Alzheimer's Disease Diagnosis with Deep Learning

Machine Learning Project

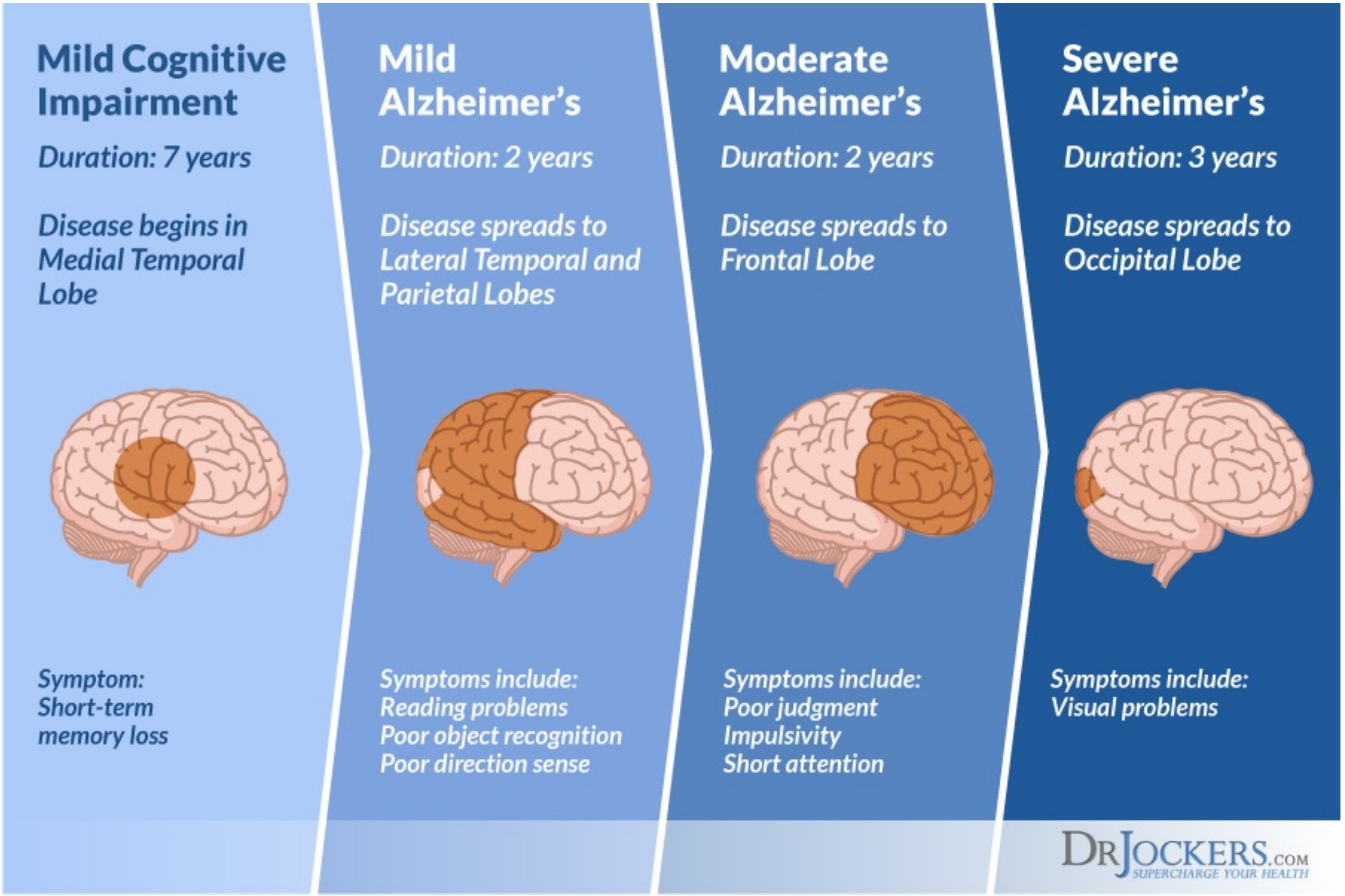
The Bridge Data Science Bootcamp

Sara Picó

09.04.2024

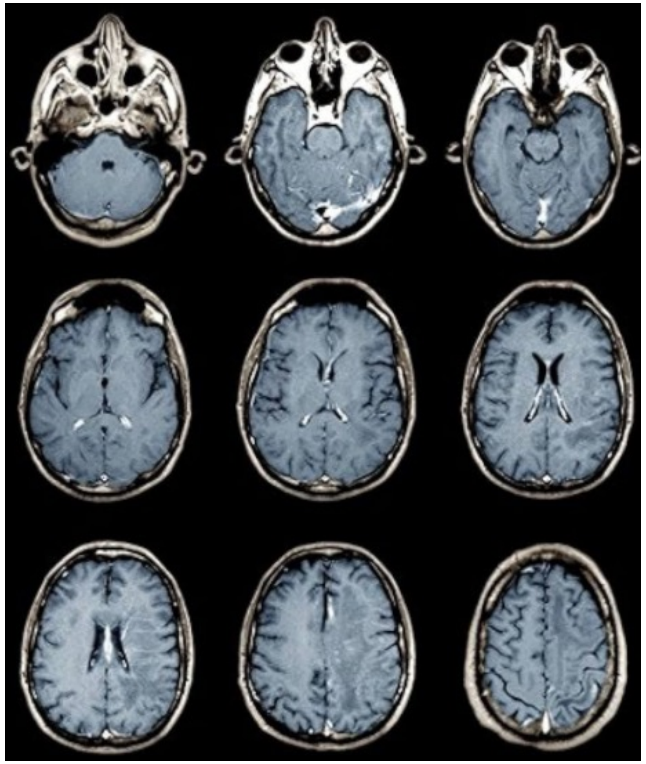
Problem statement

Different stages



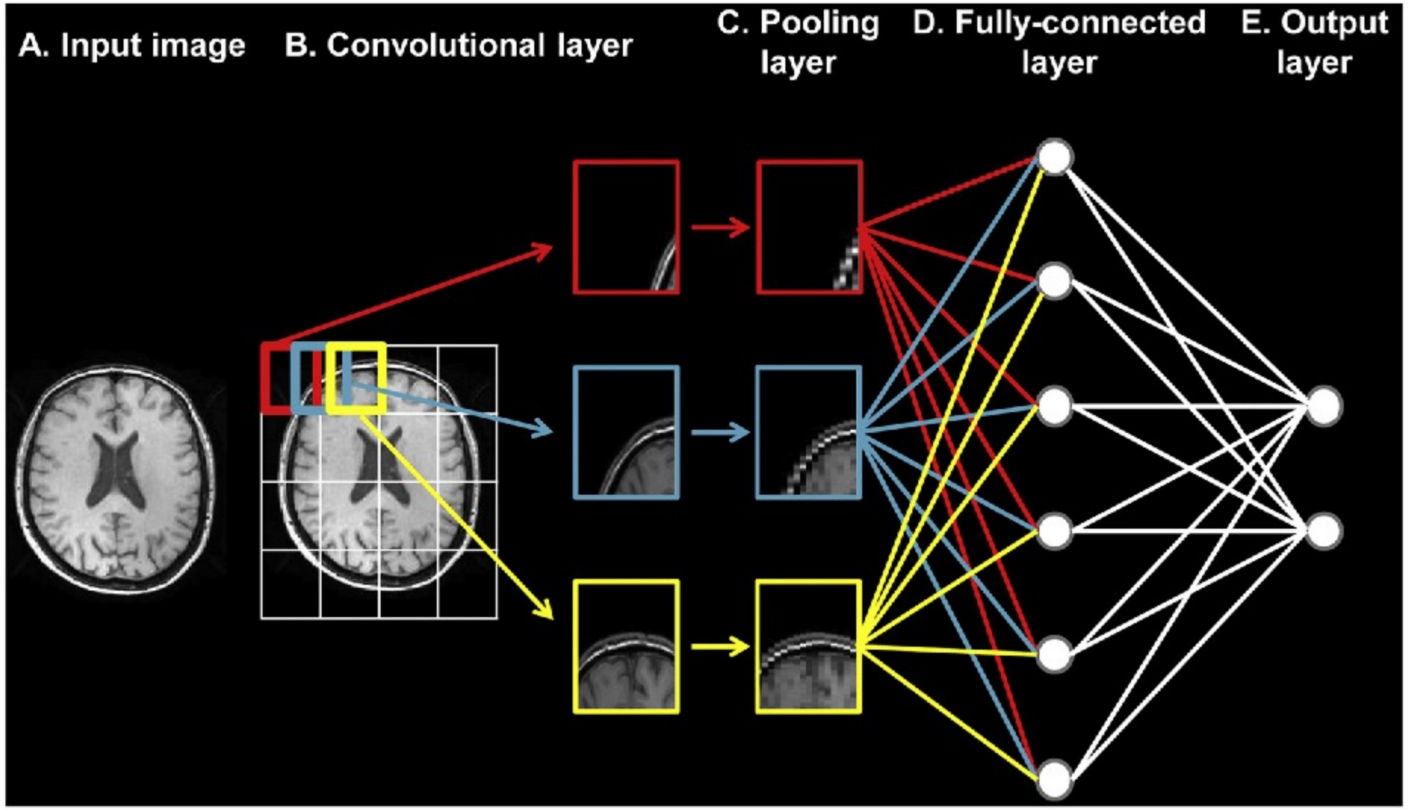
