COSC 6373 - HW2-ICA - Minh Nguyen #2069407

In [77]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow import keras
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras import layers, models
from sklearn.metrics import confusion_matrix, classification_report
```

In [78]:

```
# Define dataset paths
train_dir = "train"
test_dir = "test"

# Define parameters
IMG_SIZE = (256, 256)
BATCH_SIZE = 32

# Load training and validation data
train_data = tf.keras.utils.image_dataset_from_directory(
    train_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    validation_split=0.1,
    subset="training",
    seed=42
)

val_data = tf.keras.utils.image_dataset_from_directory(
    train_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    validation_split=0.1,
    subset="validation",
    seed=42
)

# Load test dataset
test_data = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    shuffle=False
)
```

```
Found 360 files belonging to 2 classes.
Using 324 files for training.
Found 360 files belonging to 2 classes.
Using 36 files for validation.
Found 40 files belonging to 2 classes.
```

In [79]:
```python
# Apply ResNet50 preprocessing
train_data = train_data.map(lambda x, y: (preprocess_input(x), y))
val_data = val_data.map(lambda x, y: (preprocess_input(x), y))

# Load ResNet50 as the base model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(25
base_model.trainable = False  # Freeze base model initially
```

In [80]:
```python
# Add custom layers on top
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.5),  # Prevent overfitting
    layers.Dense(1, activation='sigmoid')  # Binary classification
])
```

In [81]:
```python
# # Compile model
# model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
#               loss='binary_crossentropy',
#               metrics=['accuracy'])

# # Train initial model
# history = model.fit(train_data, validation_data=val_data, epochs=5)
```

In [82]:
```python
# Unfreeze some layers for fine-tuning
base_model.trainable = True

# Unfreeze the last 25 layers to fine-tune the model
for layer in base_model.layers[:-25]:
    layer.trainable = False

# Recompile model with lower learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Model summary
model.summary()

# Fine-tune model
history_fine = model.fit(train_data, validation_data=val_data, epochs=15)
```

Model: "sequential_11"

Layer (type)	Output Shape	Par
resnet50 (Functional)	(None, 8, 8, 2048)	23,587
global_average_pooling2d_11 (GlobalAveragePooling2D)	(None, 2048)	
dropout_1 (Dropout)	(None, 2048)	
dense_11 (Dense)	(None, 1)	2

Total params: 23,589,761 (89.99 MB)

Trainable params: 9,992,193 (38.12 MB)

Non-trainable params: 13,597,568 (51.87 MB)

