# Asia University Winter Program

Final Report :-
Muskan Didwania
P2-0047
SRM IST (India)

# Project Title:

# Alzheimer Detection and Classification

# Summary of the Alzheimer's Disease:

Alzheimer's disease (AD) is a progressive neurodegenerative disease. Though best known for its role in declining memory function, symptoms also include: difficulty thinking and reasoning, making judgements and decisions, and planning and performing familiar tasks. It may also cause alterations in personality and behavior. The cause of AD is not well understood. There is thought to be a significant hereditary component. For example, a variation of the APOE gene, APOE e4, increases risk of Alzheimer's disease. Pathologically, AD is associated with amyloid beta plaques and neurofibrillary tangles.

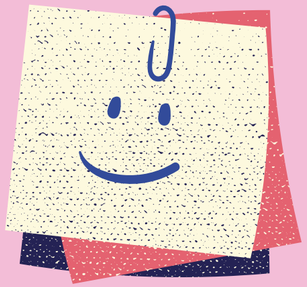
# Diagnosis

Onset of the disease is slow and early symptoms are often dismissed as normal signs of aging. A diagnosis is typically given based on history of illness, cognitive tests, medical imaging, and blood tests.

# Treatment:

There is no medication that stops or reverses the progression of AD. There are two types of drugs that attempt to treat the cognitive symptoms:Acetylcholinesterase Inhibitors that work to prevent the breakdown of acetylcholine, a neurotransmitter critical in memory and cognition.Memantine (Namenda), which works to inhibit NMDA receptors in the brain.These medications can slightly slow down the progression of the disease.

# Prevention:



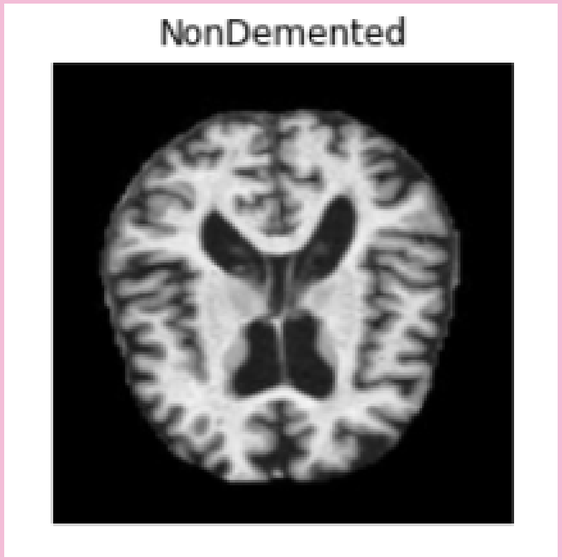
It is thought that frequent mental and physical exercise may reduce risk.

# Project Motivation:

The Alzheimer's Association estimates nearly 6 million Americans suffer from the disease and it is the 6th leading cause of death in the US. The estimated cost of AD was $277 billion in the US in 2018. The association estimates that early and accurate diagnoses could save up to $7.9 trillion in medical and care costs over the next few decades.