Early diagnosis is crucial

How?



Objective

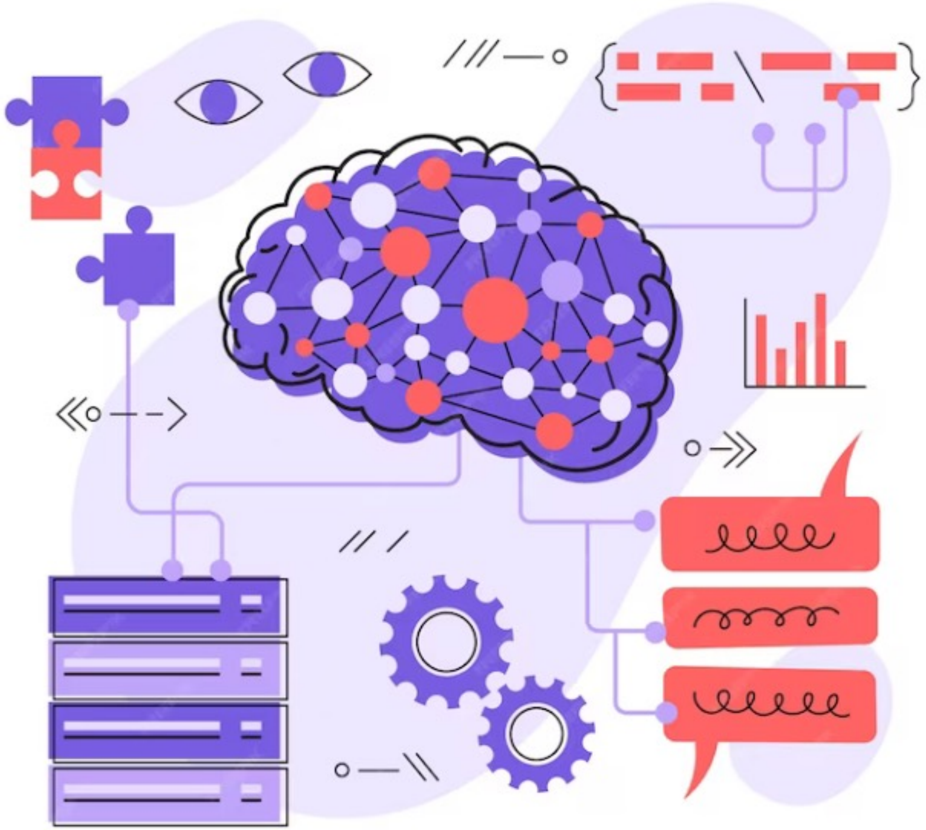
Develop and evaluate a **Deep Learning** model capable of **identifying distinctive patterns** in MRI images to discriminate between different stages



Content

Training and test sets (4 classes):

