In [1]:
```python
#imported libs
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.utils import get_file
import os
```

In [ ]:
```python
#dataset_dir  = r"E:\APPS\PythonDataSets\caltech\caltech-101\101_ObjectCategories\1
dataset_dir = r"E:\APPS\PythonDataSets\caltech\caltech-101\101_ObjectCategories\fil



# Parameters
batch_size = 32
image_size = (64, 128)

# Load the training and validation datasets
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    dataset_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=image_size,
    batch_size=batch_size
)

val_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    dataset_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=image_size,
    batch_size=batch_size
)


def dataset_to_numpy(dataset):
    """
    Convert a tf.data.Dataset into NumPy arrays for features and labels.
    Args:
        dataset: A tf.data.Dataset object.
    Returns:
        X: Numpy array of features (images).
        y: Numpy array of labels.
    """
    X = []
    y = []
    for images, labels in dataset:
        X.append(images.numpy())
        y.append(labels.numpy())
    return np.concatenate(X, axis=0), np.concatenate(y, axis=0)

# Convert the train and validation datasets to NumPy arrays
```

```
X_train, y_train = dataset_to_numpy(train_dataset)
X_test, y_test = dataset_to_numpy(val_dataset)

X_test =[cv2.resize(img.astype(np.uint8), (64, 128)) for img in X_test]
X_train=[cv2.resize(img.astype(np.uint8), (64, 128)) for img in X_train]
```

```
Found 9145 files belonging to 101 classes.
Using 7316 files for training.
Found 9145 files belonging to 101 classes.
Using 1829 files for validation.
```

In [37]:
```
#SVM def
import cv2
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, accuracy_score

def SVM(X_train, y_train,X_test,y_test):

    # Step 4: Train LinearSVC with Grid Search for Hyperparameter Tuning
    param_grid = {
        'C': [0.1,10],  # Regularization parameter
    }
    svm = LinearSVC(max_iter=10000)  # Linear SVM for multiclass classification dua
    grid_search = GridSearchCV(svm, param_grid, cv=3, verbose=2, n_jobs=-1)
    grid_search.fit(X_train, y_train)

    # Best Model
    best_svm = grid_search.best_estimator_
    print("Best Parameters:", grid_search.best_params_)

    # Step 5: Evaluate Model
    y_pred = best_svm.predict(X_test)

    # Classification Report
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))#,zero_division=1
```

In [ ]:
```
#Color Histogran Extraction def
def extract_color_histogram(image, bins=(8, 8, 8)):
    # Calculate the 3D histogram for the HSV channels
    hist = cv2.calcHist([image], [0, 1, 2], None, bins, [0, 256, 0, 256, 0, 256])
    # Normalize the histogram to ensure invariance to lighting changes
    hist = cv2.normalize(hist, hist).flatten()

    return hist
```

In [5]:
```
#HOG def
def extract_hog_features(image):
    # HOG parameters
    winSize = (64, 128)
    blockSize = (16, 16)
    blockStride = (8, 8)
    cellSize = (8, 8)
    nbins = 9
```

```
    hog = cv2.HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins)
    hog_features = hog.compute(image)

    return hog_features
```

In [13]:
```
#LBP def
from skimage.feature import local_binary_pattern

def extract_lbp_features(image, num_points=32, radius=8):
    gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    grid_size = (8, 8)  # Divide image into a 8x8 grid for histograms
     # Compute LBP
    lbp = local_binary_pattern(gray_img, num_points, radius, method="uniform")
    h, w = lbp.shape

    # Divide the image into grids and compute histograms
    grid_h, grid_w = h // grid_size[0], w // grid_size[1]
    histograms = []

    for i in range(grid_size[0]):
        for j in range(grid_size[1]):
            grid = lbp[i * grid_h:(i + 1) * grid_h, j * grid_w:(j + 1) * grid_w]
            hist, _ = np.histogram(grid, bins=np.arange(0, num_points + 3), density
            histograms.append(hist)

    return np.concatenate(histograms)
```

In [14]:
```
# Step 1: Extract LBP features for train and test
lbp_features_train = np.array([extract_lbp_features(image) for image in X_train])
lbp_features_test  = np.array([extract_lbp_features(image) for image in X_test])
```

In [15]:
```
# Step 2: Extract HOG features for train and test
hog_features_train = np.array([extract_hog_features(image) for image in X_train])
hog_features_test  = np.array([extract_hog_features(image) for image in X_test])
```

In [24]:
```
# Step 3: Extract Color Histogram features for train and test
clhg_features_train = np.array([extract_color_histogram(image) for image in X_train
clhg_features_test  = np.array([extract_color_histogram(image) for image in X_test]
```

