Rice Disease Detection

In [1]:
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Bidirectional, LSTM, Reshape, Dropout, MultiHeadAttention
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger

import seaborn as sb
from sklearn.metrics import confusion_matrix
```

Allocate memory and environment to GPU

In [2]:
```
phy_devices = tf.config.experimental.list_physical_devices('GPU')
print(phy_devices)
if phy_devices:
    print("Memory allocation and computations pushed to GPU env")
    tf.config.experimental.set_memory_growth(phy_devices[0], True)
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
Memory allocation and computations pushed to GPU env
```

Data extraction and augmentation

In [3]:
```
#dataset path
dataset_dir = 'D:/Andrei/Andrei/Prog Applications/datasets'
dataset_name = '/_Preprocessed_Rice diseases exclusively_with_valid'
dataset_dir = dataset_dir + dataset_name

#image details
size = (224, 224)
img_color_mode = 'rgb'
img_type = '.jpg'
```

In [4]:
```
class_names=['blast','blight','tungro']
```

In [5]:
```
N=[]
for i in range(len(class_names)):
    N+=[i]

normal_mapping=dict(zip(class_names,N))
reverse_mapping=dict(zip(N,class_names))

def mapper(value):
    return reverse_mapping[value]
```

Data Retrieval Functions

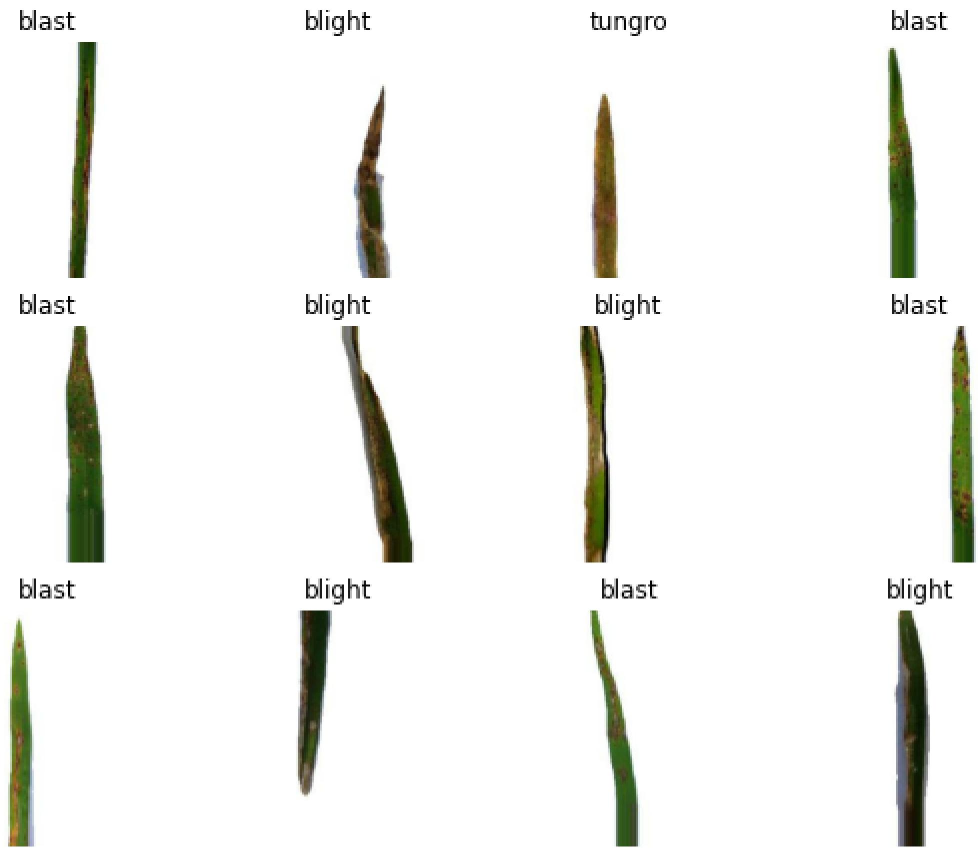
In [6]:
```
def get_trainXY_and_validXY(train_path, valid_path, size=(224,224), batch_size=1):
    train_batch = tf.keras.utils.image_dataset_from_directory(
        directory=train_path,
        image_size=size,
        labels='inferred',
        label_mode='categorical',
        shuffle=True,
        batch_size=batch_size,
        seed = 9
    )

    valid_batch = tf.keras.utils.image_dataset_from_directory(
        directory=valid_path,
        image_size=size,
        labels='inferred',
        label_mode='categorical',
        batch_size=batch_size,
        shuffle=True,
        seed = 9
    )