- Mild Demented
- Moderate Demented
- Non Demented
- Very Mild Demented



Methodology

1. Data preparation
2. Model comparison
3. Tune the model
4. Evaluate performance
5. Final conclusions

1. Data Preparation

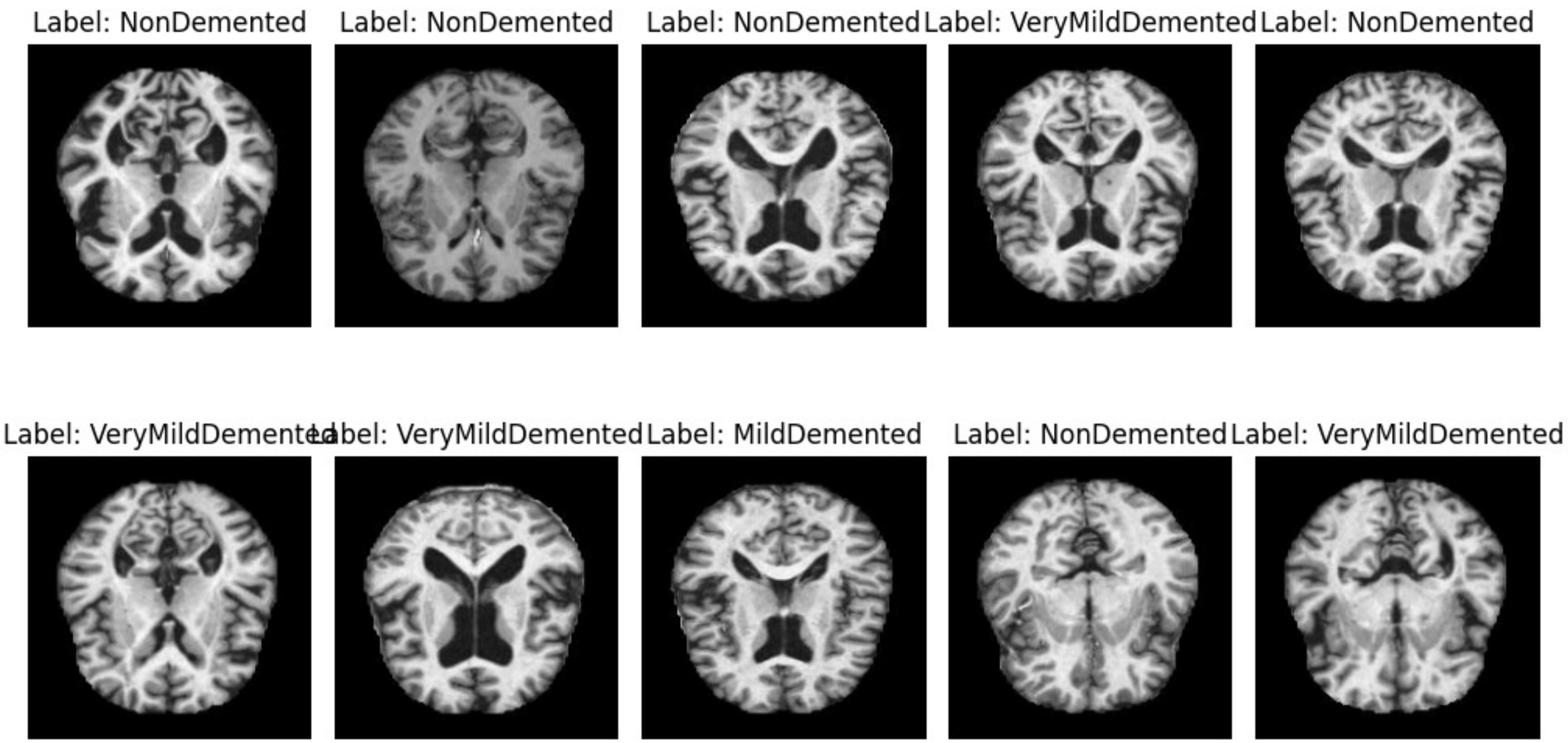
Import libraries

Built helper functions:
- read_data
- show_random_images_with_labels
- flatten_gray_scale
- plot_training_metrics
- prepare_for_test
- show_images

1. Data Preparation

View sample images:

```
show_random_images_with_labels(X_train, y_train, num_images=10)
```



1. Data Preparation

View proportion of classes:

Warning! Is unbalance

We should check AUC instead of only Accuracy

2. Model comparison

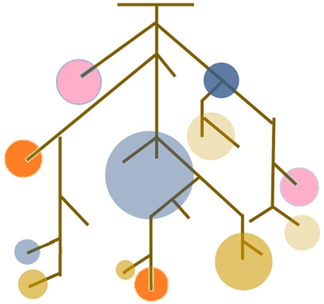
Random Forest

Basic Convolutional model

Transfer Learning:

- ResNet50
- InceptionV3
- DenseNet169
- VGG16

2. Model comparison



Random Forest

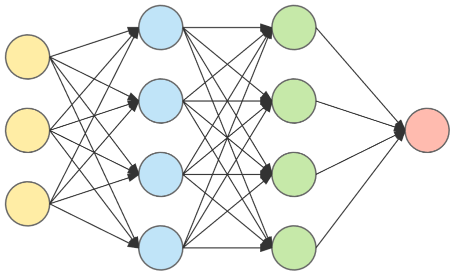
- Flatten X_train
- Apply shuffle
- Apply model

```
rf_clf = RandomForestClassifier()
np.mean(cross_val_score(rf_clf,X_train_rf_shuffled, y_train_shuffled,
cv = 5, scoring = "accuracy"))
```

92% accuracy

2. Model comparison

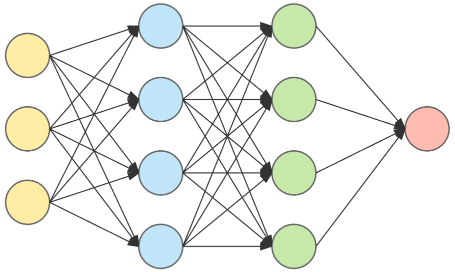
Simple convolutional model



```
datagen = IDG(rescale = 1./255, validation_split=0.1)

train_gen = datagen.flow_from_directory(directory=train_dir,
                                        target_size=DIM,
                                        batch_size=400,
                                        class_mode='categorical',
                                        subset='training',
                                        shuffle=True)

validation_gen = datagen.flow_from_directory(directory=train_dir,
                                        target_size=DIM,
                                        batch_size=400,
                                        class_mode='categorical',
                                        subset='validation',
                                        shuffle=True)

test_gen = datagen.flow_from_directory(directory=test_dir,
                                        target_size=DIM,
                                        batch_size=6400,
                                        class_mode='categorical')
```

```
Found 4610 images belonging to 4 classes.
Found 511 images belonging to 4 classes.
Found 1279 images belonging to 4 classes.
```

