VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Soft Computing Digital Assignment (ITA6004)

Submitted to:

Dr Brijendra Singh

Submitted by:

22MCA0121 Varun Tiwari

22MCA0133 Priyansh Das

22MCA0139 Rajat Singh

22MCA0153 Austin Johnson

22MCA0162 Hrishikesh S G

Q4. Use the following

datasets: https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset/code

Train the model using AlexNet, ResNet and VGGNet.

Code (All together):

```
# code cell 1
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from tensorflow.keras.applications import ResNet50, VGG16

# code cell 2
data_directory = r'archive\train'

train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.3)

train_generator = train_datagen.flow_from_directory(
    data_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    data_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

# code cell 3
alexnet_model = Sequential()
alexnet_model.add(Conv2D(96, kernel_size=(11, 11), strides=(4, 4),
activation='relu', input_shape=(224, 224, 3)))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
alexnet_model.add(Conv2D(256, kernel_size=(5, 5), activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
```

```
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
alexnet_model.add(Flatten())
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))
alexnet_model.add(Dense(5, activation='softmax'))

alexnet_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

alexnet_history = alexnet_model.fit(train_generator,
validation_data=validation_generator, epochs=3)

# code cell 4
# Save the trained model
alexnet_model.save('alexnet_model.h5')

# code cell 5
# Load the saved model
saved_model = tf.keras.models.load_model('alexnet_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])

# code cell 6
# Make predictions using the saved model
predictions = saved_model.predict(test_generator)
predictions
```

```
# code cell 7
resnet_model = Sequential()
resnet_model.add(ResNet50(include_top=False, pooling='avg',
weights='imagenet'))
resnet_model.add(Dense(5, activation='softmax'))

resnet_model.layers[0].trainable = False

resnet_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

resnet_history = resnet_model.fit(train_generator,
validation_data=validation_generator, epochs=3)

# code cell 8
resnet_model.save('resnet_model.h5')

# code cell 9
# Load the saved model
saved_model = tf.keras.models.load_model('resnet_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])

# code cell 10
vgg_model = Sequential()
vgg_model.add(VGG16(include_top=False, pooling='avg', weights='imagenet'))
vgg_model.add(Dense(5, activation='softmax'))

vgg_model.layers[0].trainable = False

vgg_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
vgg_history = vgg_model.fit(train_generator,
validation_data=validation_generator, epochs=3)

# code cell 11
vgg_model.save('vgg_model.h5')

# code cell 12
# Load the saved model
saved_model = tf.keras.models.load_model('vgg_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])
```

Jupyter Notebook screenshots (code and outputs)

In [2]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.applications import ResNet50, VGG16
```

In [10]:
```
data_directory = r'archive\train'

train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.3)

train_generator = train_datagen.flow_from_directory(
    data_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    data_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 52493 images belonging to 5 classes.
Found 22497 images belonging to 5 classes.
```

In [11]:
```
alexnet_model = Sequential()
alexnet_model.add(Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu', input_shape=(224, 224, 3)))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
alexnet_model.add(Conv2D(256, kernel_size=(5, 5), activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
alexnet_model.add(Flatten())
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))
alexnet_model.add(Dense(5, activation='softmax'))

alexnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

alexnet_history = alexnet_model.fit(train_generator, validation_data=validation_generator, epochs=3)
```

```
Epoch 1/3
1641/1641 [==============================] - 1516s 923ms/step - loss: 0.2499 - accuracy: 0.9043 - val_loss: 0.2230 - val_accura
cy: 0.9279
Epoch 2/3
1641/1641 [==============================] - 1279s 779ms/step - loss: 0.1185 - accuracy: 0.9628 - val_loss: 0.1359 - val_accura
cy: 0.9586
Epoch 3/3
1641/1641 [==============================] - 1295s 789ms/step - loss: 0.1119 - accuracy: 0.9669 - val_loss: 0.1364 - val_accura
cy: 0.9560
```

In [12]:
```
# Save the trained model
alexnet_model.save('alexnet_model.h5')
```

In [3]:
```
# Load the saved model
saved_model = tf.keras.models.load_model('alexnet_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])
```