```
Epoch 1/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 41s 3s/step - accuracy: 0.4814 - loss: 0.8856 - v
al_accuracy: 0.6944 - val_loss: 0.6365
Epoch 2/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.6177 - loss: 0.6515 - v
al_accuracy: 0.7500 - val_loss: 0.5650
Epoch 3/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 33s 3s/step - accuracy: 0.7050 - loss: 0.5537 - v
al_accuracy: 0.7778 - val_loss: 0.5008
Epoch 4/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 31s 3s/step - accuracy: 0.7735 - loss: 0.4848 - v
al_accuracy: 0.8333 - val_loss: 0.4409
Epoch 5/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.7971 - loss: 0.4448 - v
al_accuracy: 0.8333 - val_loss: 0.3839
Epoch 6/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.8708 - loss: 0.3715 - v
al_accuracy: 0.8611 - val_loss: 0.3364
Epoch 7/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 29s 3s/step - accuracy: 0.8972 - loss: 0.3083 - v
al_accuracy: 0.9444 - val_loss: 0.2897
Epoch 8/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 29s 3s/step - accuracy: 0.8936 - loss: 0.3145 - v
al_accuracy: 1.0000 - val_loss: 0.2528
Epoch 9/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.9283 - loss: 0.2727 - v
al_accuracy: 1.0000 - val_loss: 0.2190
Epoch 10/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.9460 - loss: 0.2056 - v
al_accuracy: 1.0000 - val_loss: 0.1859
Epoch 11/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 30s 3s/step - accuracy: 0.9680 - loss: 0.1910 - v
al_accuracy: 1.0000 - val_loss: 0.1576
Epoch 12/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 28s 3s/step - accuracy: 0.9782 - loss: 0.1586 - v
al_accuracy: 1.0000 - val_loss: 0.1372
Epoch 13/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.9674 - loss: 0.1441 - v
al_accuracy: 1.0000 - val_loss: 0.1197
Epoch 14/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 15s 1s/step - accuracy: 0.9954 - loss: 0.1036 - v
al_accuracy: 1.0000 - val_loss: 0.1088
Epoch 15/15
11/11 ━━━━━━━━━━━━━━━━━━━━ 15s 1s/step - accuracy: 0.9883 - loss: 0.1026 - v
al_accuracy: 1.0000 - val_loss: 0.0968
```

In [83]:

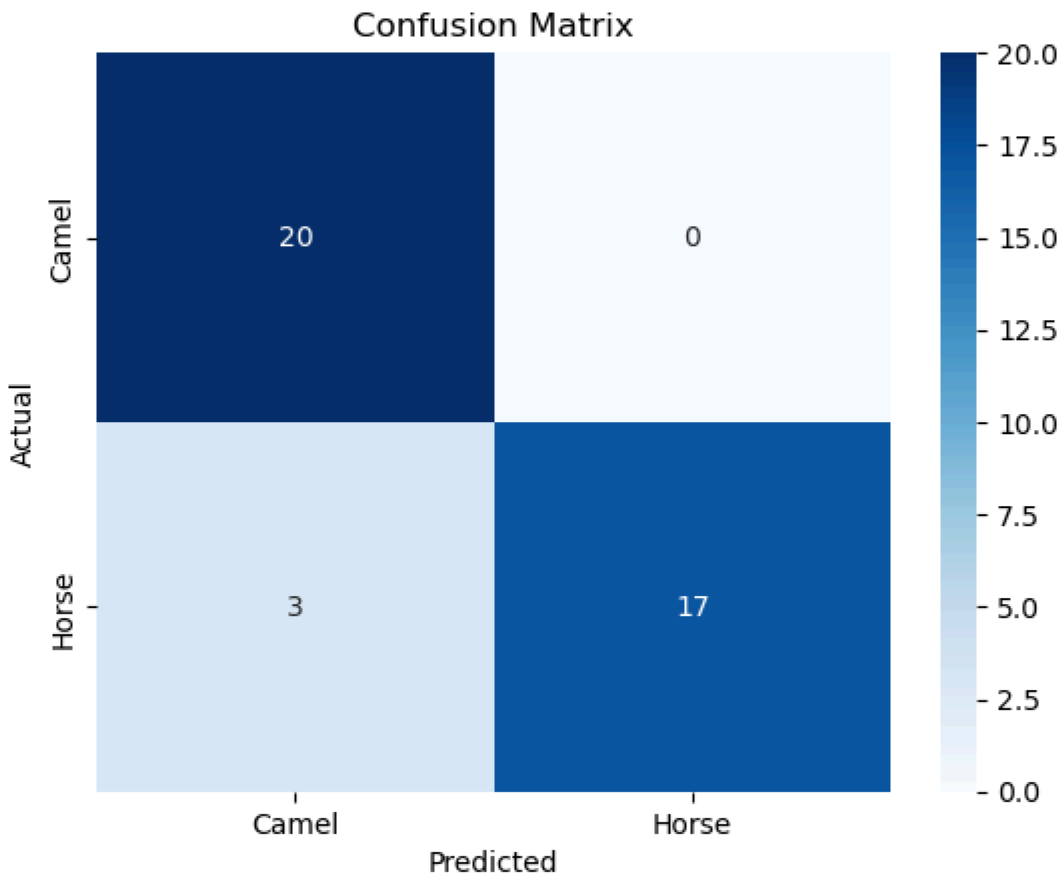
```python
# Evaluate model on test data
test_loss, test_accuracy = model.evaluate(test_data)
print(f"Test Accuracy: {test_accuracy:.4f}")

# Generate confusion matrix
y_pred = (model.predict(test_data) > 0.5).astype(int)
y_true = np.concatenate([y for x, y in test_data], axis=0)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 283ms/step - accuracy: 0.9500 - loss: 0.2286
Test Accuracy: 0.9250
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 1s/step
```

In [84]:
```python
conf_matrix = confusion_matrix(y_true, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Ca
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Classification report
print("\nClassification Report:\n", classification_report(y_true, y_pred, ta
```




```
Classification Report:
               precision    recall  f1-score   support

       Camel       0.87      1.00      0.93        20
       Horse       1.00      0.85      0.92        20

    accuracy                           0.93        40
   macro avg       0.93      0.93      0.92        40
weighted avg       0.93      0.93      0.92        40
```

Summary:

- After several attempts with different number of unfreezed layers and number of epochs, I ended up with the following configuration:
    - Unfreezed the last 25 layers
    - Epochs: 15
    - Learning rate: 1e-5
- It obtained an accuracy of 0.9500 and a loss of 0.2286 on the test dataset.
- The confusion matrix shows that the model performs pretty well on the test dataset, with only 3 misclassified images out of 40 (3 camels were misclassified as horses).