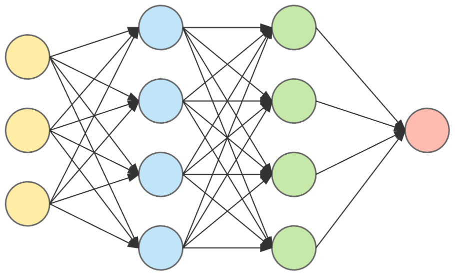
2. Model comparison



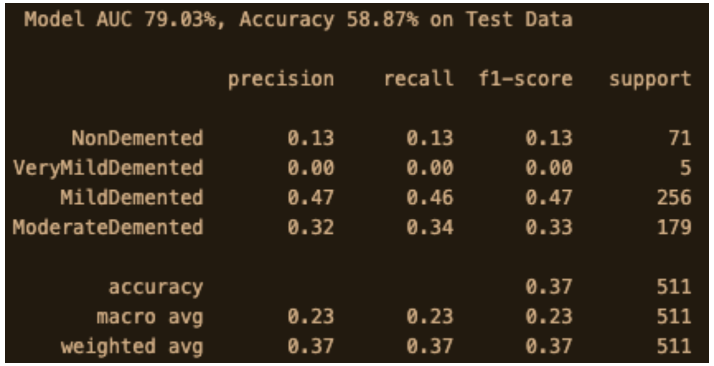
Simple convolutional model

```
# Create the model
model = Sequential()

# Add convolutional layer
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)))
# Add pooling layer
model.add(MaxPooling2D((2, 2)))

# Add another convolutional and pooling layers
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten for the dense layer
model.add(Flatten())

# Add dense layer
model.add(Dense(128, activation='relu'))

# Out layer with softmax activation for classification
model.add(Dense(len(CLASSES), activation='softmax'))

# Compile the model with personalized metrics
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=[AUC(name='auc'), Precision(name='precision'), Recall(name='recall')])

# Train the model
history = model.fit(train_gen,
                    epochs=50,
                    validation_data=validation_gen)
```

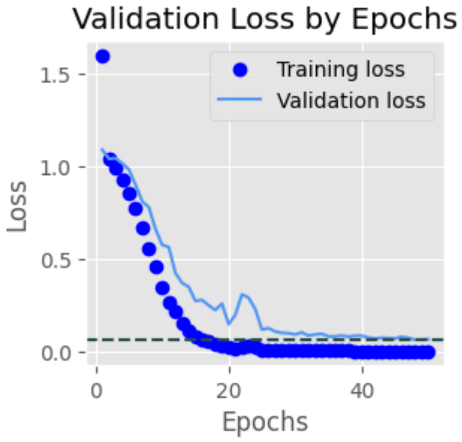
2. Model comparison

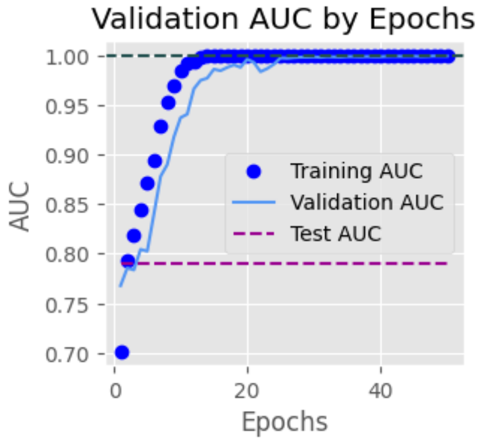


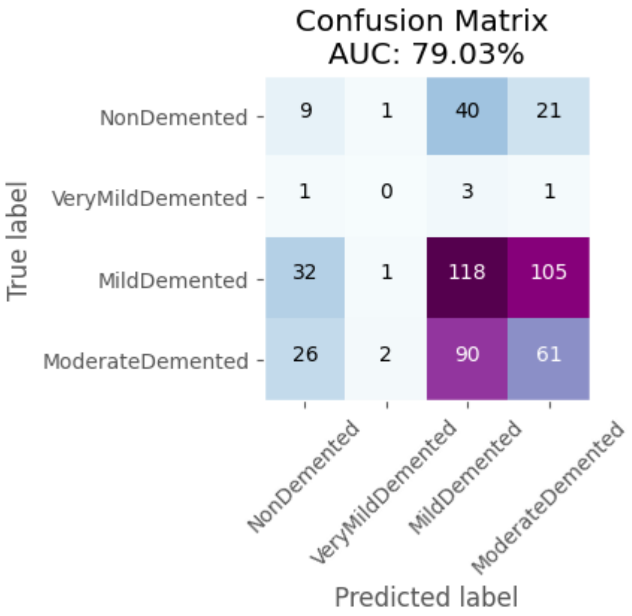
Simple convolutional model


Model AUC 79.03%, Accuracy 58.87% on Test Data

	precision	recall	f1-score	support
NonDemented	0.13	0.13	0.13	71
VeryMildDemented	0.00	0.00	0.00	5
MildDemented	0.47	0.46	0.47	256
ModerateDemented	0.32	0.34	0.33	179
accuracy			0.37	511
macro avg	0.23	0.23	0.23	511
weighted avg	0.37	0.37	0.37	511







2. Model comparison

Transfer Learning: ResNet50

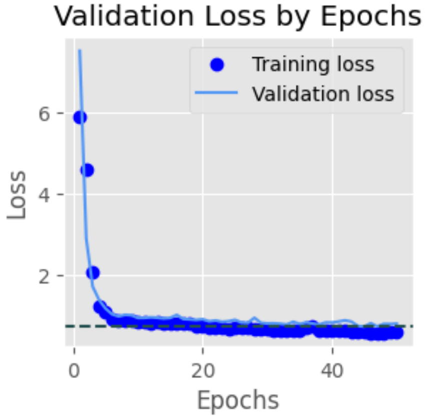
```
rn = ResNet50(input_shape=(IMG_SIZE,IMG_SIZE,3), weights='imagenet', include_top=False)
for layer in rn.layers:
    layer.trainable = False
x = Flatten()(rn.output)

prediction = Dense(4, activation='softmax')(x)

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

model.compile(optimizer='adam',
loss=tensorflow.losses.CategoricalCrossentropy(),
metrics=[keras.metrics.AUC(name='auc'),'acc'])
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=8,
                                          restore_best_weights=True)

tic = time.perf_counter()
history = model.fit(train_gen,
                    steps_per_epoch=len(train_gen),
                    validation_data=validation_gen,
                    validation_steps=len(validation_gen),
                    epochs=50, callbacks=callback)
# time
toc = time.perf_counter()
print("Total Time:{}".format(round((toc-tic)/60,2)))
```