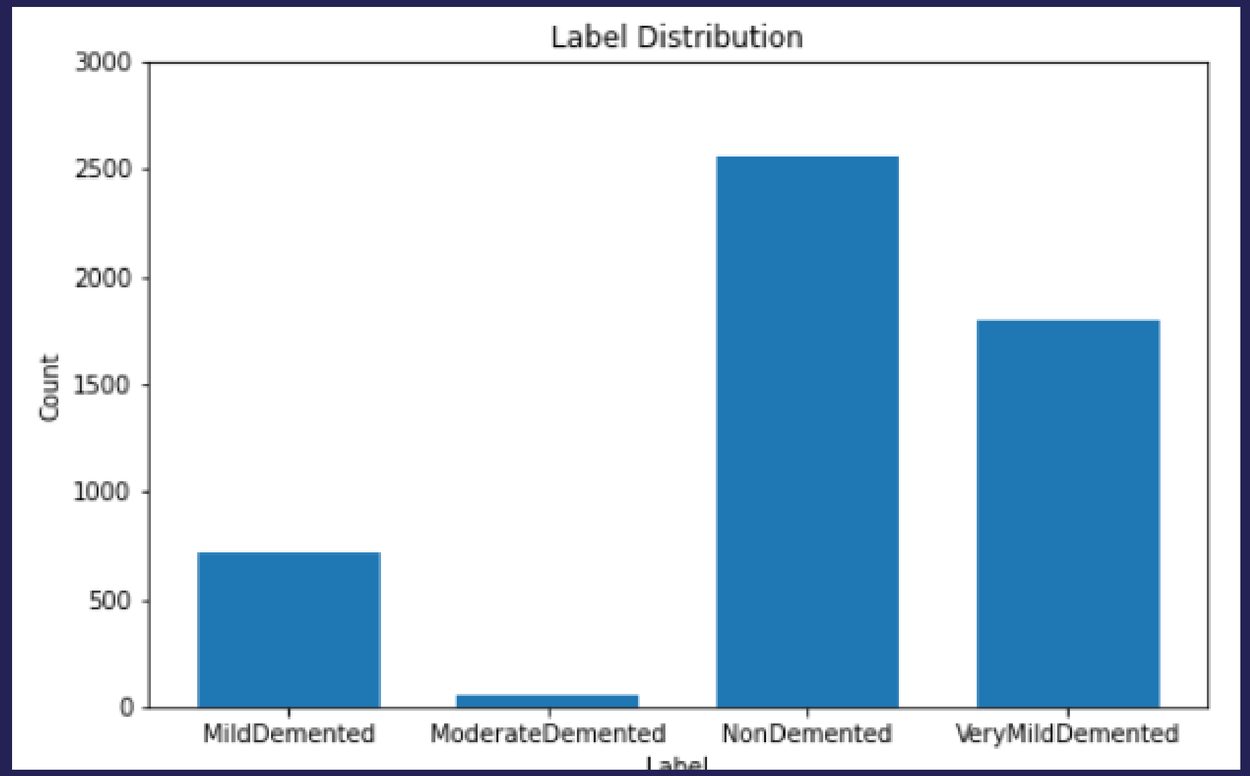
# The Dataset:

Source:Kaggle
Link:
https://www.kaggle.com/tourist55/alzheimers-dataset-4-class-of-images






# The Dataset:




**Data Explorer**

33.1 MB

- Alzheimer_s Dataset
  - test
    - MildDemented
    - ModerateDemented
    - NonDemented
    - VeryMildDemented
  - train
    - MildDemented
    - ModerateDemented
    - NonDemented
    - VeryMildDemented

# Project pipeline:

1. Download the dataset from Kaggle
2. Convert it to Tensorflow dataset format
3. Make Train Validation split from the train dataset
4. Visualize the dataset
5. One-Hot Encoding of the Labels/Classes
6. Build the VGG16 Model
7. Train
8. Evaluate the model
9. Test
10. Deploy

# Let's see the code:

- Python code implemented on Colab Notebook
- Model Used:VGG16
- Preprocessing on raw data:Imagees converted to (224,224,3)
- Dataset Annotation:Keras(Folder Format)
- Features Extracted:"HIPPOCAMPUS" region of the MRI scan

```
from google.colab import files
uploaded = files.upload()
!ls -lha kaggle.json
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle
!chmod 600 ~/.kaggle/kaggle.json
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Choose Files kaggle.json
• kaggle.json(application/json) - 70 bytes, last modified: 11/2/2020 - 100% done
Saving kaggle.json to kaggle.json
-rw-r--r-- 1 root root 70 Jan 31 08:28 kaggle.json
Collecting kaggle
  Downloading https://files.pythonhosted.org/packages/99/33/365c0d13f07a2a54744d027fe20b60dacdfdfb33bc04746db6ad0b79340b/kaggle-1.5.10.tar.gz (59kB)
     |████████████████████████████████| 61kB 8.7MB/s
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.10-cp36-none-any.whl size=73269 sha256=a472c7492ce1fdac467be9669c8ae66e57d2f8b9413ecc32fff417a3da1687b1
  Stored in directory: /root/.cache/pip/wheels/3a/d1/7e/6ce09b72b770149802c653a02783821629146983ee5a360f10
Successfully built kaggle
Installing collected packages: kaggle
  Found existing installation: kaggle 1.5.10
    Uninstalling kaggle-1.5.10:
      Successfully uninstalled kaggle-1.5.10
Successfully installed kaggle-1.5.10
```

```
[2] !kaggle datasets download -d tourist55/alzheimers-dataset-4-class-of-images
```

```
Downloading alzheimers-dataset-4-class-of-images.zip to /content
 59% 20.0M/34.1M [00:00<00:00, 98.9MB/s]
100% 34.1M/34.1M [00:00<00:00, 151MB/s]
```

```
[3] !unzip alzheimers-dataset-4-class-of-images.zip
```

