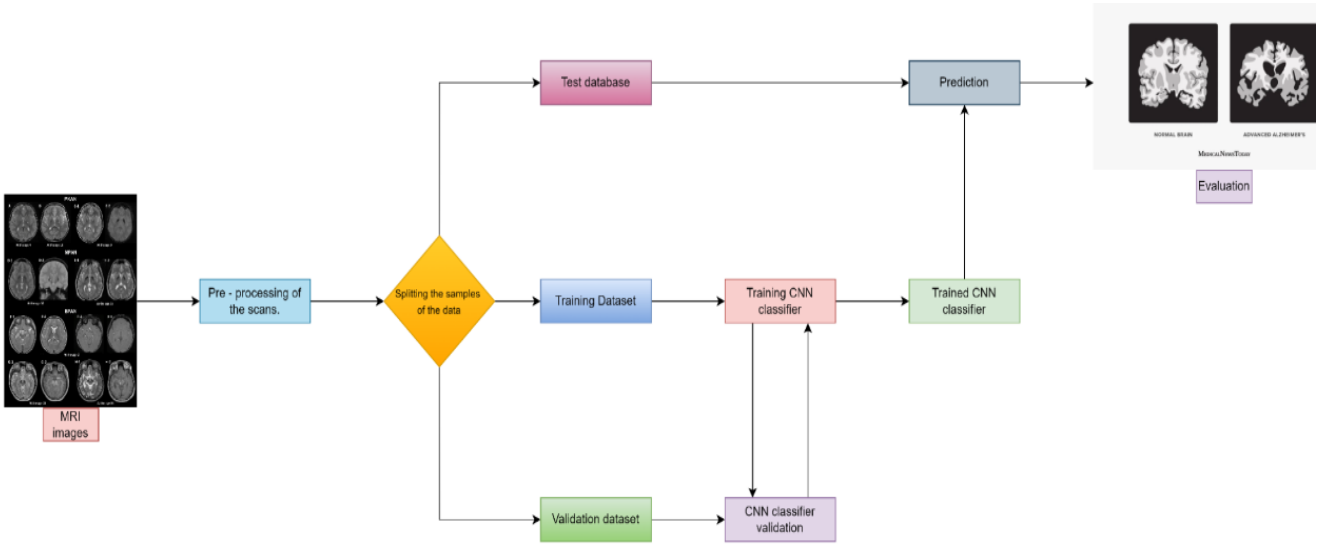
Cognitive Care: Early Intervention for Alzheimer's Disease

Project Description –

Alzheimer's disease (AD) is a progressive and irreversible neurological disorder that affects the brain, leading to memory loss, cognitive impairment, and changes in behavior and personality. It is the most common cause of dementia among older adults and is characterized by the buildup of abnormal protein deposits in the brain, including amyloid plaques and tau tangles.

The exact cause of Alzheimer's disease is not yet fully understood, but it is believed to be influenced by a combination of genetic, environmental, and lifestyle factors. Age is also a significant risk factor, with the risk of developing the disease increasing significantly after the age of 65. The early symptoms of Alzheimer's disease may include mild memory loss, difficulty with problem-solving, and changes in mood or behavior. As the disease progresses, these symptoms become more severe, with individuals experiencing significant memory loss, difficulty communicating, and a loss of the ability to perform daily activities.

By using deep learning models like Xception to analyze medical imaging data, it may be able to identify early signs of Alzheimer's disease before symptoms become severe. This can help healthcare providers to provide early treatment and support for patients and their families, ultimately leading to better outcomes for all involved.



Project Flow –

- The user interacts with the UI to choose an image.
- The chosen image is processed by a Xception deep learning model.
- The Xception model is integrated with a Flask application.
- The Xception model analyzes the image and generates predictions.
- The predictions are displayed on the Flask UI for the user to see.
- This process enables users to input an image and receive accurate predictions quickly.

Data Collection –

1) Dataset Download – https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

2)Creating a testing and training path

```
In [9]: from tensorflow.keras.layers import Dense, Flatten, Input, Dropout
        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
        import numpy as np
```

```
In [10]: train_path = "C://Users//Chetan 696//Desktop//Alzheimers//Combined Dataset//train"
         test_path = "C://Users//Chetan 696//Desktop//Alzheimers//Combined Dataset//test"
```

3)Image Pre-processing.

