Model Development Phase Template

Date	13 July 2024
Team ID	SWTID1720157891
Project Title	Rice Classification using CNN
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

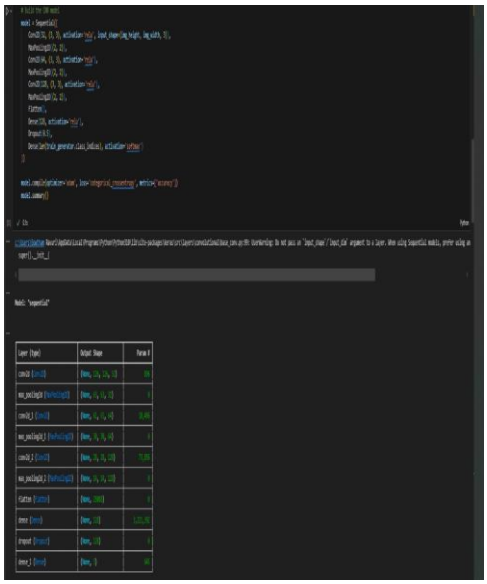
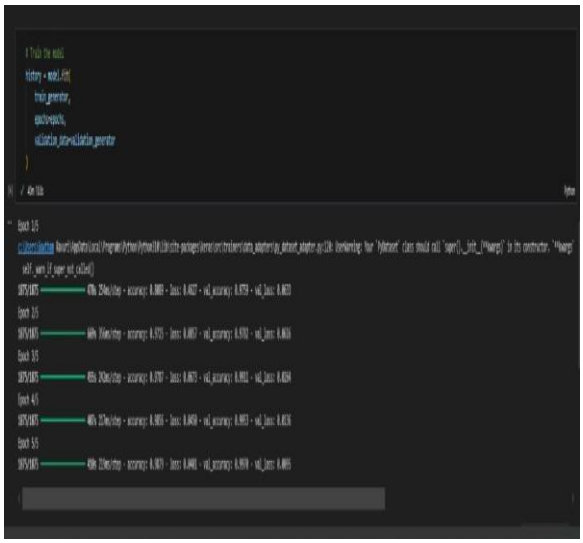
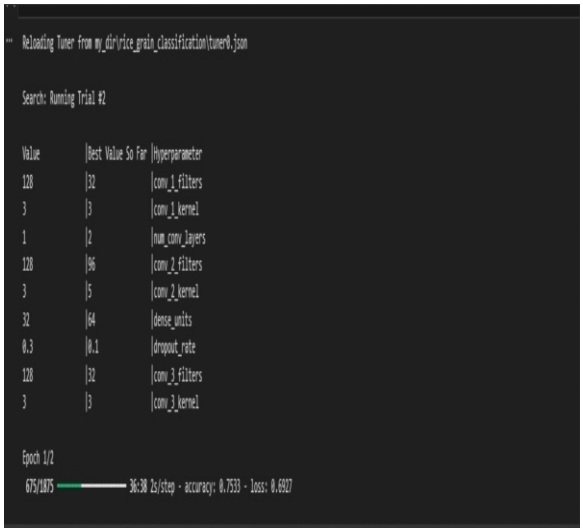
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

```
from tensorflow.keras.layers import BatchNormalization

# Create ImageDataGenerator instances for training and validation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)



# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(train_generator.class_indices), activation='softmax')
])

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


# Train the model
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator
)
# Evaluate the model and print the accuracy
validation_loss, validation_accuracy = model.evaluate(validation_generator)
print(f'Validation Accuracy: {validation_accuracy * 100:.2f}%')
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

Model 1		
Model 2		

	 ``` # Perform hyperparameter search tuner.search(train_generator, epochs=2, validation_data=validation_generator) # Get the optimal hyperparameters best_hps = tuner.get_best_hyperparameters(num_trials=1)[0] # Build and train the model with the optimal hyperparameters model = tuner.hypermodel.build(best_hps) history = model.fit(train_generator, epochs=epochs, validation_data=validation_generator) # Evaluate the model and print the accuracy validation_loss, validation_accuracy = model.evaluate(validation_generator) print(f'Validation Accuracy: {validation_accuracy * 100:.2f}%') # Predict with the trained model using a random example image img_preprocessed = np.expand_dims(img_array, axis=0) predictions = model.predict(img_preprocessed) predicted_class_index = np.argmax(predictions, axis=1) predicted_class = list(train_generator.class_indices.keys())[predicted_class_index[0]] # Display the prediction result plt.imshow(img) plt.title(f'True Class: {true_class}\nPredicted Class: {predicted_class}') plt.axis('off') plt.show() print(f'Predicted Class: {predicted_class}') print(f'True Class: {true_class}') ```	
...	...	...