Model Optimization and Tuning Phase Template

Date	20 july 2024
Team ID	SWTID1720084639
Project Title	Beneath the Waves: Unraveling Coral Mysteries through Deep Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Selection Phase involves evaluating and choosing the best deep learning models for image classification tasks. It includes assessing VGG16 for its straightforward architecture and strong feature extraction, ResNet for its scalable depth and residual connections that prevent vanishing gradients, Inception for its multi-scale feature extraction and computational efficiency, Xception for its enhanced depthwise separable convolutions, and DenseNet for its dense connections that improve gradient flow and parameter efficiency. This phase ensures the final model selection is justified based on performance metrics, accuracy, and suitability for diverse computer vision applications.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters	Optimal Values

RESNET

```python
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import models, layers

# Load the pre-trained ResNet50 model
resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the layers of the pre-trained model
for layer in resnet.layers:
    layer.trainable = False

# Create a new model using the ResNet50 base
resnet_model = models.Sequential()
resnet_model.add(resnet)

# Flatten the output and add dense layers for classification
resnet_model.add(layers.Flatten())
resnet_model.add(layers.Dense(256, activation='relu'))
resnet_model.add(layers.Dropout(0.5))
resnet_model.add(layers.Dense(2, activation='sigmoid'))

# Compile the model
resnet_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Display the model summary
resnet_model.summary()
```

```python
val_loss, val_accuracy = resnet_model.evaluate(test_set, steps=test_set.samples // batch_size)
print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')
```

```
1/1 [==============================] - 4s 4s/step - loss: 0.6900 - accuracy: 0.5938
Validation Accuracy: 59.38%
```

INCEPTION

```python
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras import models, layers

# Load the pre-trained InceptionV3 model
inception = InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the layers of the pre-trained model
for layer in inception.layers:
    layer.trainable = False

# Create a new model using the InceptionV3 base
inception_model = models.Sequential()
inception_model.add(inception)

# Add a Global Average Pooling layer
inception_model.add(layers.GlobalAveragePooling2D())

# Add dense layers for classification
inception_model.add(layers.Dense(256, activation='relu'))
inception_model.add(layers.Dropout(0.5))
inception_model.add(layers.Dense(2, activation='sigmoid'))

# Compile the model
inception_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Display the model summary
inception_model.summary()
```

```python
val_loss, val_accuracy = inception_model.evaluate(test_set, steps=test_set.samples // batch_size)
print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')
```

```
1/1 [==============================] - 3s 3s/step - loss: 0.4108 - accuracy: 0.8125
Validation Accuracy: 81.25%
```

XCEPTION

```python
from tensorflow.keras.applications import Xception
from tensorflow.keras import models, layers

# Load the pre-trained Xception model
xception = Xception(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the layers of the pre-trained model
for layer in xception.layers:
    layer.trainable = False

# Create a new model using the Xception base
xception_model = models.Sequential()
xception_model.add(xception)

# Add a Global Average Pooling layer
xception_model.add(layers.GlobalAveragePooling2D())

# Add dense layers for classification
xception_model.add(layers.Dense(256, activation='relu'))
xception_model.add(layers.Dropout(0.5))
xception_model.add(layers.Dense(2, activation='sigmoid'))

# Compile the model
xception_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Display the model summary
xception_model.summary()
```

```python
val_loss, val_accuracy = xception_model.evaluate(test_set, steps=test_set.samples // batch_size)
print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')
```

```
1/1 [==============================] - 1s 957ms/step - loss: 0.4534 - accuracy: 0.7188
Validation Accuracy: 71.88%
```

DENSENET	`from tensorflow.keras.applications import DenseNet121`
	`from tensorflow.keras import models, layers`
	
	`# Load the pre-trained DenseNet121 model`
	`densenet = DenseNet121(weights='imagenet', include_top=False, input_shape=(299, 299, 3))`
	
	`# Freeze the layers of the pre-trained model`
	`for layer in densenet.layers:`
	`    layer.trainable = False`
	
	`# Create a new model using the DenseNet121 base`
	`densenet_model = models.Sequential()`
	`densenet_model.add(densenet)`
	
	`# Add a Global Average Pooling layer`
	`densenet_model.add(layers.GlobalAveragePooling2D())`
	
	`# Add dense layers for classification`
	`densenet_model.add(layers.Dense(256, activation='relu'))`
	`densenet_model.add(layers.Dropout(0.5))`
	`densenet_model.add(layers.Dense(2, activation='sigmoid'))`
	
	`# Compile the model`
	`densenet_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])`
	
	`# Display the model summary`
	`densenet_model.summary()`
		`val_loss, val_accuracy = densenet_model.evaluate(test_set, steps=test_set.samples // batch_size)`
		`print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')`
		
		`1/1 [==============================] - 1s 1s/step - loss: 0.4950 - accuracy: 0.7812`
		`Validation Accuracy: 78.12%`

VGG16	`# Freeze the layers of the pre-trained model`
	`for layer in vgg.layers:`
	`    layer.trainable = False`
	
	`# Create a new model using the VGG16 base`
	`vggmodel = models.Sequential()`
	`vggmodel.add(vgg)`
	
	`# Flatten the output and add dense layers for classification`
	`vggmodel.add(layers.Flatten())`
	`vggmodel.add(layers.Dense(256, activation='relu'))`
	`vggmodel.add(layers.Dropout(0.5))`
	`vggmodel.add(layers.Dense(2, activation='sigmoid'))`
	
	`# Compile the model`
	`vggmodel.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])`
	
	`# Display the model summary`
	`vggmodel.summary()`
		`val_loss, val_accuracy = vggmodel.evaluate(test_set, steps=test_set.samples // batch_size)`
		`print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')`
		
		`1/1 [==============================] - 1s 969ms/step - loss: 0.6468 - accuracy: 0.8125`
		`Validation Accuracy: 81.25%`

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
VGG16	We chose VGG16 as our final model for the "BENEATH THE WAVES" project because it achieved the highest accuracy during hyperparameter tuning. Its deep architecture captures complex patterns in coral images effectively, and its proven performance in image

	recognition tasks ensures reliability. Additionally, VGG16's pre-trained weights facilitate transfer learning, improving our results with less data.