    X = []
    Y = []
    for images, labels in train_batch.take(-1):
        X.append(images.numpy()[0,:,:,:])
        Y.append(labels.numpy()[0])
    vX = []
    vY = []
    for images, labels in valid_batch.take(-1):
        vX.append(images.numpy()[0,:,:,:])
        vY.append(labels.numpy()[0])

    return np.array(X), np.array(Y), np.array(vX), np.array(vY)


def get_testXYbatch(test_path, size=(224,224), batch_size=1):
    test_batch = tf.keras.utils.image_dataset_from_directory(
        directory=test_path,
        image_size=size,
        labels='inferred',
        label_mode='categorical',
        batch_size=batch_size,
```

```
    )

    X = []
    Y = []
    for images, labels in test_batch.take(-1):
        X.append(images.numpy()[0,:,:,:])
        Y.append(labels.numpy()[0])
    return np.array(X), np.array(Y), test_batch
```

In [7]:
```
trainx, trainy, validx, validy = get_trainXY_and_validXY(f'{dataset_dir}/training', f'{dataset_dir}/validation')
testx, testy, testbatch = get_testXYbatch(f'{dataset_dir}/testing')
```

```
Found 1200 files belonging to 3 classes.
Found 48 files belonging to 3 classes.
Found 48 files belonging to 3 classes.
```

In [8]:
```
plt.figure(figsize=(10, 10))
for i in range(12):
    ax = plt.subplot(4, 4, i + 1)
    image = trainx[i]
    plt.imshow(image.astype("uint8"))
    plt.title(class_names[np.argmax(trainy[i], axis=-1)])
    plt.axis("off")
```



In [9]:
```
# normalize/standardize dataset
trainx /= 255
validx /= 255
testx /= 255
```

In [10]:
```
print(f"Training data shape: {trainx.shape}")
print(f"Validation data shape: {validx.shape}")
print(f"Testing data shape: {testx.shape}")
print(f"Classifications: {len(class_names)}, {class_names}")
```

```
Training data shape: (1200, 224, 224, 3)
Validation data shape: (48, 224, 224, 3)
Testing data shape: (48, 224, 224, 3)
Classifications: 3, ['blast', 'blight', 'tungro']
```

Custom Layers

In [11]:
```
def ReshapeLayer(x):
    shape = x.shape
    reshape = Reshape((shape[1],shape[2]*shape[3]))(x)
    return reshape

def BiLSTMLayer(x, neurons=128):
    # Tanh Activation provides access of the LSTM to the cuDNN which provides faster computation
    return Bidirectional(LSTM(neurons, activation='tanh', recurrent_dropout=0))(x)

def AttentionLayer(x, heads = 1, dim = 1, training = False):
    return MultiHeadAttention(num_heads=heads, key_dim=dim)(x, x, training=training)
```

Models

In [12]:
```
def DenseBilstm(attention=False):
    cnn = tf.keras.applications.DenseNet201(
        input_shape=(size[0],size[1],3),
        include_top=False,
        weights='imagenet'
    )
    cnn.trainable = False

    #Model Sequence
    images = cnn.input
    x = cnn.output
    if attention:
        x = AttentionLayer(x, heads = 2, dim = 2, training = True)
    x = ReshapeLayer(x)
    x = BiLSTMLayer(x, 128)
    pred = Dense(3, activation='softmax')(x)

    model = tf.keras.Model(inputs=images, outputs=pred)
    model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

    return model
```

In [13]:
```
def MoBilstm(attention=False):
    cnn = tf.keras.applications.MobileNet(
        input_shape=(size[0],size[1],3),
        include_top=False,
        weights='imagenet'
    )
    cnn.trainable = False

