SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

Model Development Phase Template

Date	22 March 2024
Team ID	738171
Project Title	Neural Networks Ahoy: Cutting-Edge Ship Classification for Maritime Mastery
Maximum Marks	10 Marks

Initial Model Training Code (5 marks):

```
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator

[ ] train_datagen = ImageDataGenerator(rescale=1./255,zoom_range=0.2,shear_range=0.2)
    test_datagen = ImageDataGenerator(rescale=1./255)

[ ] train = train_datagen.flow_from_directory("/content/D:/SmartBridge/Ship Classification/input/train",target_size=(224,224),batch_size=5000)
    #test = test_datagen.flow_from_directory(Testpath,target_size=(224,224),batch_size=5000)#

    Found 6252 images belonging to 5 classes.
```

Model Validation and Evaluation Report (5 marks):

<table>
<tr><th>Model</th><th>Summary</th><th>Training and Validation Performance Metrics</th></tr>
<tr><td>VGG16</td><td>The VGG16 model is a convolutional neural network (CNN) architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. It gained significant popularity due to its simplicity and effectiveness in image classification tasks
let's adapt the analogy for ship classification in maritime mastery:1)Input Layer: Imagine this as the ship's deck where we lay out our radar images. Just like</td><td><pre>
[ ] from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.layers import Dense,Flatten
    from tensorflow.keras.models import Model

    def transfer_learning():
      base_model = VGG16(include_top = False,input_shape = (224,224,3))
      thr=149
      for layers in base_model.layers[:thr]:
        layers.trainable = False
      for layers in base_model.layers[thr:]:
        layers.trainable = False
      return base_model

[ ] def create_model():
      model=Sequential()
      vgg=transfer_learning()
      model.add(vgg)
      model.add(GlobalAveragePooling2D())
      x = Flatten()(vgg.output)
      output = Dense(5, activation ='softmax')(x)
      vgg16 = Model(vgg.input, output)
      vgg16.summary()
      return vgg16

[ ] from tensorflow.keras.models import Model
    from keras.models import Sequential
</pre></td></tr>
</table>

a chef's table, it's where we prepare our ingredients, but in this case, our ingredients are radar images of ships. These radar images are usually in a specific size, like spreading them out on a big radar screen that's 224x224 pixels big.

1. Convolutional Blocks: These are like different rooms in the ship where our experts are stationed. Each block is like a room where our experts analyze parts of the radar image closely. They're looking for important features, like the shape and size of ships, and they squash down the less important details. Just like VGG16, our ship classifier might have 5 blocks, and each one of these rooms does this process a couple of times before moving on.
2. Fully Connected Layers: Now we've gathered all the important details about the ship from the radar images. These layers are like the control room of the ship. Our experts gather all the important features they found and mix them together to get a complete understanding of what kind of ship it might be. Finally, they give us an idea of what type of ship is in the radar image.
3. Activation Function: This is like our ship classifier's captain. They check if the ship classification is good or

```
[ ] from tensorflow.keras.models import Model
    from keras.models import Sequential
```

```
model = create_model()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (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_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_3 (Flatten)         (None, 25088)             0

 dense_3 (Dense)             (None, 5)                 125445

=================================================================
Total params: 14840133 (56.61 MB)
Trainable params: 125445 (490.02 KB)
Non-trainable params: 14714688 (56.13 MB)
```

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

```
[ ] import keras
    from tensorflow import keras
    from keras.callbacks import EarlyStopping
    #from livelossplot import PlotLossesKeras
```

```
from keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(
    monitor='val_accuracy',  # Monitor the validation accuracy
    baseline=0.8,  # Desired target accuracy (e.g., 80%)
    patience=0,  # Number of epochs to wait before stopping if not improved
    verbose=1,  # Verbosity mode (0 or 1)
    mode='max',  # Mode for the metric (maximizing validation accuracy)
    restore_best_weights=True  # Restore model weights with the best validation accuracy
)
```

```
[ ] model.fit(x = train_set,validation_data=val_set,epochs=1,callbacks=[early_stopping])

26/26 [==============================] - 45s 2s/step - loss: 7.4717 - accuracy: 0.6557 - val_loss: 3.5493 - val_accuracy: 0.8128
<keras.src.callbacks.History at 0x7cadb4166e00>
```

	not. They ensure everything's running smoothly and the classification is accurate. 4. Dropout: Sometimes, there might be too many experts analyzing the radar images, and things could get a bit chaotic. Dropout is like giving some of our experts a break for a while. It helps prevent the ship classification process from becoming too complex and making mistakes. 5. Optimizer and Loss Function: These are like our ship classifier's navigators. They work to make sure our ship classification is getting better and better. They adjust things to ensure our classification is accurate and aligns with what we want. In summary, just like a recipe in a kitchen, our ship classification model follows a structured process to analyze radar images and classify ships accurately in maritime scenarios.	