# Downloading dataset from Kaggle & Unzipping it

# Preprocessing raw data

Converting raw data to a "Tensorflow Object "format.
Defining the batch size,image size and epochs.

```
[21] AUTOTUNE = tf.data.experimental.AUTOTUNE
     BATCH_SIZE = 16 * strategy.num_replicas_in_sync
     IMAGE_SIZE = [224, 224]
     EPOCHS = 5
     print(BATCH_SIZE)
```

```
16
```

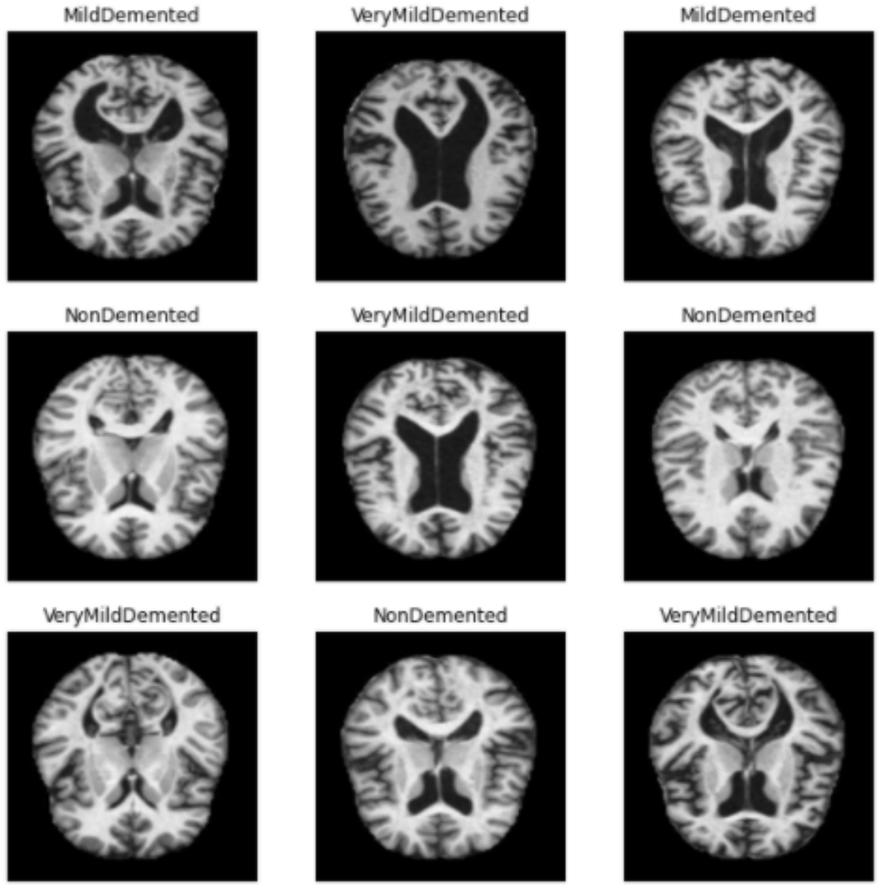
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "Alzheimer_s Dataset/train",
    labels='inferred',
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "Alzheimer_s Dataset/train",
    labels='inferred',
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
```

```
Found 5121 files belonging to 4 classes.
Using 4097 files for training.
Found 5121 files belonging to 4 classes.
Using 1024 files for validation.
```

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(train_ds.class_names[labels[i]])
    plt.axis("off")
    print(images[i].numpy().shape)
```

```
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
```




# Visualize the datset

We visualize the train dataset for better understanding.

Associate labels to classnames.

```
[26] NUM_CLASSES = len(class_names)
     class_names = train_ds.class_names
     print(class_names)
     NUM_CLASSES = len(class_names)
```

```
['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
```

```
[28] def one_hot_label(image, label):
         label = tf.one_hot(label, NUM_CLASSES)
         return image, label

     train_ds = train_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
     val_ds = val_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
```

```
[29] train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
     val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

```
[30] NUM_IMAGES = []

     for label in class_names:
         dir_name = "Alzheimer_s Dataset/train/" + label[:-2] + 'ed'
         NUM_IMAGES.append(len([name for name in os.listdir(dir_name)]))
```

```
[31] NUM_IMAGES

     [717, 52, 2560, 1792]
```

## One-Hot-Encoding

Since the labels are categorical string type we convert them to One Hot Format.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
output (Dense)               (None, 4)                 16388
=================================================================
Total params: 134,276,932
Trainable params: 134,276,932
Non-trainable params: 0
_________________________________________________________________
```

# Building our Custom VGG Model

The model layers are visualized using model.summary().