    #Model Sequence
    images = cnn.input
    x = cnn.output
    if attention:
        x = AttentionLayer(x, heads = 2, dim = 2, training = True)
    x = ReshapeLayer(x)
    x = BiLSTMLayer(x, 128)
    pred = Dense(3, activation='softmax')(x)

    model = tf.keras.Model(inputs=images, outputs=pred)
    model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
    return model
```

Training

In [14]:
```
model_label = 'mnet_bilstm_att_sample1'

# set early stopping criteria
pat = 5 # this is the number of epochs with no improvment after which the training will stop
early_stopping = EarlyStopping(monitor='val_loss', patience=pat, verbose=1, baseline=None)

# to save the history of models
csv_logger = CSVLogger(f'logs/{model_label}.log', separator=",", append=True)

# define the model checkpoint callback -> this will keep on saving the model as a physical file
def ModelCheckPointCB(model_label = 'mnet_bilstm_sample1', save_best_only=True):
    return ModelCheckpoint(
        f'model_checkpoints/{model_label}.h5',
        verbose=1, save_best_only=save_best_only
    )
```

In [15]:
```
model1 = DenseBilstm(attention = True)
model1.summary()
```

```
conv5_block31_2_conv (Conv2D)   (None, 7, 7, 32)     36864       conv5_block31_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block31_concat (Concatena (None, 7, 7, 1888)   0           conv5_block30_concat[0][0]
                                                                 conv5_block31_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block32_0_bn (BatchNormal (None, 7, 7, 1888)   7552        conv5_block31_concat[0][0]
__________________________________________________________________________________________________
conv5_block32_0_relu (Activatio (None, 7, 7, 1888)   0           conv5_block32_0_bn[0][0]
__________________________________________________________________________________________________
conv5_block32_1_conv (Conv2D)   (None, 7, 7, 128)    241664      conv5_block32_0_relu[0][0]
__________________________________________________________________________________________________
conv5_block32_1_bn (BatchNormal (None, 7, 7, 128)    512         conv5_block32_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block32_1_relu (Activatio (None, 7, 7, 128)    0           conv5_block32_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block32_2_conv (Conv2D)   (None, 7, 7, 32)     36864       conv5_block32_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block32_concat (Concatena (None, 7, 7, 1920)   0           conv5_block31_concat[0][0]
                                                                 conv5_block32_2_conv[0][0]
```

In [16]:

```
# model 1, DenseBiLSTM_noAttention
model_label = "DenseBiLSTM_withAttention"
history1 = model1.fit(
        x = ImageDataGenerator().flow(trainx,trainy,batch_size=16),
        validation_data = ImageDataGenerator().flow(validx, validy, batch_size=16),
        batch_size=32,
        epochs=50,
        callbacks=[early_stopping, ModelCheckPointCB(model_label), csv_logger],
        verbose=1,
    )
```

```
Epoch 12/50
75/75 [==============================] - 13s 173ms/step - loss: 1.1594e-04 - accuracy: 1.0000 - val_loss: 0.
2597 - val_accuracy: 0.9583

Epoch 00012: val_loss did not improve from 0.25488
Epoch 13/50
75/75 [==============================] - 13s 174ms/step - loss: 1.0179e-04 - accuracy: 1.0000 - val_loss: 0.
2629 - val_accuracy: 0.9583

Epoch 00013: val_loss did not improve from 0.25488
Epoch 14/50
75/75 [==============================] - 13s 173ms/step - loss: 9.0534e-05 - accuracy: 1.0000 - val_loss: 0.
2681 - val_accuracy: 0.9583


Epoch 00014: val_loss did not improve from 0.25488
Epoch 15/50
75/75 [==============================] - 13s 173ms/step - loss: 8.1028e-05 - accuracy: 1.0000 - val_loss: 0.
2737 - val_accuracy: 0.9583