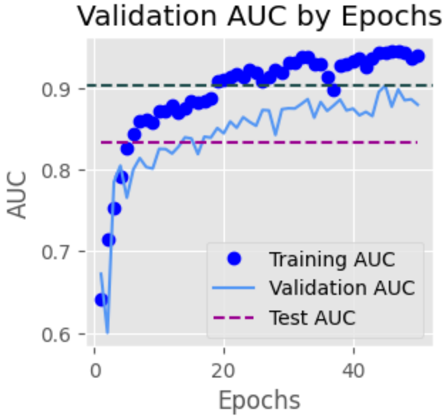
2. Model comparison

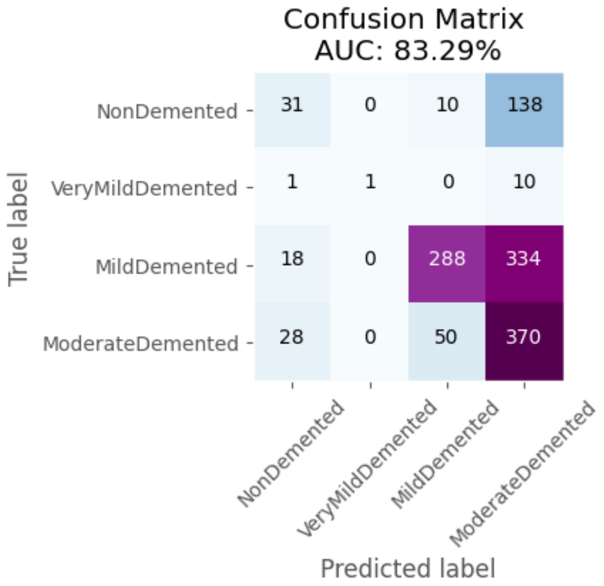
Transfer Learning: ResNet50

Model AUC 83.29%, Accuracy 53.95% on Test Data

	precision	recall	f1-score	support
NonDemented	0.40	0.17	0.24	179
VeryMildDemented	1.00	0.08	0.15	12
MildDemented	0.83	0.45	0.58	640
ModerateDemented	0.43	0.83	0.57	448
accuracy			0.54	1279
macro avg	0.66	0.38	0.39	1279
weighted avg	0.63	0.54	0.53	1279







2. Model comparison

Transfer Learning: InceptionV3

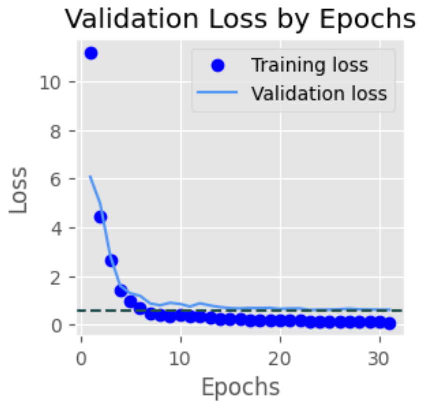
```
inception = InceptionV3(input_shape=(IMG_SIZE,IMG_SIZE,3), weights='imagenet', include_top=False)
for layer in inception.layers:
    layer.trainable = False
x = Flatten()(inception.output)

prediction = Dense(4, activation='softmax')(x)

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

model.compile(optimizer='adam',
loss=tensorflow.losses.CategoricalCrossentropy(),
metrics=[keras.metrics.AUC(name='auc'),'acc'])
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
                                           patience=8,
                                           restore_best_weights=True)

tic = time.perf_counter()
history = model.fit(train_gen,
                    steps_per_epoch=len(train_gen),
                    validation_data=validation_gen,
                    validation_steps=len(validation_gen),
                    epochs=50, callbacks=callback)
# time
toc = time.perf_counter()
print("Total Time:{}".format(round((toc-tic)/60,2)))
```

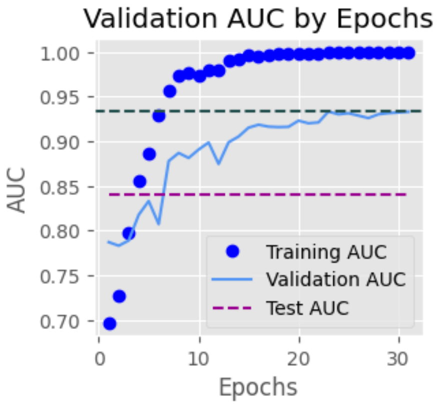
2. Model comparison

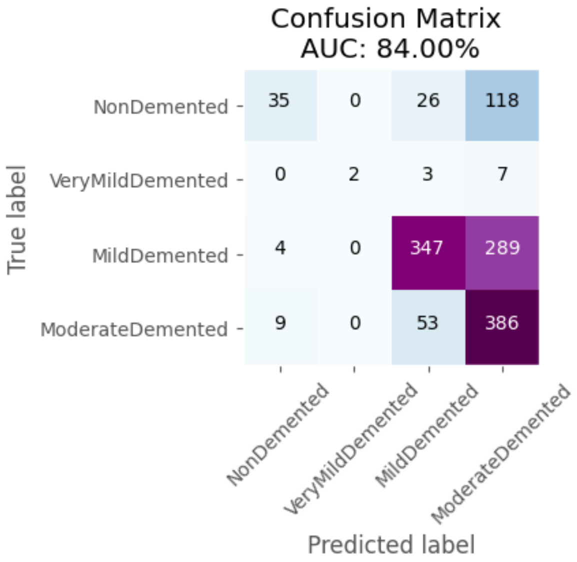
Transfer Learning: InceptionV3

Model AUC 84.00%, Accuracy 60.20% on Test Data

	precision	recall	f1-score	support
NonDemented	0.73	0.20	0.31	179
VeryMildDemented	1.00	0.17	0.29	12
MildDemented	0.81	0.54	0.65	640
ModerateDemented	0.48	0.86	0.62	448
accuracy			0.60	1279
macro avg	0.76	0.44	0.47	1279
weighted avg	0.69	0.60	0.59	1279







