Model Development Phase Template

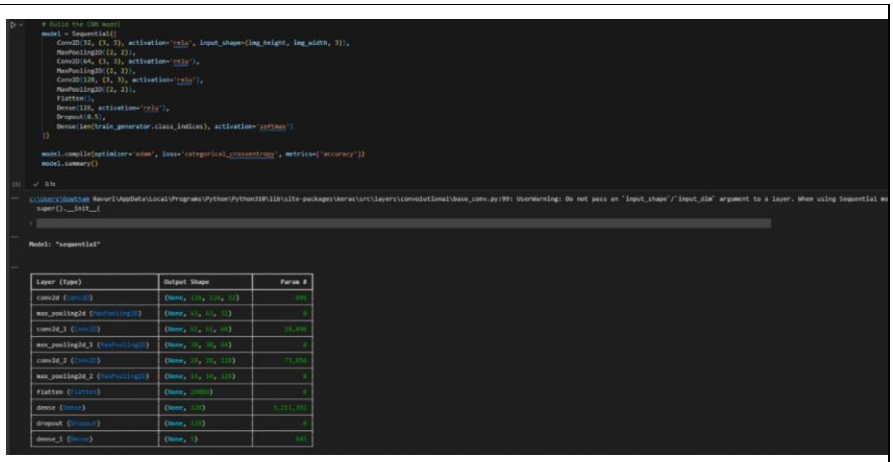
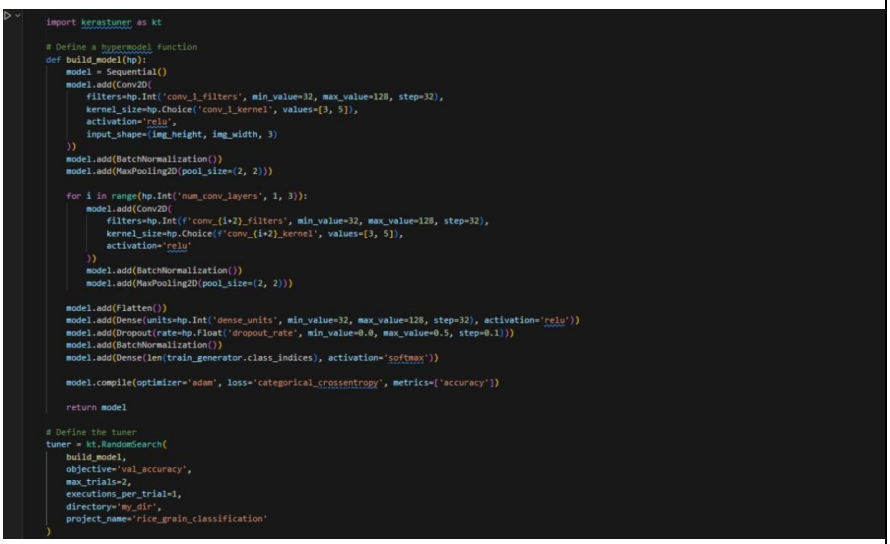
Date	13 July 2024
Team ID	SWTID1720157891
Project Title	Rice Classification using CNN
Maximum Marks	5 Marks

Model Selection Report

In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

Model Selection Report:

Model	Description
Model 1	The standard CNN model is a deep learning model designed for image classification tasks. It consists of multiple convolutional layers followed by pooling layers, fully connected layers, and a softmax output layer. **Key Features:** • Standard architecture with convolutional, pooling, and fully connected layers. • No hyperparameter tuning applied.

	• 
Model 2	The hyper-tuned CNN model is the same architecture as the standard CNN but with optimized hyperparameters. Hyperparameter tuning was performed using grid search to find the best combination of hyperparameters. **Key Features:** • Same base architecture as the standard CNN. • Hyperparameters optimized for improved performance. • 

	• 
...	...

```python
# 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}')
```