Model Optimization and Tuning Phase Template

Date	24 March 2024
Team ID	738171
Project Title	NEURAL NETWORKS AHOY: CUTTING-EDGE SHIP CLASSIFICATION FOR MARITIME MASTERY
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th></tr>
<tr><td>VGG16</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
</pre></td></tr>
</table>

SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

	(code screenshots below)

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

model = create_model()

Model: "model_3"
```

```
[ ] 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>
```

```
[ ] import numpy as np

[ ] #save model
    model.save('vgg16-ship-classification.h5')

[ ] from tensorflow.keras.models import load_model
    model.load_weights('vgg16-ship-classification.h5')

from keras.preprocessing.image import load_img

[ ] !pip install pillow

Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)

[ ] from PIL import Image

[ ] from tensorflow.keras.preprocessing.image import load_img, img_to_array
```


```
[ ] img = load_img('/content/train/images/2778062.jpg')
    img = img.resize((224, 224)) #Resize the image to match the expected shape
    x = img_to_array(img)
    x = np.expand_dims(x, axis=0)
    pred = np.argmax(model.predict(x))
    op = ['Cargo', 'Military', 'Carrier', 'Cruise', 'Tankers']
    print(op[pred])

    1/1 [==============================] - 0s 20ms/step
    Cargo
```

```
img = load_img('/content/train/images/1642121.jpg')
img = img.resize((224, 224)) #Resize the image to match the expected shape
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = np.argmax(model.predict(x))
op = ['Cargo', 'Military', 'Carrier', 'Cruise', 'Tankers']
print(op[pred])

1/1 [==============================] - 0s 486ms/step
Cruise
```

```
from keras.preprocessing import image
import numpy as np

# Assuming you have already defined 'model' somewhere in your code

# Testing 1
img = image.load_img('/content/train/images/1653183.jpg', target_size=(224, 224))  # Reading Image
x = image.img_to_array(img)  # Converting Image into array
x = np.expand_dims(x, axis=0)  # Expanding dimensions
pred = np.argmax(model.predict(x))  # Predicting the higher probability index
op = ['Cargo', 'Military', 'Carrier', 'Cruise', 'Tankers']
print(op[pred])  # List indexing with output

1/1 [==============================] - 0s 18ms/step
Carrier
```

```
from keras.preprocessing import image
import numpy as np

# Assuming you have already defined 'model' somewhere in your code

# Testing 1
img = image.load_img('/content/train/images/2016356.jpg', target_size=(224, 224))  # Reading Image
x = image.img_to_array(img)  # Converting Image into array
x = np.expand_dims(x, axis=0)  # Expanding dimensions
pred = np.argmax(model.predict(x))  # Predicting the higher probability index
op = ['Cargo', 'Military', 'Carrier', 'Cruise', 'Tankers']
print(op[pred])  # List indexing with output

1/1 [==============================] - 0s 87ms/step
Tankers
```

This code implements a ship classification model using transfer learning with the VGG16 architecture in TensorFlow Keras.

1. The transfer_learning function initializes the VGG16 model with pre-trained weights and freezes layers except the last few.
2. The create_model function builds a sequential model with VGG16 as the base, adds a GlobalAveragePooling2D layer, and appends a Dense layer for classification.
3. Early stopping is implemented to monitor validation accuracy and halt training if it doesn't improve beyond a baseline.
4. The model is trained with the specified loss function, optimizer, and

	metrics. 5. After training, the model is saved to 'vgg16-ship-classification.h5' and loaded for inference. 6. An image is loaded, preprocessed, and fed into the model for prediction. 7. The predicted class is then mapped to the appropriate ship category and printed out. This code essentially provides a pipeline for creating, training, and using a ship classification model based on VGG16.

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
VGG16	1. **Transfer Learning**: VGG16 is a pre-trained convolutional neural network (CNN) model that has been trained on a large dataset (ImageNet) for image recognition tasks. Leveraging transfer learning, we can utilize the learned features of VGG16 and fine-tune them for our specific ship classification task. 2. **Deep Architecture**: VGG16 consists of 16 convolutional layers followed by fully connected layers. This deep architecture allows it to learn intricate features from images, which is beneficial for distinguishing between different types of ships based on their visual characteristics. 3. **Proven Performance:** VGG16 has demonstrated strong performance across various image classification benchmarks. Its architecture, with relatively simple 3x3 convolutional filters

	and max-pooling layers, strikes a balance between model complexity and effectiveness. 4. **Adjustable for Task:** By freezing the convolutional layers up to a certain depth (in this case, up to layer 149), we can retain the general features learned by VGG16 while allowing the model to adapt its fully connected layers to our specific ship classification problem. This approach helps prevent overfitting, especially when working with limited training data. Overall, VGG16 was chosen as the final model because of its robust architecture, proven performance in image classification, and suitability for transfer learning in the context of ship classification tasks.