2. Model comparison

Transfer Learning: DenseNet169

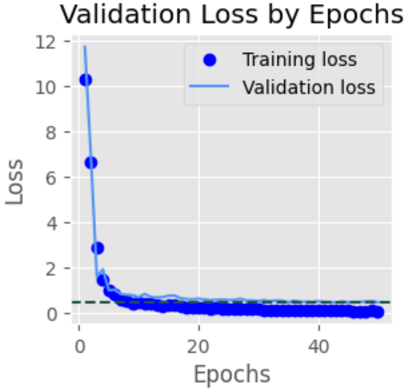
```
dense = DenseNet169(input_shape=(IMG_SIZE,IMG_SIZE,3), weights='imagenet', include_top=False)
for layer in dense.layers:
    layer.trainable = False
x = Flatten()(dense.output)

prediction = Dense(4, activation='softmax')(x)

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

model.compile(optimizer='adam',
loss=tensorflow.losses.CategoricalCrossentropy(),
metrics=[keras.metrics.AUC(name='auc'),'acc'])
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
                                         patience=8,
                                         restore_best_weights=True)

tic = time.perf_counter()
history = model.fit(train_gen,
                    steps_per_epoch=len(train_gen),
                    validation_data=validation_gen,
                    validation_steps=len(validation_gen),
                    epochs=50, callbacks=callback)
# time
toc = time.perf_counter()
print("Total Time:{}".format(round((toc-tic)/60,2)))
```

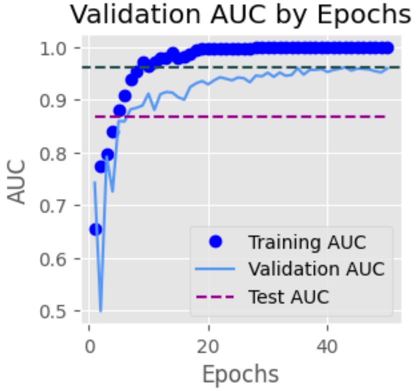
2. Model comparison

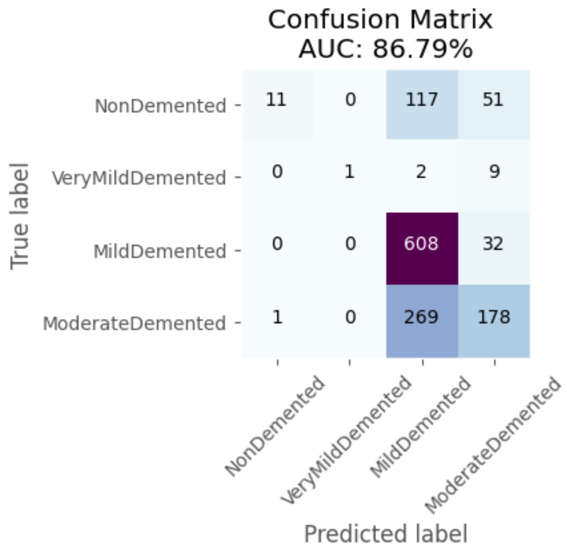
Transfer Learning: DenseNet169

Model AUC 86.79%, Accuracy 62.39% on Test Data

	precision	recall	f1-score	support
NonDemented	0.92	0.06	0.12	179
VeryMildDemented	1.00	0.08	0.15	12
MildDemented	0.61	0.95	0.74	640
ModerateDemented	0.66	0.40	0.50	448
accuracy			0.62	1279
macro avg	0.80	0.37	0.38	1279
weighted avg	0.67	0.62	0.56	1279







2. Model comparison

Transfer Learning: VGG16

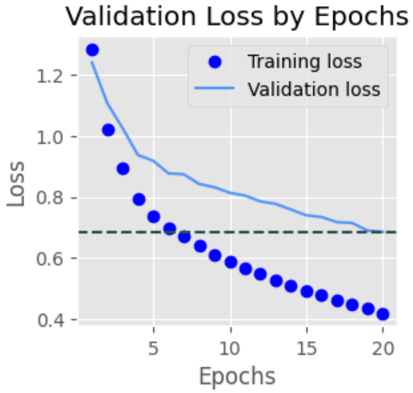
```
vgg = VGG16(input_shape=(IMG_SIZE,IMG_SIZE,3), weights='imagenet', include_top=False)
for layer in vgg.layers:
    layer.trainable = False
x = Flatten()(vgg.output)

prediction = Dense(4, activation='softmax')(x)

model = Model(inputs=vgg.input, outputs=prediction)
model.summary()

model.compile(optimizer='adam',
loss=tensorflow.losses.CategoricalCrossentropy(),
metrics=[keras.metrics.AUC(name='auc'),'acc'])
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=3,
                                          restore_best_weights=True)
tic = time.perf_counter()
history = model.fit(train_gen,
                    steps_per_epoch=len(train_gen),
                    validation_data=validation_gen,
                    validation_steps=len(validation_gen),
                    epochs=20, callbacks=callback)
# time
toc = time.perf_counter()
print("Total Time:{}".format(round((toc-tic)/60,2)))
```

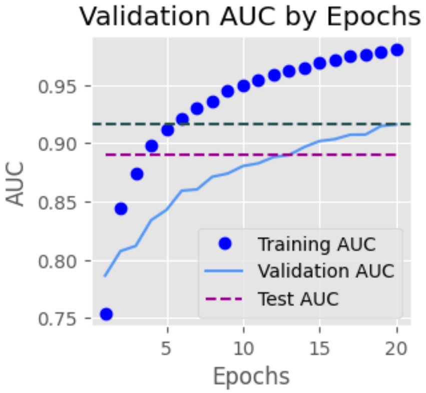
2. Model comparison

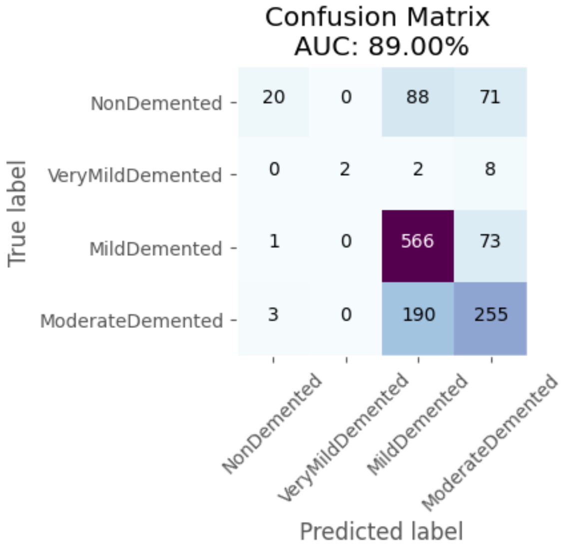
Transfer Learning: VGG16

Model AUC 89.00%, Accuracy 65.91% on Test Data

	precision	recall	f1-score	support
NonDemented	0.83	0.11	0.20	179
VeryMildDemented	1.00	0.17	0.29	12
MildDemented	0.67	0.88	0.76	640
ModerateDemented	0.63	0.57	0.60	448
accuracy			0.66	1279
macro avg	0.78	0.43	0.46	1279
weighted avg	0.68	0.66	0.62	1279





