
            optimizer = 'adam',
            metrics = [‘accuracy'])
history = cnn.fit(train_df,epochs=30)
```

PREDICTION

```
image_path = '/Users/jagger/Downloads/Externship/Dataset/test/
ModerateDemented/28.jpg'
image = tf.keras.preprocessing.image.load_img(image_path,
target_size=(224, 224), color_mode='grayscale')
image_array = tf.keras.preprocessing.image.img_to_array(image)
image_array = np.expand_dims(image_array, axis=0)

image_array = image_array / 255.0
predictions = cnn.predict(image_array)

class_names = train_df.class_names
predicted_class_index = np.argmax(predictions)
predicted_class = class_names[predicted_class_index]
```