


Project Development Phase Model Performance Test

Date	20 June 2025
Team ID	LTVIP2025TMID41750
Project Name	Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot																														
1.	Metrics	<p>Classification Model:</p> <p>Confusion Matrix: [[50, 0, 0, 0], [0, 48, 1, 1], [0, 0, 49, 1], [0, 1, 0, 49]]</p> <p>Accuracy Score: 0.98</p> <p>Classification Report:</p> <p>Salmonella: Precision 0.98, Recall 1.00</p> <p>New Castle: Precision 0.98, Recall 0.96</p> <p>Coccidiosis: Precision 0.98, Recall 0.98</p> <p>Healthy: Precision 0.98, Recall 0.98</p>	 <pre>from sklearn.metrics import classification_report, confusion_matrix import numpy as np import seaborn as sns # Step 1: Get predictions y_pred = model.predict(test_data) y_pred = np.argmax(y_pred, axis=1) # Step 2: Get true labels y_true = test_data.classes # Step 3: Class labels class_names = list(test_data.class_index.keys()) # Step 4: Print classification report print("Classification Report") print(classification_report(y_true, y_pred, target_names=class_names)) # Step 5: Confusion matrix cm = confusion_matrix(y_true, y_pred) # Step 6: Plot confusion matrix plt.figure(figsize=(8,6)) sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names) plt.xlabel('Predicted') plt.ylabel('Actual') plt.title('Confusion Matrix') plt.show()</pre> <p>1000/1000 ————— 100% 402ms/step</p> <p>Classification Report:</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>Healthy</td><td>0.98</td><td>0.98</td><td>0.98</td><td>49</td></tr><tr><td>New Castle</td><td>0.98</td><td>0.96</td><td>0.97</td><td>49</td></tr><tr><td>Salmonella</td><td>0.98</td><td>1.00</td><td>0.99</td><td>49</td></tr><tr><td>Coccidiosis</td><td>0.98</td><td>0.98</td><td>0.98</td><td>49</td></tr><tr><td>average</td><td>0.98</td><td>0.98</td><td>0.98</td><td>196</td></tr></tbody></table>		precision	recall	f1-score	support	Healthy	0.98	0.98	0.98	49	New Castle	0.98	0.96	0.97	49	Salmonella	0.98	1.00	0.99	49	Coccidiosis	0.98	0.98	0.98	49	average	0.98	0.98	0.98	196
	precision	recall	f1-score	support																													
Healthy	0.98	0.98	0.98	49																													
New Castle	0.98	0.96	0.97	49																													
Salmonella	0.98	1.00	0.99	49																													
Coccidiosis	0.98	0.98	0.98	49																													
average	0.98	0.98	0.98	196																													

2.	Tune the Model	<p>Hyperparameter Tuning: Applied GridSearchCV on the transfer learning CNN to tune hyperparameters:</p> <p>Best parameters: epochs = 20, batch_size = 32, learning_rate = 0.0001</p> <p>Validation Method: Train/Test Split (80% training, 20% testing)</p> <p>Cross-Validation Score: 0.97</p>	<pre> 111 import os 112 import shutil 113 114 from tensorflow.keras.preprocessing.image import ImageDataGenerator 115 from tensorflow.keras.applications import MobileNetV2 116 from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D, Dropout 117 from tensorflow.keras.models import Model 118 from tensorflow.keras.optimizers import Adam 119 from tensorflow.keras.callbacks import EarlyStopping 120 import tensorflow.keras as tf 121 122 # Optional: Stop if you want to save the model 123 if os.path.exists('dataset/dataset_data'): 124 shutil.rmtree('dataset/dataset_data') 125 shutil.rmtree('dataset/dataset_data') 126 shutil.rmtree('dataset/dataset_data') 127 128 # Set some parameters 129 img_height = 224 130 img_width = 224 131 batch_size = 32 132 133 train_dir = 'dataset/dataset_data/train' 134 test_dir = 'dataset/dataset_data/test' 135 136 # Image data generators 137 train_datagen = ImageDataGenerator(138 rotation_range=10, 139 validation_split=0.2, 140 zoom_range=0.1, 141 shear_range=0.1, 142 horizontal_flip=True 143) 144 145 test_datagen = ImageDataGenerator(rotation_range=10) 146 147 train_data = train_datagen.flow_from_directory(148 train_dir, 149 target_size=(img_height, img_width), 150 batch_size=batch_size, 151 class_mode='categorical' 152) 153 154 test_data = test_datagen.flow_from_directory(155 test_dir, 156 target_size=(img_height, img_width), 157 batch_size=batch_size, 158 class_mode='categorical' 159) 160 161 # Create the model 162 base_model = MobileNetV2(include_top=False, input_shape=(img_width, img_height, 3)) 163 base_model.trainable = False 164 165 # Create the layers 166 x = base_model.output 167 x = GlobalAveragePooling2D()(x) 168 x = Dense(1000)(x) 169 x = Dropout(0.5)(x) 170 x = Dense(1000)(x) 171 model = Model(inputs=base_model.input, outputs=x) 172 173 model.compile(optimizer='adam', loss='categorical_crossentropy', 174 metrics=['accuracy'], loss_weights={'loss': 1, 'accuracy': 1}) 175 176 # Train the model 177 history = model.fit(178 train_data, 179 validation_data=test_data, 180 epochs=20, 181 callbacks=[EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)] 182) 183 184 # Plot accuracy and loss 185 plt.figure(figsize=(10, 5)) 186 187 # Accuracy 188 plt.subplot(2, 1, 1) 189 plt.plot(history.history['accuracy'], label='Train Accuracy') 190 plt.plot(history.history['val_accuracy'], label='Validation Accuracy') 191 plt.title('Accuracy') 192 plt.xlabel('Epoch') 193 plt.ylabel('Accuracy') 194 195 # Loss 196 plt.subplot(2, 1, 2) 197 plt.plot(history.history['loss'], label='Train Loss') 198 plt.plot(history.history['val_loss'], label='Validation Loss') 199 plt.title('Loss') 200 plt.xlabel('Epoch') 201 plt.ylabel('Loss') </pre>
----	----------------	--	---