In [22]:
```
SVM(lbp_features_train,y_train,lbp_features_test,y_test)
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Best Parameters: {'C': 1}
Accuracy: 0.47785675232367414
```

	precision	recall	f1-score	support
0	0.15	0.18	0.17	92
1	0.83	0.95	0.89	175
2	0.70	0.95	0.81	44
3	0.89	0.97	0.93	176
4	0.53	0.73	0.62	11
5	0.80	0.93	0.86	162
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	6
8	0.00	0.00	0.00	10
9	0.00	0.00	0.00	17
10	0.12	0.14	0.13	7
11	1.00	0.00	0.00	9
12	0.20	0.24	0.22	25
13	0.37	0.37	0.37	19
14	0.33	0.43	0.38	7
15	0.21	0.25	0.23	12
16	0.15	0.12	0.14	16
17	0.10	0.08	0.09	12
18	0.00	0.00	0.00	11
19	0.63	0.84	0.72	31
20	0.00	0.00	0.00	8
21	0.56	0.60	0.58	15
22	0.18	0.18	0.18	11
23	0.29	0.32	0.30	19
24	0.50	0.07	0.12	14
25	0.20	0.12	0.15	16
26	0.07	0.11	0.09	9
27	0.00	0.00	0.00	16
28	0.33	0.11	0.17	9
29	0.00	0.00	0.00	10
30	0.08	0.10	0.09	10
31	0.00	0.00	0.00	13
32	0.23	0.33	0.27	9
33	0.07	0.07	0.07	15
34	0.46	0.35	0.40	17
35	0.15	0.25	0.19	8
36	0.08	0.10	0.09	10
37	0.33	0.23	0.27	13
38	0.29	0.33	0.31	15
39	0.19	0.18	0.18	17
40	0.06	0.09	0.07	11
41	0.00	0.00	0.00	17
42	0.12	0.14	0.13	7
43	0.75	0.43	0.55	7
44	0.00	0.00	0.00	5
45	0.00	0.00	0.00	11
46	0.67	0.60	0.63	20
47	0.06	0.07	0.06	14
48	0.40	0.33	0.36	6
49	0.10	0.12	0.11	8
50	0.14	0.12	0.13	17

51	0.24	0.26	0.25	19
52	0.60	0.50	0.55	6
53	0.20	0.27	0.23	11
54	0.19	0.20	0.19	15
55	0.33	0.42	0.37	24
56	0.29	0.25	0.27	8
57	0.54	0.41	0.47	17
58	0.17	0.18	0.17	17
59	0.00	0.00	0.00	10
60	0.18	0.18	0.18	11
61	0.25	0.10	0.14	10
62	0.00	0.00	0.00	5
63	0.36	0.29	0.32	17
64	0.33	0.33	0.33	3
65	0.71	0.91	0.80	11
66	0.00	0.00	0.00	11
67	0.25	0.25	0.25	4
68	0.20	0.20	0.20	10
69	0.73	0.73	0.73	11
70	0.40	0.22	0.29	9
71	0.29	0.22	0.25	9
72	0.25	0.38	0.30	8
73	1.00	0.00	0.00	7
74	0.42	0.33	0.37	15
75	0.50	0.50	0.50	14
76	0.08	0.09	0.09	11
77	0.50	0.14	0.22	14
78	0.43	0.50	0.46	6
79	0.27	0.31	0.29	13
80	0.33	0.09	0.14	11
81	0.14	0.11	0.12	19
82	0.00	0.00	0.00	17
83	0.50	0.18	0.27	11
84	0.31	0.31	0.31	13
85	0.14	0.20	0.17	5
86	0.00	0.00	0.00	10
87	0.50	0.31	0.38	16
88	0.56	0.45	0.50	11
89	0.50	0.17	0.25	6
90	0.25	0.40	0.31	10
91	0.50	0.33	0.40	9
92	0.56	0.77	0.65	13
93	0.35	0.44	0.39	16
94	0.49	0.69	0.57	52
95	0.20	0.12	0.15	8
96	0.27	0.50	0.35	8
97	0.00	0.00	0.00	7
98	0.54	0.58	0.56	12
99	0.71	0.36	0.48	14
100	0.67	0.89	0.76	9
accuracy			0.48	1829
macro avg	0.30	0.27	0.27	1829
weighted avg	0.45	0.48	0.45	1829