Categorical Crossentropy is used for the loss metric.

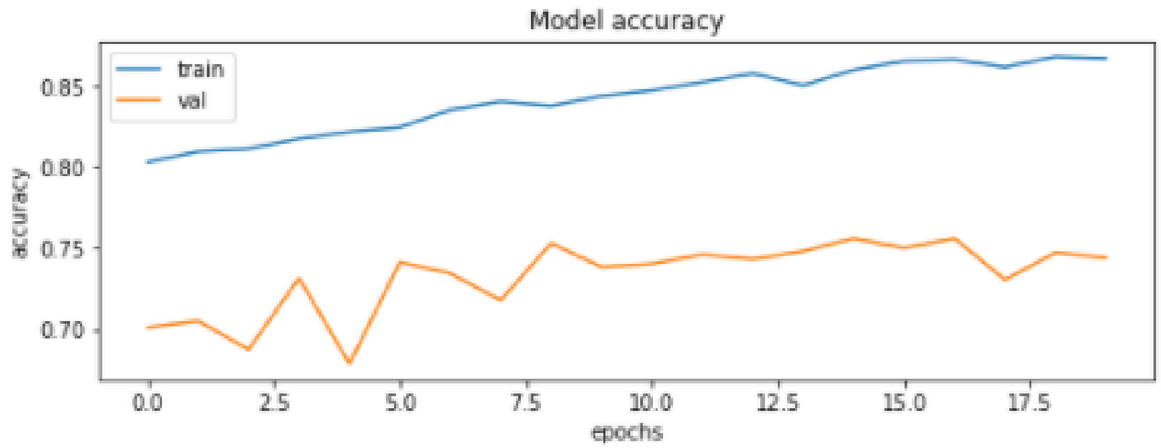
```
[36] hist=custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
```

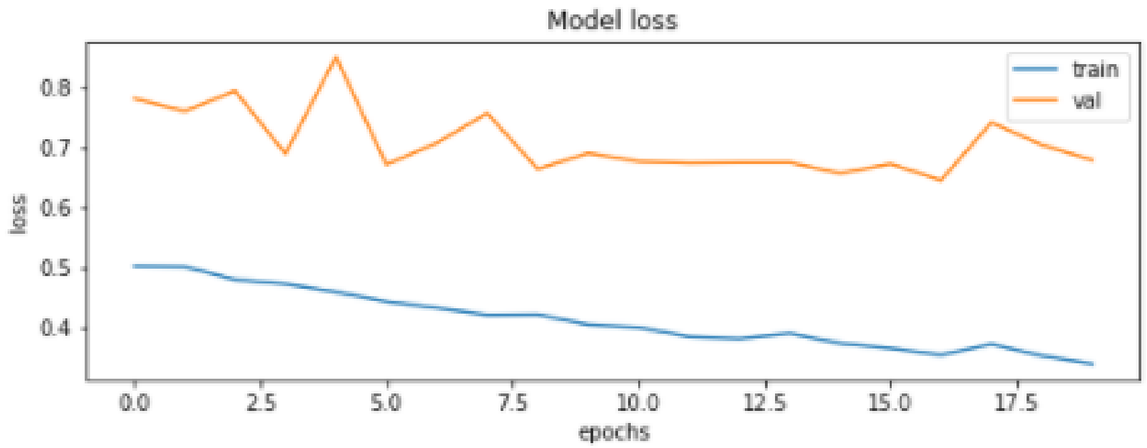
```
[82] hist = custom_vgg_model.fit(train_ds, batch_size=32, epochs=20, verbose=1,validation_data=val_ds)
```