Epoch 00015: val_loss did not improve from 0.25488
```

In [21]:
```
# model 2, MoBiLSTM_noAttention
model2 = MoBilstm(attention = True)
model2.summary()
```

```
conv_dw_13_relu (ReLU)         (None, 7, 7, 1024)   0           conv_dw_13_bn[0][0]
__________________________________________________________________________________________________
conv_pw_13 (Conv2D)            (None, 7, 7, 1024)   1048576     conv_dw_13_relu[0][0]
__________________________________________________________________________________________________
conv_pw_13_bn (BatchNormalizati (None, 7, 7, 1024)  4096        conv_pw_13[0][0]
__________________________________________________________________________________________________
conv_pw_13_relu (ReLU)         (None, 7, 7, 1024)   0           conv_pw_13_bn[0][0]
__________________________________________________________________________________________________
multi_head_attention_3 (MultiHe (None, 7, 7, 1024)  17420       conv_pw_13_relu[0][0]
                                                                 conv_pw_13_relu[0][0]
__________________________________________________________________________________________________
reshape_3 (Reshape)            (None, 7, 7168)      0           multi_head_attention_3[0][0]
__________________________________________________________________________________________________
bidirectional_3 (Bidirectional) (None, 256)         7472128     reshape_3[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                (None, 3)            771         bidirectional_3[0][0]
==================================================================================================
Total params: 10,719,183
Trainable params: 7,490,319
Non-trainable params: 3,228,864
```

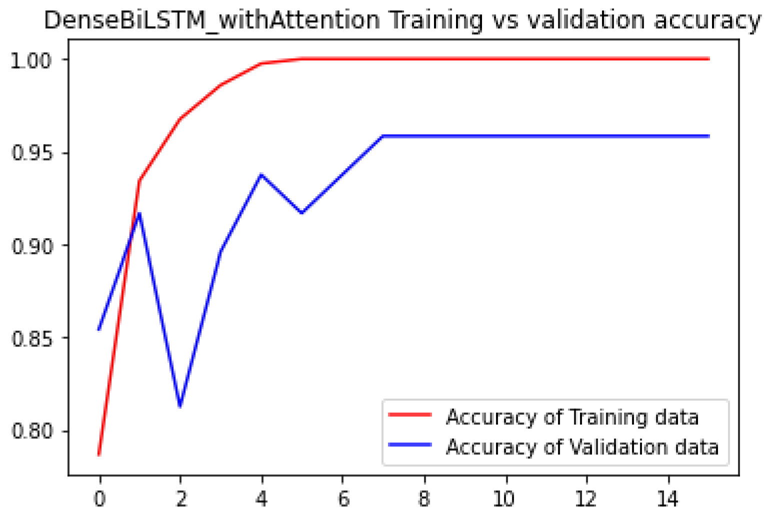
In [22]:
```
model_label2 = "MoBiLSTM_withAttention"
history2 = model2.fit(
        x = ImageDataGenerator().flow(trainx,trainy,batch_size=16),
        validation_data = ImageDataGenerator().flow(validx, validy, batch_size=16),
        batch_size=32,
        epochs=50,
        callbacks=[early_stopping, ModelCheckPointCB(model_label2), csv_logger],
        verbose=1,
    )
```

```
Epoch 00005: val_loss did not improve from 0.41249
Epoch 6/50
75/75 [==============================] - 4s 55ms/step - loss: 0.0300 - accuracy: 0.9933 - val_loss: 1.0289 -
val_accuracy: 0.8125

Epoch 00006: val_loss did not improve from 0.41249
Epoch 7/50
75/75 [==============================] - 4s 56ms/step - loss: 0.0290 - accuracy: 0.9867 - val_loss: 0.6456 -
val_accuracy: 0.8333

Epoch 00007: val_loss did not improve from 0.41249
Epoch 8/50
75/75 [==============================] - 4s 54ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.6483 -
val_accuracy: 0.8958

