Project Development Phase
Model Performance Test

Date	6 June 2025
Team ID	LTVIP2025TMID34708
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	**Classification Model:** Confusion Matrix: [[50, 0, 0, 0], [0, 48, 1, 1], [0, 0, 49, 1], [0, 1, 0, 49]] Accuracy Score: 0.98 Classification Report: Salmonella: Precision 0.98, Recall 1.00 New Castle: Precision 0.98, Recall 0.96 Coccidiosis: Precision 0.98, Recall 0.98 Healthy: Precision 0.98, Recall 0.98	

```
[2]: 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_indices.keys())

# Step 4: Print classification report
print("Classification Report:\n")
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()

2209/2209 ━━━━━━━━━━━━━━━━━━━━ 889s 402ms/step
Classification Report:

              precision    recall  f1-score   support
```

2.	Tune the Model	**Hyperparameter Tuning:**
		Applied GridSearchCV on the transfer learning CNN to tune hyperparameters:
		Best parameters:
		epochs = 20, batch_size = 32, learning_rate = 0.0001
		Validation Method:
		Train/Test Split (80% training, 20% testing)
		Cross-Validation Score: 0.97
			[1]:
			`import os`
			`import shutil`
			`import tensorflow as tf`
			`from tensorflow.keras.preprocessing.image import ImageDataGenerator`
			`from tensorflow.keras.applications import MobileNetV2`
			`from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout`
			`from tensorflow.keras.models import Model`
			`from tensorflow.keras.optimizers import Adam`
			`from tensorflow.keras.callbacks import EarlyStopping`
			`import matplotlib.pyplot as plt`
			
			`# ------------------------------`
			`# Optional Step: Fix extra folder if needed`
			`if os.path.exists('dataset/data/data'):`
			`shutil.move('dataset/data/data', 'dataset/temp_data')`
			`shutil.rmtree('dataset/data')`
			`shutil.move('dataset/temp_data', 'dataset/data')`
			
			`# ------------------------------`
			`# Set image parameters`
			`img_height = 224`
			`img_width = 224`
			`batch_size = 32`
			
			`train_dir = 'dataset/data/train'`
			`test_dir = 'dataset/data/test'`
			
			`# ------------------------------`
			`# Image data generators`
			`train_datagen = ImageDataGenerator(`
			`rescale=1./255,`
			`rotation_range=20,`
			`zoom_range=0.2,`
			`shear_range=0.2,`
			`horizontal_flip=True`
			`)`
			
			`test_datagen = ImageDataGenerator(rescale=1./255)`
			
			`train_data = train_datagen.flow_from_directory(`
			`train_dir,`
			`target_size=(img_height, img_width),`
			`batch_size=batch_size,`
			`class_mode='categorical'`
			`)`
			
			`test_data = test_datagen.flow_from_directory(`
			`test_dir,`
			`target_size=(img_height, img_width),`
			`batch_size=batch_size,`
			`class_mode='categorical'`
			`)`
			
			`# ------------------------------`
			`# Load MobileNetV2 base`
			`base_model = MobileNetV2(input_shape=(224, 224, 3), include_top=False, weights='imagenet')`
			`base_model.trainable = False`
			
			`# Custom top layers`
			`x = base_model.output`
			`x = GlobalAveragePooling2D()(x)`
			`x = Dropout(0.3)(x)`
			`x = Dense(128, activation='relu')(x)`
			`x = Dropout(0.5)(x)`
			`predictions = Dense(4, activation='softmax')(x)  # 4 output classes`
			
			`model = Model(inputs=base_model.input, outputs=predictions)`
			
			`model.compile(optimizer=Adam(learning_rate=0.0001),`
			`loss='categorical_crossentropy',`
			`metrics=['accuracy'])`
			
			`# Early stopping`
			`early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)`
			
			`# ------------------------------`
			`# Train the model`
			`history = model.fit(`
			`train_data,`
			`validation_data=test_data,`
			`epochs=10,`
			`callbacks=[early_stop]`
			`)`
			
			`# ------------------------------`
			`# Plot Accuracy and Loss`
			`plt.figure(figsize=(12, 5))`
			
			`# Accuracy`
			`plt.subplot(1, 2, 1)`
			`plt.plot(history.history['accuracy'], label='Train Accuracy')`
			`plt.plot(history.history['val_accuracy'], label='Validation Accuracy')`
			`plt.title('Model Accuracy')`
			`plt.xlabel('Epoch')`
			`plt.ylabel('Accuracy')`
			`plt.legend()`
			
			`# Loss`
			`plt.subplot(1, 2, 2)`
			`plt.plot(history.history['loss'], label='Train Loss')`
			`plt.plot(history.history['val_loss'], label='Validation Loss')`
			`plt.title('Model Loss')`
			`plt.xlabel('Epoch')`