# Plotting the train and validation accuracy-loss curves

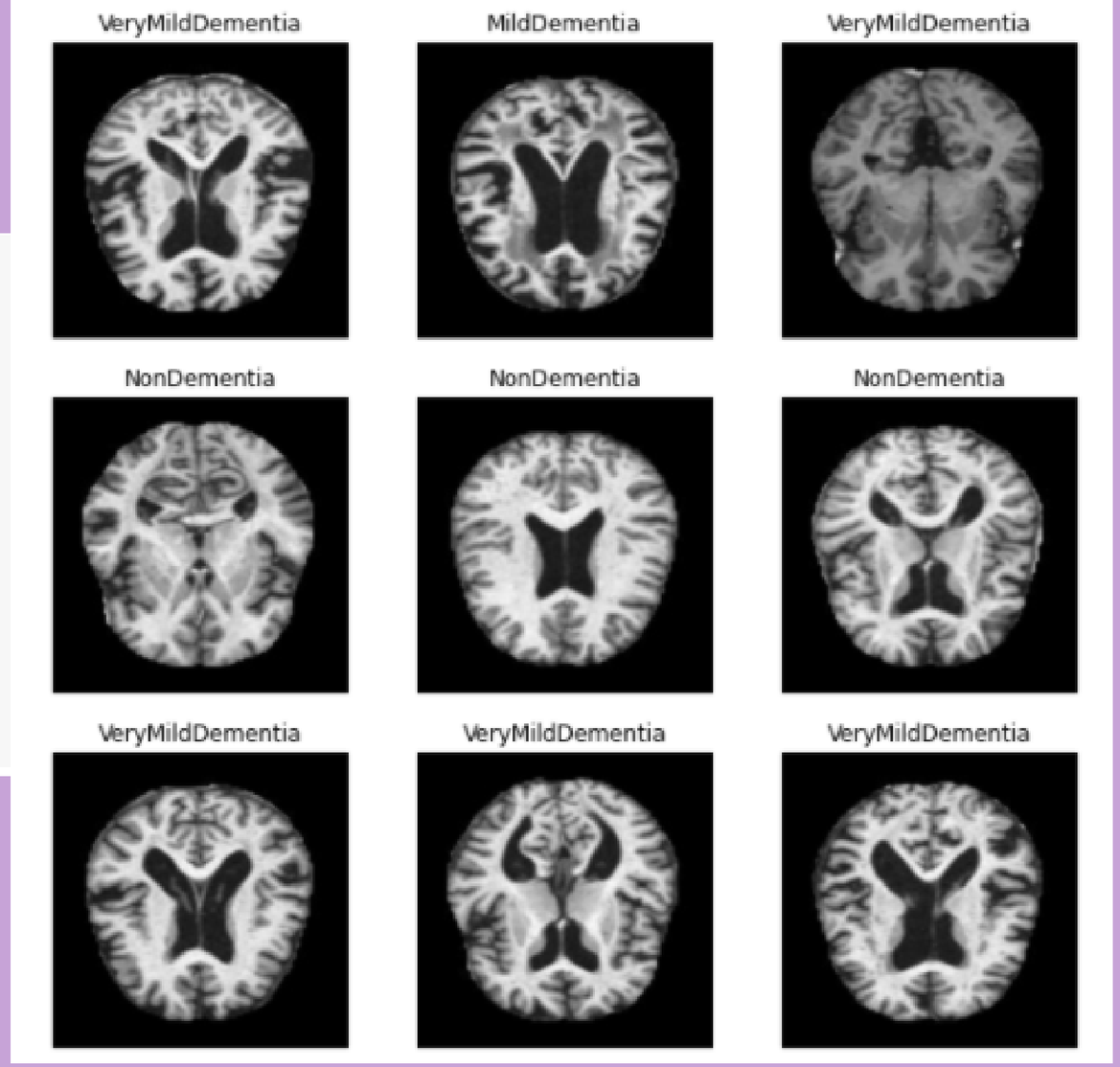
```
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()

for i, met in enumerate(['accuracy', 'loss']):
    ax[i].plot(hist.history[met])
    ax[i].plot(hist.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])
```