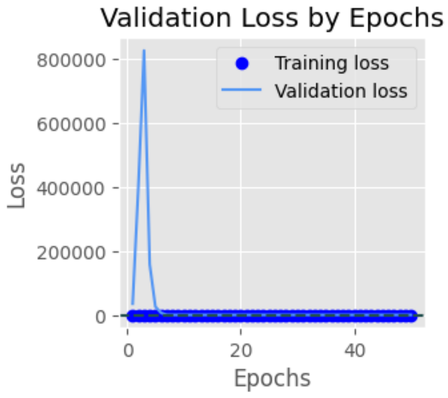


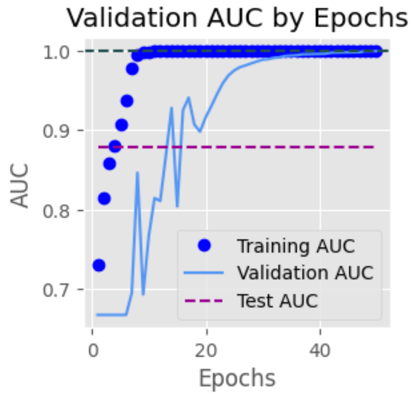
3. Tune the model

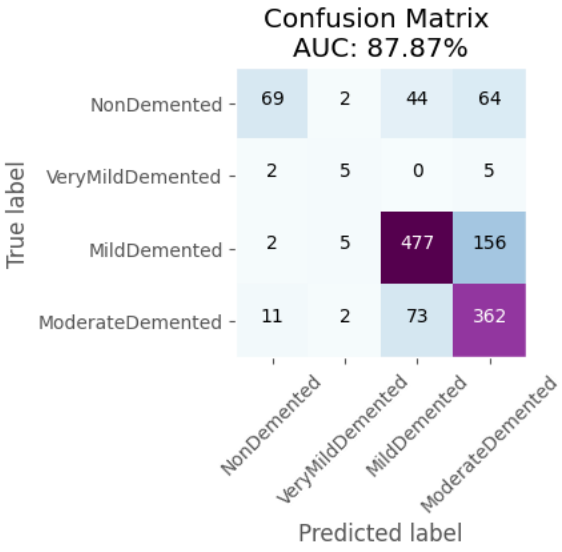
DenseNet169 Fine-tuned

```python
# Unfreeze the last few convolutional blocks for fine-tuning
for layer in dense.layers[:-10]:  # Unfreeze the last 10 layers for fine-tuning
    layer.trainable = True
```

```
Model AUC 87.87%, Accuracy 71.38% on Test Data

                   precision    recall  f1-score   support

     NonDemented       0.82      0.39      0.52       179
VeryMildDemented       0.36      0.42      0.38        12
    MildDemented       0.80      0.75      0.77       640
ModerateDemented       0.62      0.81      0.70       448

        accuracy                           0.71      1279
       macro avg       0.65      0.59      0.60      1279
    weighted avg       0.74      0.71      0.71      1279
```







True \ Predicted	NonDemented	VeryMildDemented	MildDemented	ModerateDemented
NonDemented	69	2	44	64
VeryMildDemented	2	5	0	5
MildDemented	2	5	477	156
ModerateDemented	11	2	73	362

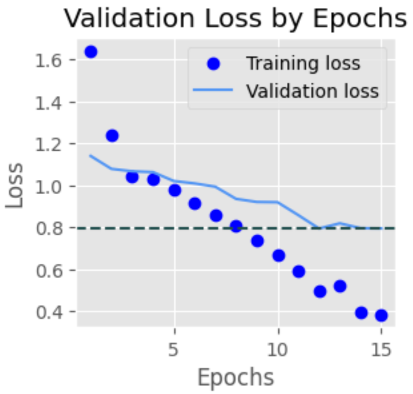
3. Tune the model

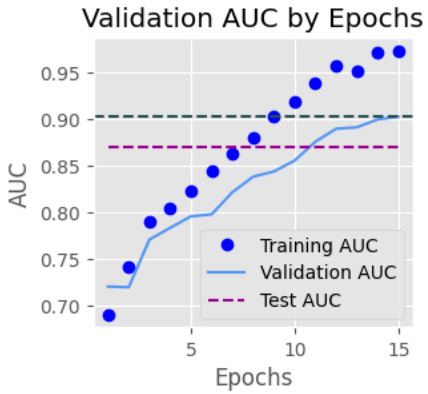
VGG16
Fine-tuned

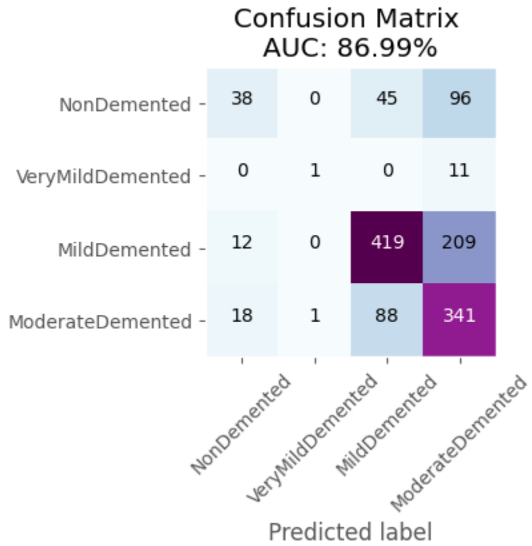
```
# Freeze all layers except the last convolutional block
for layer in vgg.layers[:-4]:
    layer.trainable = False
```