- Import the required library
- Configure ImageDataGenerator class
- Handling imbalance data
- Splitting into train-test split

```
In [11]: from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
         IMG_SIZE=180
         IMAGE_SIZE=[180, 180]
         DIM=(IMG_SIZE, IMG_SIZE)
         ZOOM=[.99, 1.01]
         BRIGHT_RANGE=[0.8, 1.2]
         HORZ_FLIP=True
         FILL_MODE="constant"
         DATA_FORMAT="channels_last"
         WORK_DIR="C://Users//Chetan 696//Desktop//Alzheimers//Combined Dataset//train"
         work_dr = IDG(rescale=1./255, brightness_range=BRIGHT_RANGE, zoom_range=ZOOM, data_format=DATA_FORMAT, fill_mode=FILL_MODE, hori
         train_data_gen = work_dr.flow_from_directory(directory=WORK_DIR, target_size=DIM, batch_size=6500, shuffle=False)

         Found 10240 images belonging to 4 classes.
```

4)Model Building

- Pre-trained CNN model as a Feature Extractor
- Creating Sequential layers
- Configure the Learning Process
- Train the model
- Save the Model
- Test the model

In [12]:
```
train_data,train_labels=train_data_gen.next()
```

In [13]:
```
print(train_data.shape,train_labels.shape)
```

```
(6500, 180, 180, 3) (6500, 4)
```

In [14]:
```
from sklearn.model_selection import train_test_split

# Splitting the data into train, test, and validation sets
train_data, test_data, train_labels, test_labels = train_test_split(train_data, train_labels, test_size=0.2, random_state=42)
train_data, val_data, train_labels, val_labels = train_test_split(train_data, train_labels, test_size=0.2, random_state=42)
```

In [15]:
```
from keras.applications import Xception

IMAGE_SIZE = [180, 180]
input_shape = tuple(IMAGE_SIZE + [3])

xception_model = Xception(input_shape=input_shape, include_top=False, weights='imagenet')
```

In [16]:
```
for layer in xception_model.layers:
    layer.trainable = False
```

In [17]:
```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D

custom_inception_model = Sequential([
    xception_model,
    Dropout(0.5),
    GlobalAveragePooling2D(),
    Flatten(),
    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="inception_cnn_model")
```

In [18]:
```
import tensorflow as tf
METRICS = [
    tf.keras.metrics.CategoricalAccuracy(name='acc'),
    tf.keras.metrics.AUC(name='auc')
]

custom_inception_model.compile(optimizer='rmsprop', loss=tf.losses.CategoricalCrossentropy(), metrics=METRICS)
```

Total Epochs = 80

50 Epoch –

```
history=custom_inception_model.fit(train_data,train_labels,validation_data=(val_data,val_labels), epochs=50)
```

```
Epoch 1/50
84/84 [==============================] - 87s 999ms/step - loss: 1.1858 - acc: 0.5199 - auc: 0.7643 - val_loss: 0.5679 - val_ac
c: 0.7988 - val_auc: 0.9552
...
Epoch 50/50
84/84 [==============================] - 84s 1s/step - loss: 0.1723 - acc: 0.9425 - auc: 0.9933 - val_loss: 0.1581 - val_acc:
0.9429 - val_auc: 0.9940
```

Val_Acc = 99.4%

30 Epoch –

```
history=custom_inception_model.fit(train_data,train_labels,validation_data=(val_data,val_labels), epochs=30)
```

```
Epoch 1/30
130/130 [==============================] - 149s 1s/step - loss: 1.0240 - acc: 0.6002 - auc: 0.8284 - val_loss: 0.5805 - val_ac
c: 0.7923 - val_auc: 0.9429

Epoch 30/30
130/130 [==============================] - 241s 2s/step - loss: 0.1967 - acc: 0.9240 - auc: 0.9918 - val_loss: 0.1481 - val_ac
c: 0.9365 - val_auc: 0.9958
```

Final Val_Acc = 99.58%

Saving The Model –

```
custom_inception_model.save("Xception_model.h5")
```

```
C:\Users\Chetan 696\anaconda3\Lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an
HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `
model.save('my_model.keras')`.
  saving_api.save_model(
```

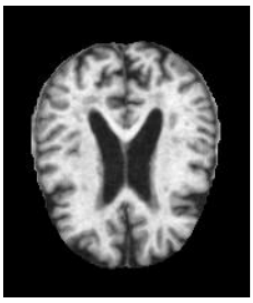
Testing The Model –

```
import numpy as np
from keras.preprocessing import image
from keras.applications.xception import Xception, preprocess_input, decode_predictions
import numpy as np
from keras.preprocessing import image
from keras.models import load_model
```

```
model_path = "C://Users//Chetan 696//Desktop//Alzheimers//Xception_model_image.h5"
model = load_model(model_path)
```

Mild Demented –

```
from IPython.display import display
from PIL import Image

# Use the file uploader to select an image
uploaded_image_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//MildDemented//28.jpg"
img = Image.open(uploaded_image_path)
display(img)
```