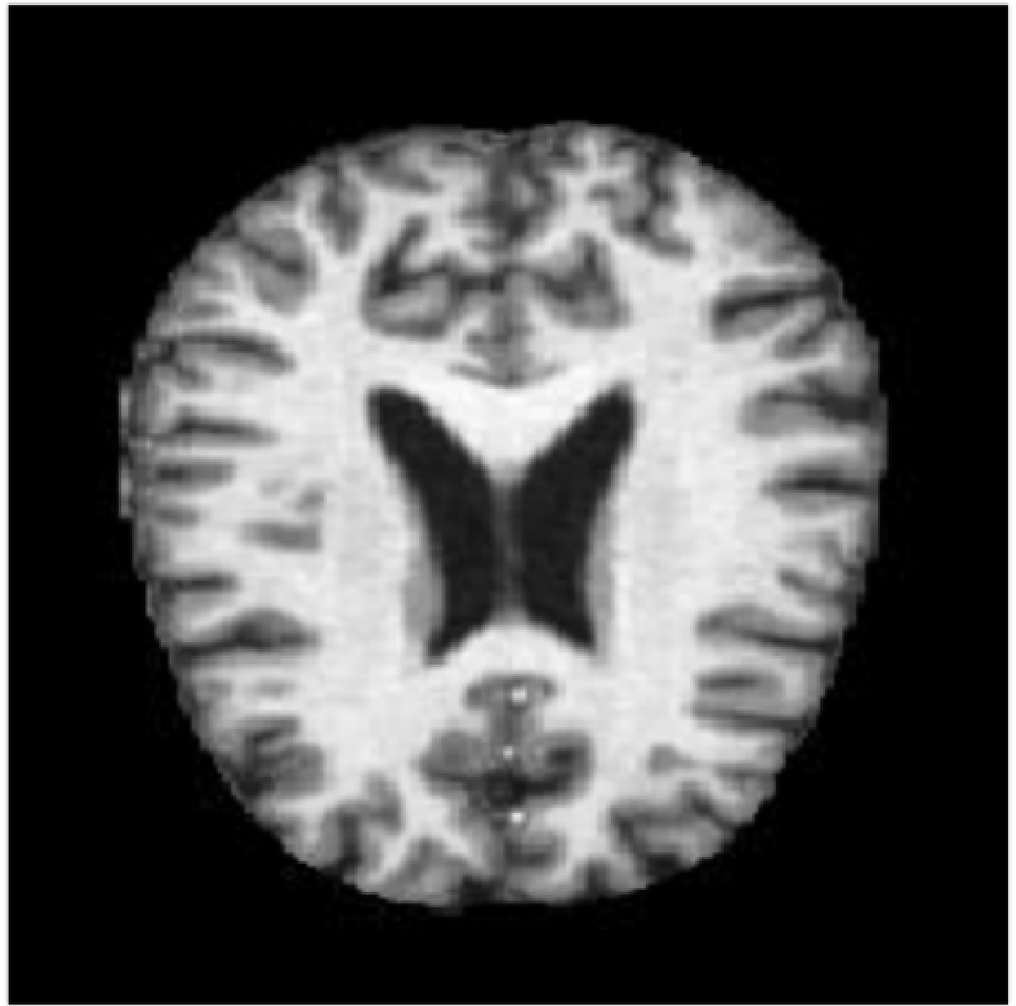
```
plt.figure(figsize=(10, 10))
for images, labels in val_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(decoder(np.argmax(y_predicts[i])))
    plt.axis("off")
    print(images[i].numpy().shape)
```



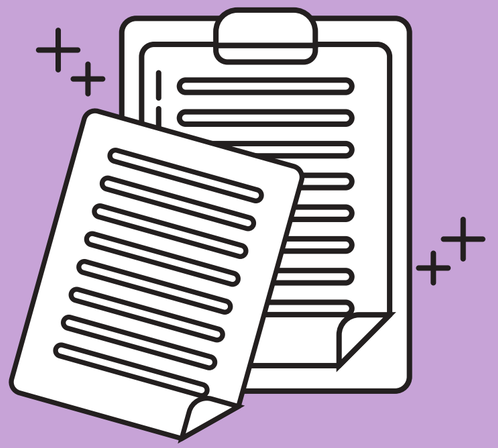

# Testing on Validation dataset

```
images = image.load_img("Alzheimer_s Dataset/test/NonDemented/26 (100).jpg", target_size=( 224, 224))
x = image.img_to_array(images)
x = x.reshape(1,x.shape[0],x.shape[1],x.shape[2])
prediction=custom_vgg_model.predict(x)
print(decoder(np.argmax(prediction)))
plt.figure(figsize=(10, 10))
plt.imshow(x[0].astype("uint8"))
plt.axis("off")
```

```
NonDementia
(-0.5, 223.5, 223.5, -0.5)
```



# Testing on images from local drive



# Results

The model gives an accuracy of 0.744% on validation dataset and 0.898% on train dataset.

```
[92] (loss, accuracy) = custom_vgg_model.evaluate(val_ds, batch_size=100, verbose=10)
```

