Model Optimization and Tuning Phase Template

Date	19 July 2024
Team ID	SWTID1720110092
Project Title	Beneath the Waves: Unraveling Coral Mysteries Through Deep Learning
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):

Model	Tuned Hyperparameters

Model 1	**IMG_SIZE**: Defines the target size for input images.
	`IMG_SIZE = (224, 224)`
	BATCH_SIZE: Specifies the number of samples per gradient update.
	EPOCHS: Number of times the entire training dataset is passed through the model.
	`BATCH_SIZE = 16  EPOCHS = 5`
	train_datagen and test_datagen: Configures the image data generators for training and testing, including data augmentation parameters

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

fine_tune_at: Number of layers to freeze during fine-tuning.

```
fine_tune_at = 100
```

Dropout rates: Fraction of the input units to drop for the dropout layers.

```
x = Dropout(0.3)(x)
```

Learning rate: Learning rate for the Adam optimizer.

```
model = Model(inputs, outputs)
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Model 1	**EfficientNetB0 Backbone**: EfficientNetB0 is known for its efficient architecture, providing a good balance between accuracy and

	computational efficiency. It is a strong feature extractor due to its pre-trained weights on the ImageNet dataset. **Data Augmentation**: The use of ImageDataGenerator with various augmentation techniques helps to generalize the model better by simulating real-world variations in the training images. **Fine-Tuning**: By setting fine_tune_at = 100, the model partially fine-tunes the EfficientNetB0 backbone. **Dropout Layers**: Dropout layers are used to prevent overfitting by randomly dropping a fraction of the units during training. This makes the model more robust and improves generalization. **Reduced Complexity**: By reducing the dropout rates and the data augmentation parameters, the model converges faster, which is beneficial when working with smaller subsets of the data. **Model Compilation**: The model is compiled with the Adam optimizer, a popular choice for its adaptive learning rate properties, along with categorical cross-entropy loss and accuracy as the metric. **Training and Validation Strategy**: The training and validation strategy ensures that the model is evaluated on unseen data, providing a good indication of its generalization capability.