```
Found 50 images belonging to 5 classes.
2/2 [==============================] - 0s 63ms/step - loss: 0.0904 - accuracy: 0.9600
Test Loss: 0.09042396396398544
Test Accuracy: 0.9599999785423279
```

In [16]:
```
# Make predictions using the saved model
predictions = saved_model.predict(test_generator)
predictions
```

```
2/2 [==============================] - 0s 74ms/step
```

Out[16]:
```
array([[9.91539896e-01, 2.13779754e-06, 3.16425030e-05, 2.86038521e-05,
        8.39765277e-03],
       [9.68086481e-01, 4.26077080e-04, 1.10732147e-03, 3.04619316e-03,
        2.73338556e-02],
       [1.84629500e-01, 2.91937922e-05, 1.07955157e-05, 1.06019324e-04,
        8.15224528e-01],
       [9.92770493e-01, 1.38189068e-06, 8.48506606e-05, 1.14904969e-05,
        7.13177538e-03],
       [9.71647382e-01, 6.52114477e-06, 1.68600556e-04, 3.72680952e-05,
        2.81401202e-02],
       [9.77940023e-01, 7.85245429e-05, 9.74769413e-04, 4.64776211e-04,
        2.05419753e-02],
       [5.95463812e-01, 1.98034148e-04, 7.52603577e-04, 6.65900472e-04,
        4.02919620e-01],
       [2.14261264e-01, 6.28716271e-06, 2.46522495e-06, 3.50250666e-05,
        7.85695016e-01],
       [8.83774757e-01, 4.22853074e-04, 3.00153741e-03, 1.62796467e-03,
        1.11172900e-01],
       [9.61094558e-01, 1.05437357e-04, 2.43932661e-03, 5.15449152e-04,
        3.58452573e-02],
       [6.51392135e-12, 9.99985337e-01, 1.66503859e-11, 1.46579350e-05,
        4.59637381e-14],
       [4.97236306e-06, 9.91396129e-01, 8.69758060e-06, 8.58986285e-03,
        2.99155744e-07],
       [3.94106365e-07, 9.96886313e-01, 8.35139929e-07, 3.11251241e-03,
        1.30491982e-08],
```

```
 [5.53318591e-09, 9.99710739e-01, 1.23903039e-08, 2.89284420e-04,
  1.51993293e-10],
 [1.00528279e-08, 9.99581516e-01, 2.04353778e-08, 4.18540876e-04,
  2.47467880e-10],
 [5.00998283e-07, 9.96948302e-01, 9.46903754e-07, 3.05032823e-03,
  2.01894803e-08],
 [1.26276814e-06, 9.95135248e-01, 2.37248219e-06, 4.86111687e-03,
  5.60989157e-08],
 [2.49305185e-06, 9.94105935e-01, 4.22009089e-06, 5.88715030e-03,
  1.35478899e-07],
 [6.53380861e-09, 9.99644756e-01, 1.38838523e-08, 3.55280266e-04,
  1.47064680e-10],
 [1.04061464e-05, 9.87655222e-01, 1.77371221e-05, 1.23159969e-02,
  6.62076218e-07],
 [1.90635352e-09, 1.36068336e-15, 9.99999881e-01, 9.75753238e-08,
  1.57420957e-17],
 [4.53319984e-24, 0.00000000e+00, 1.00000000e+00, 6.28217301e-21,
  0.00000000e+00],
 [2.36584955e-23, 0.00000000e+00, 1.00000000e+00, 1.37476640e-21,
  0.00000000e+00],
 [2.49291220e-16, 4.82958904e-28, 1.00000000e+00, 5.33366415e-14,
  1.32019394e-30],
 [1.54917579e-04, 7.38811623e-10, 9.99816835e-01, 2.82869423e-05,
  4.34786829e-09],
 [4.01806233e-09, 8.08653835e-18, 1.00000000e+00, 1.20017307e-09,
  5.63361349e-17],
 [9.82036067e-12, 8.50069948e-22, 1.00000000e+00, 4.79881516e-11,
  2.26117348e-22],
 [3.96369160e-08, 1.77635633e-14, 9.99999762e-01, 2.39802006e-07,
  2.26502243e-15],
 [2.28995667e-08, 9.12323412e-20, 1.00000000e+00, 4.81678863e-11,
  6.07954986e-17],
 [6.95464042e-19, 1.61641111e-32, 1.00000000e+00, 1.80785246e-16,
```

In [18]:
```
resnet_model = Sequential()
resnet_model.add(ResNet50(include_top=False, pooling='avg', weights='imagenet'))
resnet_model.add(Dense(5, activation='softmax'))