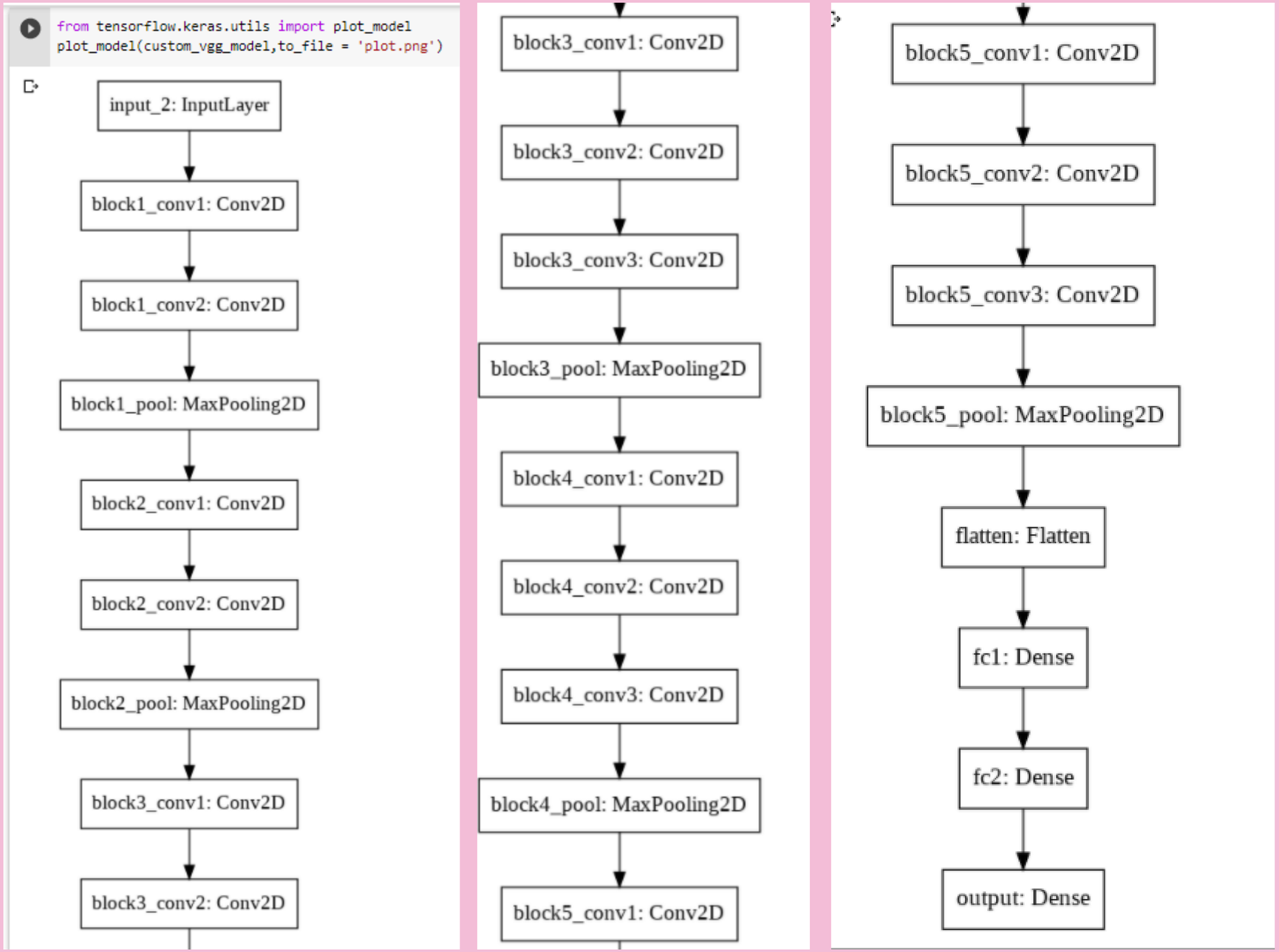
```
[93] loss
```

```
0.678227961063385
```

```
[90] accuracy
```

```
0.744140625
```

```
from tensorflow.keras.utils import plot_model
plot_model(custom_vgg_model,to_file = 'plot.png')
```




# Model Overview

# References:

1-https://www.nature.com/articles/s41598-019-54548-6
2-https://www.helpguide.org/harvard/recognizing-and-diagnosing-alzheimers.htm
3-
https://www.tensorflow.org/tutorials/distribute/custom_training


# Thank You AU!