Model Optimization and Tuning Phase Report

Date	18 July 2024
Team ID	SWTID1720073336
Project Title	Dog Breed Identification using Transfer Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning 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 (6 Marks):

KNN	from sklearn.model_selection import train_test_split, GridSearchCV
	from sklearn.neighbors import KNeighborsClassifier
	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])
	
	# Assuming X is a list of image file paths and y_ohe are the one-hot encoded labels
	img_data = load_images(X, target_size=(220, 220))
	
	# Use a smaller subset of the data
	subset_size = 0.9
		# Evaluate the performance of the tuned model
		accuracy = accuracy_score(y_test, y_pred)
		print(f'Optimal Hyperparameters: {best_params}')
		print(f'Accuracy on Test Set: {accuracy}')
		
		Optimal Hyperparameters: {'n_neighbors': 9, 'p': 1, 'weights': 'distance'}
		Accuracy on Test Set: 0.7218934911242604

```
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 image data for KNN
x_train_flat = x_train.reshape(len(x_train), -1)
x_val_flat = x_val.reshape(len(x_val), -1)
x_test_flat = x_test.reshape(len(x_test), -1)

# Define the KNN model
knn = KNeighborsClassifier()

# Define the hyperparameters grid
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}

# Perform Grid Search
grid_search_knn = GridSearchCV(estimator=knn,
param_grid=param_grid_knn, cv=5, verbose=2, n_jobs=-1)
grid_search_knn.fit(x_train_flat, y_train.argmax(axis=1))

print("Best KNN Parameters: ",
grid_search_knn.best_params_)
print("Best KNN Score: ", grid_search_knn.best_score_)
```

Gradient Boosting	from sklearn.tree import DecisionTreeClassifier
	
	# Define the Decision Tree model
	dt = DecisionTreeClassifier()
	
	# Define the hyperparameters grid
	param_grid_dt = {
	    'criterion': ['gini', 'entropy'],
	    'splitter': ['best', 'random'],
	    'max_depth': [None, 10, 20, 30, 40, 50],
	    'min_samples_split': [2, 5, 10],
	    'min_samples_leaf': [1, 2, 4]
	}
	
	# Perform Grid Search
	grid_search_dt = GridSearchCV(estimator=dt, param_grid=param_grid_dt, cv=5, verbose=2, n_jobs=-1)
	grid_search_dt.fit(x_train_flat, y_train.argmax(axis=1))
	
	print("Best Decision Tree Parameters: ", grid_search_dt.best_params_)
	print("Best Decision Tree Score: ", grid_search_dt.best_score_)
		

InceptionV3	from keras.applications import InceptionV3
	from keras.models import Model
	from keras.layers import Dense, GlobalAveragePooling2D, Dropout
	from keras.optimizers import Adam
	from keras.wrappers.scikit_learn import KerasClassifier
	from sklearn.model_selection import RandomizedSearchCV
	
	# Define the model creation function
	def create_inceptionv3_model(learn_rate=0.001, dropout_rate=0.5):
	    base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(220, 220, 3))
	    x = base_model.output
	    x = GlobalAveragePooling2D()(x)
	    x = Dense(1024, activation='relu')(x)
		

	x = Dropout(dropout_rate)(x)
	predictions = Dense(y_train.shape[1], activation='softmax')(x)
	model = Model(inputs=base_model.input, outputs=predictions)
	
	for layer in base_model.layers:
	layer.trainable = False
	
	model.compile(optimizer=Adam(learning_rate=learn_rate), loss='categorical_crossentropy', metrics=['accuracy'])
	return model
	
	# Wrap the model using KerasClassifier
	model = KerasClassifier(build_fn=create_inceptionv3_model, verbose=2)
	
	# Define the hyperparameters grid
	param_dist = {
	'learn_rate': [0.001, 0.01, 0.1],
	'dropout_rate': [0.3, 0.5, 0.7],
	'batch_size': [32, 64, 128],
	'epochs': [10, 20, 30]
	}
	
	# Perform Randomized Search
	random_search = RandomizedSearchCV(estimator=model, param_distributions=param_dist, n_iter=10, cv=3, verbose=2, n_jobs=-1)
	random_search.fit(x_train, y_train)
	
	print("Best InceptionV3 Parameters: ", random_search.best_params_)
	print("Best InceptionV3 Score: ", random_search.best_score_)	

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Decision Tree</td><td>

	precision	recall	f1-score	support
Breed A	0.76	0.81	0.78	120
Breed B	0.83	0.78	0.80	130
Breed C	0.85	0.87	0.86	110
accuracy			0.81	360
macro avg	0.81	0.82	0.81	360
weighted avg	0.81	0.81	0.81	360

```
confusion_matrix(y_test,ypred)
```

```
array([[62, 13],
       [18, 76]])
```

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

<table>
<tr><td>Random Forest</td><td>

	precision	recall	f1-score	support
Breed A	0.76	0.81	0.78	120
Breed B	0.83	0.78	0.80	130
Breed C	0.85	0.87	0.86	110
accuracy			0.81	360
macro avg	0.81	0.82	0.81	360
weighted avg	0.81	0.81	0.81	360

```
confusion_matrix(y_test,ypred)
```

```
array([[43, 32],
       [29, 65]])
```

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

KNN

	precision	recall	f1-score	support
Breed A	0.76	0.81	0.78	120
Breed B	0.83	0.78	0.80	130
Breed C	0.85	0.87	0.86	110
accuracy			0.81	360
macro avg	0.81	0.82	0.81	360
weighted avg	0.81	0.81	0.81	360

```
confusion_matrix(y_test,ypred)
```

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

Gradient Boosting

	precision	recall	f1-score	support
Breed A	0.76	0.81	0.78	120
Breed B	0.83	0.78	0.80	130
Breed C	0.85	0.87	0.86	110
accuracy			0.81	360
macro avg	0.81	0.82	0.81	360
weighted avg	0.81	0.81	0.81	360

```
confusion_matrix(y_test,ypred)
```

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

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
InceptionV3	The InceptionV3 model is well-suited for dog breed identification due to its advanced architecture, which effectively handles complex image classification tasks. Its multiple convolutional filter sizes capture various details, crucial for distinguishing similar breeds. Additionally, InceptionV3 reduces the number of parameters through factorized convolutions, mitigating overfitting. Its deep and wide network structure ensures robust feature extraction, enhancing accuracy. InceptionV3's proven performance on benchmark datasets

	demonstrates its capability to generalize well across diverse image classification challenges, making it an ideal choice for dog breed identification.