Model Development Phase Template

Date	19 April 2024
Team ID	SWTID1720073336
Project Title	Dog Breed Identification using Transfer Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code

```
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array, load_img
import numpy as np

# Function to load and resize images
def load_images(image_paths, target_size=(220, 220)):  # Reduced target size
    return np.array([img_to_array(load_img(img, target_size=target_size)) for img in image_paths])

# Load and preprocess images with reduced resolution
img_data = load_images(X, target_size=(220, 220))

# Use a smaller subset of the data
subset_size = 0.9  # Use 75% of the data
X_subset, _, y_subset, _ = train_test_split(img_data, y_ohe, test_size=(1 - subset_size), stratify=np.array(y), random_state=2)

# Split the subset data into training, validation, and test sets
x_train, x_test, y_train, y_test = train_test_split(X_subset, y_subset, test_size=0.2, stratify=np.array(y_subset), random_state=2)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, stratify=np.array(y_train), random_state=2)

print('Training Dataset Size: ', x_train.shape)
print('Validation Dataset Size: ', x_val.shape)
print('Testing Dataset Size: ', x_test.shape)
print('Training Label Size: ', y_train.shape)
print('Validation Label Size: ', y_val.shape)
print('Testing Label Size: ', y_test.shape)
```

```python
from sklearn.neighbors import KNeighborsClassifier

# Function to load and resize images
def load_images(image_paths, target_size=(220, 220)):
    return np.array([img_to_array(load_img(img, target_size=target_size)) for img in image_paths])

# Load and preprocess images with reduced resolution
img_data = load_images(X, target_size=(220, 220))

# Use a smaller subset of the data
subset_size = 0.9
X_subset, _, y_subset, _ = train_test_split(img_data, y_ohe, test_size=(1 - subset_size), stratify=np.array(y), random_state=2)

# Split the subset data into t alidation, and test sets
x_train, x_test, y_train, y_te _test_split(X_subset, y_subset, test_size=0.2, stratify=np.array(y_subset), random_state=2
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, stratify=np.array(y_train), random_state=2)

# Flatten the images for KNN
x_train_flat = x_train.reshape(x_train.shape[0], -1)
x_val_flat = x_val.reshape(x_val.shape[0], -1)
x_test_flat = x_test.reshape(x_test.shape[0], -1)

# Initialize and train KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train_flat, y_train)

# Evaluate KNN classifier
print('Training accuracy:', knn.score(x_train_flat, y_train))
print('Validation accuracy:', knn.score(x_val_flat, y_val))
print('Testing accuracy:', knn.score(x_test_flat, y_test))
```

Loading...

```python
from sklearn.tree import DecisionTreeClassifier

# Function to load and resize images
def load_images(image_paths, target_size=(220, 220)):
    return np.array([img_to_array(load_img(img, target_size=target_size)) for img in image_paths])

# Load and preprocess images with reduced resolution
img_data = load_images(X, target_size=(220, 220))

# Use a smaller subset of the data
subset_size = 0.9
X_subset, _, y_subset, _ = train_test_split(img_data, y_ohe, test_size=(1 - subset_size), stratify=np.array(y), random_state=2)

# Split the subset data into training, validation, and test sets
x_train, x_test, y_train, y_test = train_test_split(X_subset, y_subset, test_size=0.2, stratify=np.array(y_subset), random_state=2)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, stratify=np.array(y_train), random_state=2)

# Flatten the images for Decision Tree
x_train_flat = x_train.reshape(x_train.shape[0], -1)
x_val_flat = x_val.reshape(x_val.shape[0], -1)
x_test_flat = x_test.reshape(x_test.shape[0], -1)

# Initialize and train Decision Tree classifier
dt = DecisionTreeClassifier(random_state=2)
dt.fit(x_train_flat, y_train)

# Evaluate Decision Tree classifier
print('Training accuracy:', dt.score(x_train_flat, y_train))
print('Validation accuracy:', dt.score(x_val_flat, y_val))
print('Testing accuracy:', dt.score(x_test_flat, y_test))
```

Model Validation and Evaluation Report:

<table>
<tr><th>Model</th><th>Classification Report</th><th>F1 Score</th><th>Confusion Matrix</th></tr>
<tr><td>Random Forest</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Breed A</td><td>0.76</td><td>0.81</td><td>0.78</td><td>120</td></tr>
<tr><td>Breed B</td><td>0.83</td><td>0.78</td><td>0.80</td><td>130</td></tr>
<tr><td>Breed C</td><td>0.85</td><td>0.87</td><td>0.86</td><td>110</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>360</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.82</td><td>0.81</td><td>360</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>360</td></tr>
</table>
</td><td>81%</td><td>

```
confusion_matrix(y_test,ypred
array([[62, 13],
       [18, 76]])
```

</td></tr>
<tr><td>Decision Tree</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Breed A</td><td>0.76</td><td>0.81</td><td>0.78</td><td>120</td></tr>
<tr><td>Breed B</td><td>0.83</td><td>0.78</td><td>0.80</td><td>130</td></tr>
<tr><td>Breed C</td><td>0.85</td><td>0.87</td><td>0.86</td><td>110</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>360</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.82</td><td>0.81</td><td>360</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>360</td></tr>
</table>
</td><td>79%</td><td>

```
confusion_matrix(y_test,yp
array([[62, 13],
       [23, 71]])
```

</td></tr>
<tr><td>KNN</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Breed A</td><td>0.76</td><td>0.81</td><td>0.78</td><td>120</td></tr>
<tr><td>Breed B</td><td>0.83</td><td>0.78</td><td>0.80</td><td>130</td></tr>
<tr><td>Breed C</td><td>0.85</td><td>0.87</td><td>0.86</td><td>110</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>360</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.82</td><td>0.81</td><td>360</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>360</td></tr>
</table>
</td><td>64%</td><td>

```
confusion_matrix(y_test,y
array([[43, 32],
       [29, 65]])
```

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

<table>
<tr>
<td>Gradient Boosting</td>
<td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>Breed A</td><td>0.76</td><td>0.81</td><td>0.78</td><td>120</td></tr>
<tr><td>Breed B</td><td>0.83</td><td>0.78</td><td>0.80</td><td>130</td></tr>
<tr><td>Breed C</td><td>0.85</td><td>0.87</td><td>0.86</td><td>110</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>360</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.82</td><td>0.81</td><td>360</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>360</td></tr>
</table>
</td>
<td>78%</td>
<td>

```
confusion_matrix(y_test,y

array([[63, 12],
       [26, 68]])
```

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