In [38]:
```
SVM(hog_features_train,y_train,hog_features_test,y_test)
```

```
Fitting 3 folds for each of 2 candidates, totalling 6 fits
```

```
c:\Users\Omar Wessam\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklea
rn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `
True` to `'auto'` in 1.5. Set the value of `dual` explicitly to suppress the warnin
g.
  warnings.warn(
```

```
Best Parameters: {'C': 0.1}
Accuracy: 0.6260251503553854
```

	precision	recall	f1-score	support
0	0.29	0.45	0.35	92
1	0.91	0.99	0.95	175
2	0.52	0.75	0.61	44
3	0.96	1.00	0.98	176
4	0.71	0.91	0.80	11
5	0.96	0.98	0.97	162
6	0.50	0.43	0.46	7
7	0.00	0.00	0.00	6
8	0.67	0.40	0.50	10
9	0.60	0.18	0.27	17
10	0.33	0.29	0.31	7
11	0.75	0.33	0.46	9
12	0.52	0.68	0.59	25
13	0.50	0.58	0.54	19
14	0.60	0.43	0.50	7
15	0.47	0.67	0.55	12
16	0.47	0.56	0.51	16
17	0.62	0.42	0.50	12
18	0.00	0.00	0.00	11
19	0.78	0.94	0.85	31
20	1.00	0.25	0.40	8
21	0.88	0.47	0.61	15
22	0.56	0.45	0.50	11
23	0.32	0.32	0.32	19
24	0.57	0.29	0.38	14
25	0.62	0.50	0.55	16
26	0.13	0.22	0.17	9
27	0.50	0.31	0.38	16
28	0.00	0.00	0.00	9
29	0.11	0.10	0.11	10
30	0.50	0.30	0.38	10
31	0.36	0.31	0.33	13
32	0.71	0.56	0.62	9
33	0.40	0.13	0.20	15
34	0.86	0.71	0.77	17
35	0.45	0.62	0.53	8
36	0.23	0.30	0.26	10
37	0.14	0.08	0.10	13
38	0.69	0.60	0.64	15
39	0.64	0.53	0.58	17
40	0.10	0.09	0.10	11
41	0.27	0.24	0.25	17
42	0.20	0.14	0.17	7
43	0.80	0.57	0.67	7
44	0.33	0.60	0.43	5
45	0.75	0.55	0.63	11
46	0.83	0.75	0.79	20
47	0.29	0.43	0.34	14
48	0.50	0.33	0.40	6
49	0.25	0.25	0.25	8
50	0.64	0.41	0.50	17
51	0.36	0.42	0.39	19

```
          52       0.80      0.67      0.73         6
          53       0.60      0.55      0.57        11
          54       0.21      0.40      0.28        15
          55       0.76      0.54      0.63        24
          56       0.33      0.25      0.29         8
          57       0.89      0.94      0.91        17
          58       0.32      0.41      0.36        17
          59       0.12      0.10      0.11        10
          60       0.33      0.27      0.30        11
          61       0.50      0.40      0.44        10
          62       0.00      0.00      0.00         5
          63       0.72      0.76      0.74        17
          64       0.33      0.33      0.33         3
          65       0.73      0.73      0.73        11
          66       0.14      0.09      0.11        11
          67       0.27      0.75      0.40         4
          68       0.57      0.40      0.47        10
          69       1.00      0.91      0.95        11
          70       0.00      0.00      0.00         9
          71       0.71      0.56      0.62         9
          72       0.33      0.50      0.40         8
          73       0.00      0.00      0.00         7
          74       0.67      0.53      0.59        15
          75       0.80      0.86      0.83        14
          76       0.38      0.27      0.32        11
          77       0.77      0.71      0.74        14
          78       0.80      0.67      0.73         6
          79       0.57      0.62      0.59        13
          80       0.80      0.36      0.50        11
          81       0.29      0.21      0.24        19
          82       0.12      0.06      0.08        17
          83       0.75      0.27      0.40        11
          84       0.50      0.38      0.43        13
          85       1.00      0.40      0.57         5
          86       0.18      0.30      0.22        10
          87       0.62      0.62      0.62        16
          88       0.89      0.73      0.80        11
          89       1.00      0.17      0.29         6
          90       0.40      0.80      0.53        10
          91       0.50      0.33      0.40         9
          92       0.57      0.92      0.71        13
          93       0.75      0.75      0.75        16
          94       0.72      0.92      0.81        52
          95       0.00      0.00      0.00         8
          96       0.30      0.38      0.33         8
          97       0.00      0.00      0.00         7
          98       1.00      0.67      0.80        12
          99       1.00      0.36      0.53        14
         100       0.80      0.89      0.84         9

    accuracy                           0.63      1829
   macro avg       0.52      0.45      0.46      1829
weighted avg       0.64      0.63      0.62      1829
```

