Model Optimization and Tuning Phase

Date	15 July 2024
Team ID	SWTID1720085076
Project Title	Rice Type Classification using CNN
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	MobileNET Hyperparam1: Two Conv2D Layers This model incorporates two additional Conv2D layers to deepen the network. The extra convolutional layers aim to capture more intricate features from the images, potentially improving accuracy

	`` # Build fifth CNN Model: by two Conv2D layer model_5 = tf.keras.Sequential([tf.keras.layers.Rescaling(1./255, input_shape=(150, 150, 3)), tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.MaxPool2D(pool_size=2, strides=2), tf.keras.layers.Flatten(), tf.keras.layers.Dense(units=300, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.Dense(units=100, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.Dropout(0.5), tf.keras.layers.Dense(units=5),]) model_5.summary() ``` Hyperparam2: Regularization with L2 This model aims to reduce overfitting by adding L2 regularization to the convolutional and dense layers. L2 regularization penalizes large weights in the network, encouraging the model to keep the weights small and thus reducing overfitting. ``` # Build third CNN Model to prevent overffiting: by add regularization-L2 model_3 = tf.keras.Sequential([tf.keras.layers.Rescaling(1./255, input_shape=(224, 224, 3)), tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.MaxPool2D(pool_size=2, strides=2), tf.keras.layers.Flatten(), tf.keras.layers.Dense(units=300, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)), tf.keras.layers.Dropout(0.5), tf.keras.layers.Dense(units=5),]) model_3.summary() ```

SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

<table>
<tr>
<td>Model 2</td>
<td>

ALEXNET;

Hyperparam1:

Increase Epoch from 3 to 5

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 conv2d (Conv2D)             (None, 58, 58, 96)        34944

 batch_normalization (Batch  (None, 58, 58, 96)        384
 Normalization)

 max_pooling2d (MaxPooling2  (None, 28, 28, 96)        0
 D)

 conv2d_1 (Conv2D)           (None, 24, 24, 256)       614656

 batch_normalization_1 (Bat  (None, 24, 24, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 256)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 9, 9, 384)         885120

 conv2d_3 (Conv2D)           (None, 7, 7, 384)         1327488

 conv2d_4 (Conv2D)           (None, 5, 5, 256)         884992

 max_pooling2d_2 (MaxPoolin  (None, 2, 2, 256)         0
 g2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 4096)              4198400

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 5)                 20485
```

```python
# Assuming 'Alex_model' is the history object from your model training
training_acc_alex = Alex_model.history['accuracy']
val_acc_alex = Alex_model.history['val_accuracy']

best_training_acc = max(training_acc_alex)
best_val_acc = max(val_acc_alex)

# Converting to percentage and rounding off to 2 decimal places
best_training_acc_percentage = round(best_training_acc * 100, 2)
best_val_acc_percentage = round(best_val_acc * 100, 2)

print("Best Training Accuracy: ", best_training_acc_percentage, "%")
print("Best Validation Accuracy: ", best_val_acc_percentage, "%")
```

```
Best Training Accuracy:  97.61 %
Best Validation Accuracy:  97.99 %
```

</td>
</tr>
<tr>
<td>Model 3</td>
<td>

CGGNET;

Hyperparam1:

</td>
</tr>
</table>

Increase Epoch 3 to 5

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 240, 240, 3)]     0

 conv2d_5 (Conv2D)           (None, 238, 238, 64)      1792

 conv2d_6 (Conv2D)           (None, 236, 236, 64)      36928

 batch_normalization_2 (Bat  (None, 236, 236, 64)      256
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 118, 118, 64)      0
 g2D)

 conv2d_7 (Conv2D)           (None, 116, 116, 128)     73856

 conv2d_8 (Conv2D)           (None, 114, 114, 128)     147584

 batch_normalization_3 (Bat  (None, 114, 114, 128)     512
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 57, 57, 128)       0
 g2D)

 conv2d_9 (Conv2D)           (None, 55, 55, 256)       295168

 conv2d_10 (Conv2D)          (None, 53, 53, 256)       590080
```

```python
# Assuming 'VGG_model' is the history object from your model training
training_acc_alex = VGG_model.history['accuracy']
val_acc_alex = VGG_model.history['val_accuracy']

best_training_acc = max(training_acc_alex)
best_val_acc = max(val_acc_alex)

# Converting to percentage and rounding off to 2 decimal places
best_training_acc_percentage = round(best_training_acc * 100, 2)
best_val_acc_percentage = round(best_val_acc * 100, 2)

print("Best Training Accuracy: ", best_training_acc_percentage, "%")
print("Best Validation Accuracy: ", best_val_acc_percentage,"%")
```

```
Best Training Accuracy:  98.39 %
Best Validation Accuracy:  98.66 %
```

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Model 1 MobileNet CNN	The MobileNet model has demonstrated higher accuracy compared to other models like VGG-Net and AlexNet, which is why I selected this project.