resnet_model.layers[0].trainable = False

resnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

resnet_history = resnet_model.fit(train_generator, validation_data=validation_generator, epochs=3)
```

```
Epoch 1/3
1641/1641 [==============================] - 3064s 2s/step - loss: 0.5459 - accuracy: 0.8578 - val_loss: 0.4688 - val_accuracy:
0.8757
Epoch 2/3
1641/1641 [==============================] - 3042s 2s/step - loss: 0.2688 - accuracy: 0.9379 - val_loss: 0.3776 - val_accuracy:
0.8892
Epoch 3/3
1641/1641 [==============================] - 3004s 2s/step - loss: 0.2066 - accuracy: 0.9475 - val_loss: 0.3254 - val_accuracy:
0.8994
```

In [19]:
```
resnet_model.save('resnet_model.h5')
```

In [4]:
```
# Load the saved model
saved_model = tf.keras.models.load_model('resnet_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])
```

```
Found 50 images belonging to 5 classes.
2/2 [==============================] - 3s 717ms/step - loss: 0.1599 - accuracy: 0.9200
Test Loss: 0.15987545251846313
Test Accuracy: 0.9200000166893005
```

In [22]:
```
vgg_model = Sequential()
vgg_model.add(VGG16(include_top=False, pooling='avg', weights='imagenet'))
vgg_model.add(Dense(5, activation='softmax'))

vgg_model.layers[0].trainable = False

vgg_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

vgg_history = vgg_model.fit(train_generator, validation_data=validation_generator, epochs=3)
```

```
Epoch 1/3
1641/1641 [==============================] - 4822s 3s/step - loss: 0.5993 - accuracy: 0.9161 - val_loss: 0.3832 - val_accuracy:
0.8873
Epoch 2/3
1641/1641 [==============================] - 5091s 3s/step - loss: 0.2076 - accuracy: 0.9661 - val_loss: 0.2755 - val_accuracy:
0.8981
Epoch 3/3
1641/1641 [==============================] - 5406s 3s/step - loss: 0.1377 - accuracy: 0.9717 - val_loss: 0.2260 - val_accuracy:
0.9085
```

In [23]:
```
vgg_model.save('vgg_model.h5')
```

In [5]:
```
# Load the saved model
saved_model = tf.keras.models.load_model('vgg_model.h5')

# Test the model on new data
test_directory =  r'archive\test'

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_directory,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Evaluate the saved model
scores = saved_model.evaluate(test_generator)
print("Test Loss:", scores[0])
print("Test Accuracy:", scores[1])
```

```
Found 50 images belonging to 5 classes.
2/2 [==============================] - 3s 1s/step - loss: 0.0835 - accuracy: 0.9800
Test Loss: 0.08350679278373718
Test Accuracy: 0.9800000190734863
```

Conclusion:

This assignment involves the comparative study of AlexNet, ResNet and VGGNet models on Rice Image Dataset. From the above analysis (code) it was observed that:

- AlexNet model: gives a accuracy of 95.99% with a loss of 9%
- ResNet model: gives a accuracy of 92% with a loss of 15%
- VGGNet model: gives a accuracy of 98% with a loss of 8.3%

The number of epochs used is 3. This was done to quickly train the model as the dataset was big. For the first model, each epoch took 20 minutes, for the second it took 40 minutes and for the third it took 60 minutes.

If the number of epochs are increased than the loss can be decreased significantly

At the end, it is observed that the VGGNet model gives comparatively higher accuracy and has less test loss than the other 2 models. So VGGNet deep learning model best fits for this Rice Image Dataset.