Model AUC 86.99%, Accuracy 62.47% on Test Data

	precision	recall	f1-score	support
NonDemented	0.56	0.21	0.31	179
VeryMildDemented	0.50	0.08	0.14	12
MildDemented	0.76	0.65	0.70	640
ModerateDemented	0.52	0.76	0.62	448
accuracy			0.62	1279
macro avg	0.58	0.43	0.44	1279
weighted avg	0.64	0.62	0.61	1279





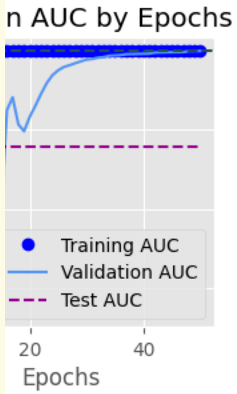


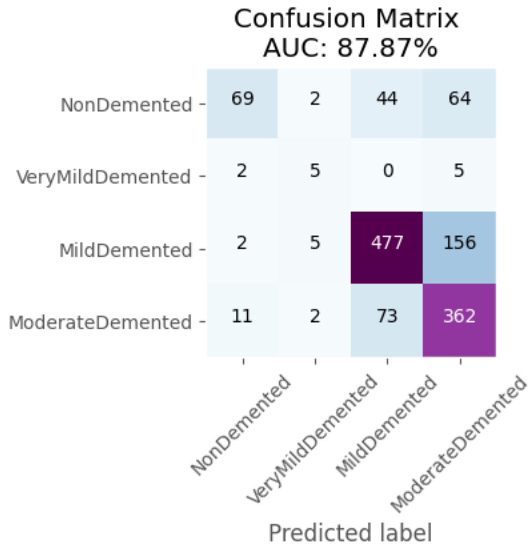
3. Tune the model

DenseNet169
Fine-tuned

```python
# Unfreeze the last few convolutional blocks for fine-tuning
for layer in dense.layers[:-10]:  # Unfreeze the last 10 layers for fine-tuning
    layer.trainable = True
```

```
Model AUC 87.87%, Accuracy 71.38% on Test Data

                  precision    recall  f1-score   supp

     NonDemented       0.82      0.39      0.52
VeryMildDemented       0.36      0.42      0.38
    MildDemented       0.80      0.75      0.77
ModerateDemented       0.62      0.81      0.70

        accuracy                           0.71
       macro avg       0.65      0.59      0.60
    weighted avg       0.74      0.71      0.71
```

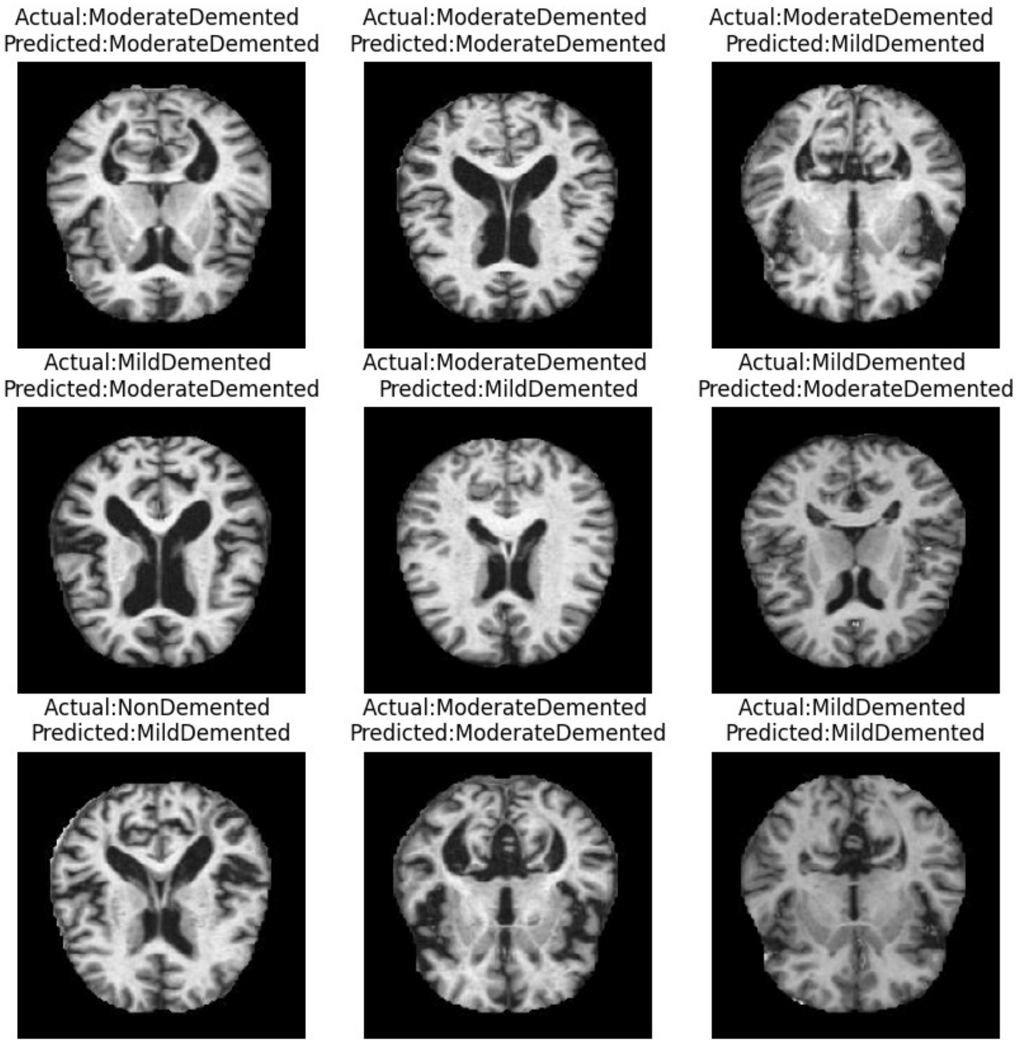




4. Evaluate with test images

DenseNet169
Fine-tuned

```
show_images(train_gen, y_pred)
```



5. Conclusions

Model performance

DenseNet169 and VGG16 exhibited the highest performance

Impact and clinical implication

model could serve as a valuable tool for radiologists and neurologists

Future perspectives

1. Refining the deep learning models
2. Data augmentation tasks
3. Collaboration with healthcare professionals and institutions