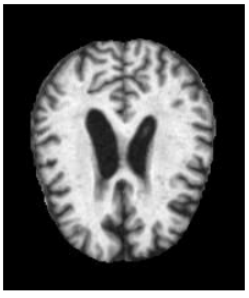


```
from PIL import Image
from keras.preprocessing import image
from keras.applications.inception_v3 import preprocess_input
import numpy as np

# Load the image
img_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//MildDemented//28.jpg"
img = Image.open(img_path)

# Convert the image to RGB if it's grayscale
img = img.convert('RGB')

# Resize the image to match the model's input shape (180, 180)
img = img.resize((180, 180))

# Convert the image to array and preprocess it
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Now you can use img_data with your model
output = np.argmax(model.predict(img_data), axis=1)

class_labels = ["MildDemented", "ModerateDemented", "NonDemented", "VeryMildDemented"]
predicted_class = class_labels[output[0]]

print("Predicted class:", predicted_class)
```

```
1/1 [==============================] - 0s 63ms/step
Predicted class: MildDemented
```

Moderate Demented –

```
from IPython.display import display
from PIL import Image

# Use the file uploader to select an image
uploaded_image_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//ModerateDemented//28.jpg"
img = Image.open(uploaded_image_path)
display(img)
```

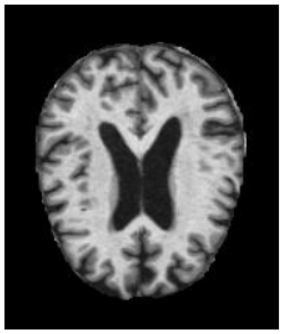


```
from PIL import Image
from keras.preprocessing import image
from keras.applications.inception_v3 import preprocess_input
import numpy as np

# Load the image
img_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//ModerateDemented//28.jpg"
img = Image.open(img_path)

# Convert the image to RGB if it's grayscale
img = img.convert('RGB')

# Resize the image to match the model's input shape (180, 180)
img = img.resize((180, 180))

# Convert the image to array and preprocess it
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Now you can use img_data with your model
output = np.argmax(model.predict(img_data), axis=1)

class_labels = ["MildDemented", "ModerateDemented", "NonDemented", "VeryMildDemented"]
predicted_class = class_labels[output[0]]

print("Predicted class:", predicted_class)
```

```
1/1 [==============================] - 0s 64ms/step
Predicted class: ModerateDemented
```

Non-Demented –

```
from IPython.display import display
from PIL import Image

# Use the file uploader to select an image
uploaded_image_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//NonDemented//26 (75).jpg"
img = Image.open(uploaded_image_path)
display(img)
```

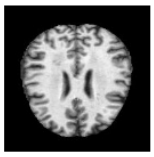


```
from PIL import Image
from keras.preprocessing import image
from keras.applications.inception_v3 import preprocess_input
import numpy as np

# Load the image
img_path = "C://Users//Chetan 696//Desktop//Alzheimers//Alzheimer_s Dataset//test//NonDemented//26 (75).jpg"
img = Image.open(img_path)

# Convert the image to RGB if it's grayscale
img = img.convert('RGB')

# Resize the image to match the model's input shape (180, 180)
img = img.resize((180, 180))

# Convert the image to array and preprocess it
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Now you can use img_data with your model
output = np.argmax(model.predict(img_data), axis=1)

class_labels = ["MildDemented", "ModerateDemented", "NonDemented", "VeryMildDemented"]
predicted_class = class_labels[output[0]]

print("Predicted class:", predicted_class)
```

```
1/1 [==============================] - 0s 60ms/step
Predicted class: NonDemented
```

Very Mild Demented –

```
from IPython.display import display
from PIL import Image

# Use the file uploader to select an image
uploaded_image_path = "C://Users//Chetan 696//Desktop//Alzheimers//Combined Dataset//test//Very Mild Impairment//30 (12).jpg"
img = Image.open(uploaded_image_path)
display(img)
```



```
from PIL import Image
from keras.preprocessing import image
from keras.applications.inception_v3 import preprocess_input
import numpy as np

# Load the image
img_path = "C://Users//Chetan 696//Desktop//Alzheimers//Combined Dataset//test//Very Mild Impairment//30 (12).jpg"
img = Image.open(img_path)

# Convert the image to RGB if it's grayscale
img = img.convert('RGB')

# Resize the image to match the model's input shape (180, 180)
img = img.resize((180, 180))

# Convert the image to array and preprocess it
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)

# Now you can use img_data with your model
output = np.argmax(model.predict(img_data), axis=1)

class_labels = ["VeryMildDemented", "ModerateDemented", "NonDemented", "MildDemented"]
predicted_class = class_labels[output[0]]

print("Predicted class:", predicted_class)
```

```
1/1 [==============================] - 1s 686ms/step
Predicted class: VeryMildDemented
```