Epoch 00008: val_loss did not improve from 0.41249
Epoch 9/50
75/75 [==============================] - 4s 57ms/step - loss: 3.9505e-04 - accuracy: 1.0000 - val_loss: 0.70
04 - val_accuracy: 0.8958
```

Data Presentation

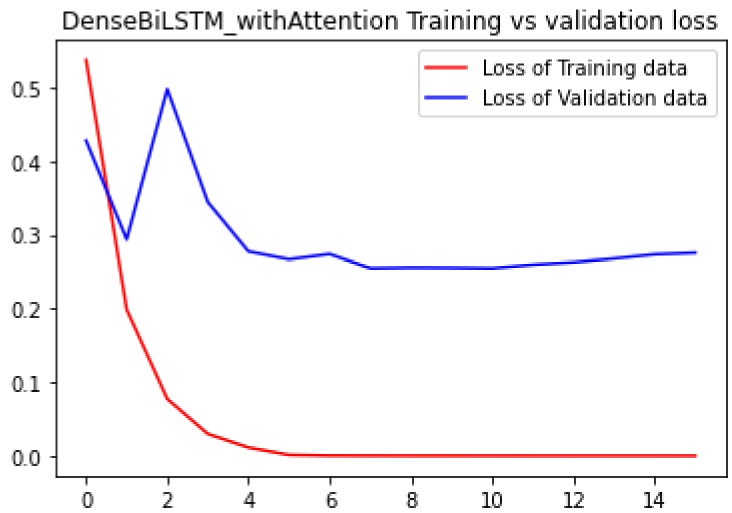
DenseBiLSTM

In [23]:
```
get_acc = history1.history['accuracy']
value_acc = history1.history['val_accuracy']
get_loss = history1.history['loss']
validation_loss = history1.history['val_loss']

epochs = range(len(get_acc))
plt.plot(epochs, get_acc, 'r', label='Accuracy of Training data')
plt.plot(epochs, value_acc, 'b', label='Accuracy of Validation data')
plt.title(f'{model_label} Training vs validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

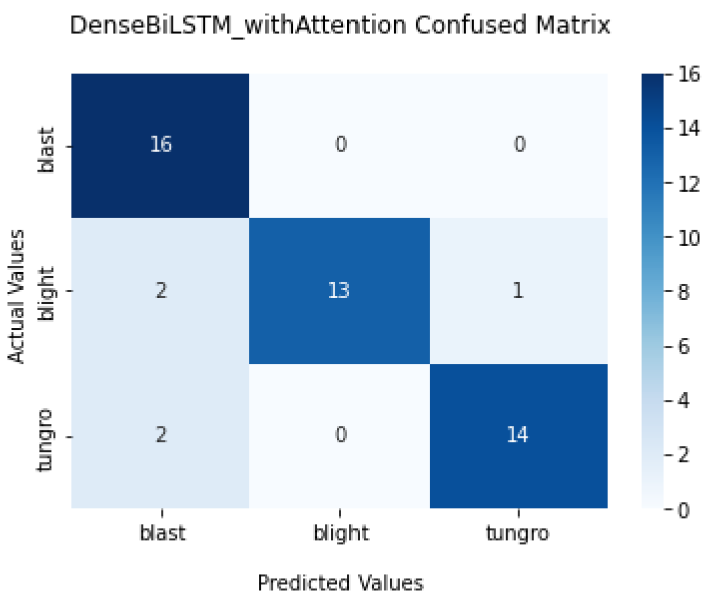
In [24]:
```
epochs = range(len(get_loss))
plt.plot(epochs, get_loss, 'r', label='Loss of Training data')
plt.plot(epochs, validation_loss, 'b', label='Loss of Validation data')
plt.title(f'{model_label} Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [25]:
```
y_pred=model1.predict(testx)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(testy,axis=1)

conf_matrix = confusion_matrix(ground, pred)
ax = sb.heatmap(conf_matrix, annot=True, cmap='Blues')
ax.set_title(f'{model_label} Confused Matrix\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ')
ax.xaxis.set_ticklabels(class_names)
ax.yaxis.set_ticklabels(class_names)
```

Out[25]:
```
[Text(0, 0.5, 'blast'), Text(0, 1.5, 'blight'), Text(0, 2.5, 'tungro')]
```



In [27]:
```
print(classification_report(ground,pred))
```