In [27]:
```
SVM(clhg_features_train,y_train,clhg_features_test,y_test)
```

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
Best Parameters: {'C': 10}
Accuracy: 0.32640787315472936
              precision    recall  f1-score   support

           0       0.17      0.10      0.12        92
           1       0.62      0.95      0.75       175
           2       0.65      0.80      0.71        44
           3       0.32      0.88      0.47       176
           4       0.12      0.27      0.17        11
           5       0.44      0.73      0.55       162
           6       1.00      0.00      0.00         7
           7       0.00      0.00      0.00         6
           8       0.00      0.00      0.00        10
           9       1.00      0.00      0.00        17
          10       1.00      0.00      0.00         7
          11       0.00      0.00      0.00         9
          12       0.29      0.08      0.12        25
          13       0.09      0.16      0.12        19
          14       0.00      0.00      0.00         7
          15       0.00      0.00      0.00        12
          16       0.11      0.06      0.08        16
          17       0.00      0.00      0.00        12
          18       0.00      0.00      0.00        11
          19       0.79      1.00      0.89        31
          20       0.00      0.00      0.00         8
          21       0.00      0.00      0.00        15
          22       0.00      0.00      0.00        11
          23       0.00      0.00      0.00        19
          24       1.00      0.00      0.00        14
          25       0.15      0.12      0.14        16
          26       0.00      0.00      0.00         9
          27       0.00      0.00      0.00        16
          28       0.00      0.00      0.00         9
          29       0.00      0.00      0.00        10
          30       0.00      0.00      0.00        10
          31       0.00      0.00      0.00        13
          32       0.33      0.67      0.44         9
          33       0.27      0.20      0.23        15
          34       0.00      0.00      0.00        17
          35       0.00      0.00      0.00         8
          36       0.00      0.00      0.00        10
          37       0.00      0.00      0.00        13
          38       0.00      0.00      0.00        15
          39       0.00      0.00      0.00        17
          40       0.00      0.00      0.00        11
          41       0.00      0.00      0.00        17
          42       0.10      0.14      0.12         7
          43       0.00      0.00      0.00         7
          44       0.00      0.00      0.00         5
          45       0.00      0.00      0.00        11
          46       0.00      0.00      0.00        20
          47       0.11      0.14      0.12        14
          48       0.00      0.00      0.00         6
          49       0.17      0.12      0.14         8
          50       0.10      0.06      0.07        17
```

51	0.14	0.05	0.08	19
52	0.50	0.17	0.25	6
53	0.08	0.09	0.09	11
54	0.00	0.00	0.00	15
55	0.22	0.25	0.24	24
56	0.00	0.00	0.00	8
57	0.00	0.00	0.00	17
58	0.00	0.00	0.00	17
59	0.00	0.00	0.00	10
60	0.30	0.27	0.29	11
61	0.00	0.00	0.00	10
62	0.00	0.00	0.00	5
63	0.00	0.00	0.00	17
64	0.00	0.00	0.00	3
65	0.07	0.18	0.10	11
66	0.00	0.00	0.00	11
67	0.00	0.00	0.00	4
68	1.00	0.00	0.00	10
69	0.20	0.36	0.26	11
70	0.00	0.00	0.00	9
71	1.00	0.00	0.00	9
72	0.25	0.75	0.38	8
73	0.00	0.00	0.00	7
74	0.00	0.00	0.00	15
75	0.00	0.00	0.00	14
76	0.00	0.00	0.00	11
77	0.00	0.00	0.00	14
78	0.00	0.00	0.00	6
79	0.10	0.08	0.09	13
80	0.00	0.00	0.00	11
81	0.31	0.21	0.25	19
82	0.00	0.00	0.00	17
83	0.30	0.27	0.29	11
84	0.00	0.00	0.00	13
85	0.25	0.20	0.22	5
86	0.00	0.00	0.00	10
87	0.00	0.00	0.00	16
88	0.24	0.36	0.29	11
89	0.29	0.33	0.31	6
90	0.33	0.70	0.45	10
91	0.25	0.22	0.24	9
92	0.12	0.31	0.17	13
93	0.00	0.00	0.00	16
94	0.14	0.08	0.10	52
95	0.00	0.00	0.00	8
96	0.00	0.00	0.00	8
97	0.00	0.00	0.00	7
98	0.00	0.00	0.00	12
99	0.00	0.00	0.00	14
100	0.22	0.22	0.22	9
accuracy			0.33	1829
macro avg	0.15	0.11	0.09	1829
weighted avg	0.25	0.33	0.25	1829