```
              precision    recall  f1-score   support

           0       0.80      1.00      0.89        16
           1       1.00      0.81      0.90        16
           2       0.93      0.88      0.90        16

    accuracy                           0.90        48
   macro avg       0.91      0.90      0.90        48
weighted avg       0.91      0.90      0.90        48
```

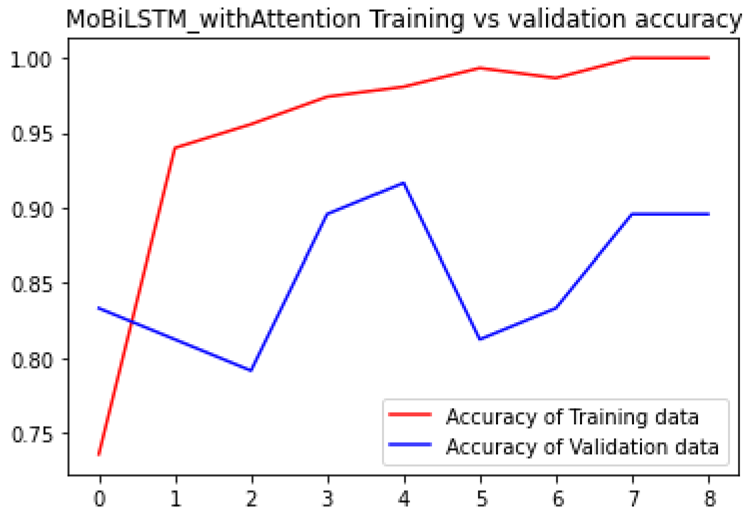
In [28]:
```
model1.evaluate(x=ImageDataGenerator().flow(testx, testy, batch_size=32), verbose = 1)
```

```
2/2 [==============================] - 0s 141ms/step - loss: 0.7898 - accuracy: 0.8958
```

Out[28]:
```
[0.7898158431053162, 0.8958333134651184]
```

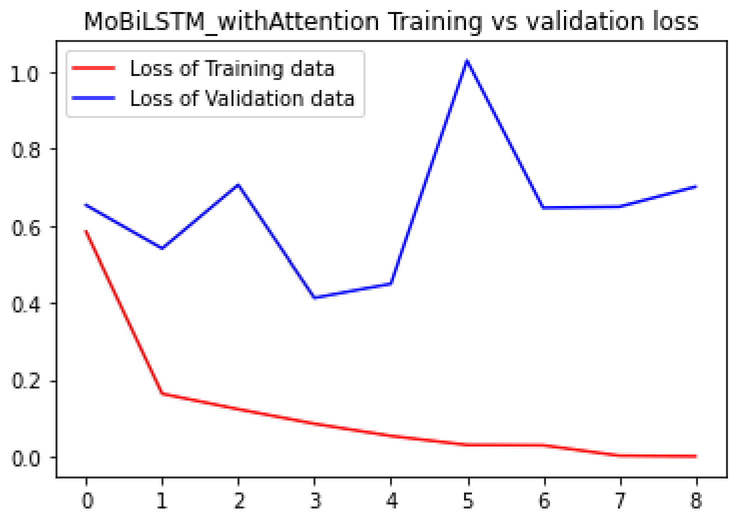
MoBiLSTM

In [29]:
```
get_acc = history2.history['accuracy']
value_acc = history2.history['val_accuracy']
get_loss = history2.history['loss']
validation_loss = history2.history['val_loss']

epochs = range(len(get_acc))
plt.plot(epochs, get_acc, 'r', label='Accuracy of Training data')
plt.plot(epochs, value_acc, 'b', label='Accuracy of Validation data')
plt.title(f'{model_label2} Training vs validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

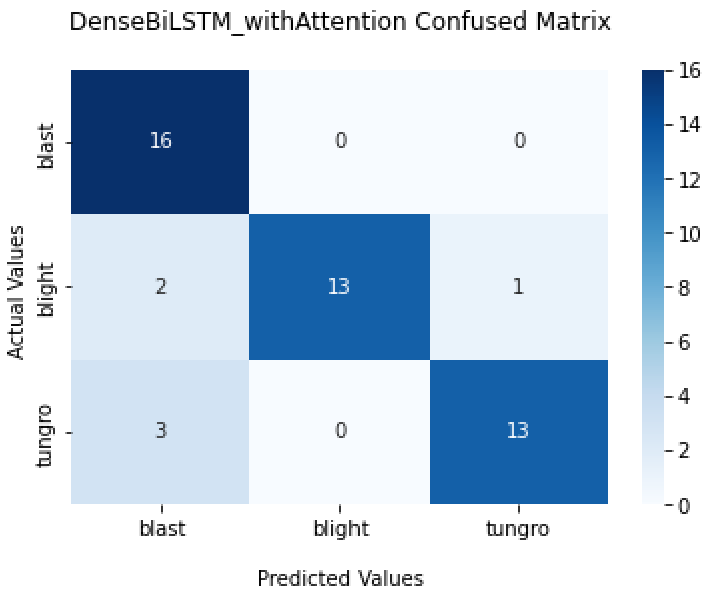
In [30]:
```
epochs = range(len(get_loss))
plt.plot(epochs, get_loss, 'r', label='Loss of Training data')
plt.plot(epochs, validation_loss, 'b', label='Loss of Validation data')
plt.title(f'{model_label2} Training vs validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [34]:
```
y_pred=model2.predict(testx)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(testy,axis=1)

conf_matrix = confusion_matrix(ground, pred)
ax = sb.heatmap(conf_matrix, annot=True, cmap='Blues')
ax.set_title(f'{model_label} Confused Matrix\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ')
ax.xaxis.set_ticklabels(class_names)
ax.yaxis.set_ticklabels(class_names)
```

Out[34]:
```
[Text(0, 0.5, 'blast'), Text(0, 1.5, 'blight'), Text(0, 2.5, 'tungro')]
```



In [35]:
```
print(classification_report(ground,pred))
```

```
              precision    recall  f1-score   support

           0       0.76      1.00      0.86        16
           1       1.00      0.81      0.90        16
           2       0.93      0.81      0.87        16

    accuracy                           0.88        48
   macro avg       0.90      0.88      0.88        48
weighted avg       0.90      0.88      0.88        48
```

In [36]:
```
model2.evaluate(x=ImageDataGenerator().flow(testx, testy, batch_size=32), verbose = 1)
```

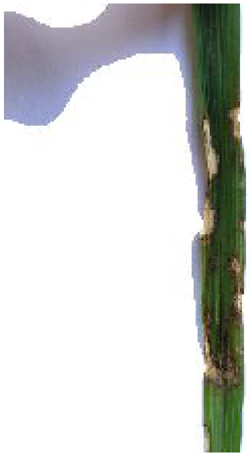
```
2/2 [==============================] - 0s 45ms/step - loss: 0.7572 - accuracy: 0.8750
```

Out[36]:
```
[0.7571511268615723, 0.875]
```

Single Prediction

In [37]:
```
# image = load_img(f"{dataset_dir}/blight/IMG_1034.jpg",target_size=(224,224))
image = load_img(f"{dataset_dir}/testing/blight/_2_7097357.jpg",target_size=(224,224))
image
```

Out[37]:



In [38]:
```
image=img_to_array(image)
image=image/255.0
prediction_image=np.array(image)
prediction_image= np.expand_dims(image, axis=0)
```

In [39]:
```
prediction=model1.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
#print(prediction)
#print(value)
print("Prediction is {}.".format(move_name))
```

```
Prediction is blight.
```

In [40]:
```
prediction=model2.predict(prediction_image)
value=np.argmax(prediction)
move_name=mapper(value)
#print(prediction)
#print(value)
print("Prediction is {}.".format(move_name))
```

```
Prediction is blight.
```

Thank you!