REPORT OF THE COMBINATION : CNN WITH SVM FOR IMAGE CLASSIFICATION

Amine Mami.

Abstract:

This report presents a study on the classification of images of cats and dogs using a Support Vector Machine (SVM) model. The approach involves extracting features from the images using the ResNet50 convolutional neural network architecture. The dataset utilized for training and testing purposes is the Dogs vs. Cats dataset from Kaggle. The performance of the SVM model is evaluated using various metrics such as accuracy, precision, recall, F1-score, and the confusion matrix.

1. Introduction

Image classification is a fundamental task in computer vision, with applications ranging from object recognition to medical diagnosis. In this study, we focus on the classification of cats and dogs images, which presents a classic problem in computer vision. The aim is to develop an accurate model that can differentiate between these two classes.

2. “How I Do These TP Tasks?”

a step-by-step explanation of how to perform various tasks related to a machine learning project. The tasks include mounting the Google Drive, importing necessary libraries, loading the dataset, splitting the dataset, initializing and training the SVM model, making predictions on the test set, printing the classification report, printing the accuracy score, and printing the confusion matrix.

- **Mounting the Google Drive:**

The code begins with mounting the Google Drive using the Google Colab library. This step allows me to access files stored in my Google Drive.

```
from google.colab import drive
drive.mount('/content/drive')
```

- **Importing the Necessary Libraries:**

```
import numpy as np
import pandas as pd
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from tensorflow.keras.preprocessing import image
import os
```

The code imports the required libraries to execute the tasks. The libraries include:

pandas: Used for data manipulation and analysis.

train_test_split from sklearn.model_selection: Used to split the dataset into training and testing sets.

SVC from sklearn.svm: Used for training the Support Vector Machine (SVM) model.

Various metrics from sklearn.metrics: Used for calculating evaluation metrics such as precision, recall, F1-score, accuracy, and confusion matrix.

- **Checks if a GPU device is available and prints the result:**

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  print('GPU device not found')
else:
  print('Found GPU at: {}'.format(device_name))
```

- **Loading the Dataset and features extraction:**
  - Loads the ResNet50 model pretrained on ImageNet:

```
model = ResNet50(weights='imagenet', include_top=False)
```

  - Sets the path to the Kaggle Dogs vs. Cats dataset:

```
dataset_path = '/content/drive/MyDrive/train'
```

  - Defines a function extract_features to extract features from an image using ResNet50:

```
def extract_features(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)

    features = model.predict(img)
    features = features.flatten()

    return features
```

  - Collects image paths and labels from the dataset directory:

```
image_paths = []
labels = []

for root, dirs, files in os.walk(dataset_path):
    for file in files:
        if file.endswith('.jpg'):
            image_paths.append(os.path.join(root, file))
            labels.append(file.split('.')[0])
```

- Extracts features for each image and stores them in a list:

```
features_list = []
for img_path in image_paths:
    features = extract_features(img_path)
    features_list.append(features)
```

- Converts the list of feature vectors and labels to a pandas DataFrame:

```
df = pd.DataFrame(features_list)
df['label'] = labels
```

- Saves the DataFrame to a CSV file named "image_features.csv":

```
df.to_csv('image_features.csv', index=False)
```

- Finally, this code loads the extracted features from the CSV file named 'image_features.csv' using the pd.read_csv() function. The loaded data is assigned to the variable 'df', which represents a pandas DataFrame:

```
# Load the extracted features from the CSV file
df = pd.read_csv('image_features.csv')
```

- **Splitting the Dataset:**

The code splits the dataset into training and testing sets using the train_test_split() function from the sklearn.model_selection module. The features (all columns except 'label') are assigned to the variables 'X_train' and 'X_test', while the labels ('label' column) are assigned to 'y_train' and 'y_test'. The splitting ratio is set to 0.3, indicating that 30% of the data is used for testing.

```
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df.drop('label', axis=1), df['label'], test_size=0.3, random_state=42
)
```

- **Initializing and Training the SVM Model:**

The code initializes an SVM model with specific hyperparameters such as C=1.0, kernel='rbf', gamma='scale', and random_state=42. The initialized model is assigned to the variable 'svm_model'. Subsequently, the model is trained on the training data using the fit() method.

```
# Initialize and train the SVM model with hyperparameters
svm_model = SVC(C=1.0, kernel='rbf', gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
```

- **Making Predictions on the Test Set:**

The code employs the trained SVM model to make predictions on the test set (X_test). The predicted values are stored in the variable 'y_pred'.

```
# Make predictions on the test set
y_pred = svm_model.predict(X_test)
```

- **Printing the Classification Report:**

The code calculates and prints the classification report using the classification_report() function from the sklearn.metrics module. The classification report provides information on precision, recall, F1-score, and support for each class (cat and dog).

```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

- **Printing the Accuracy Score:**

The code calculates and prints the accuracy score using the accuracy_score() function from the sklearn.metrics module. The accuracy score represents the proportion of correctly classified instances.

```
# Print accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

- **Printing the Confusion Matrix:**

The code calculates and prints the confusion matrix using the confusion_matrix() function from the sklearn.metrics module. The confusion matrix presents the number of true positive, true negative, false positive, and false negative predictions.

```
# Print confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_mat)
```

3. Dataset Description

The dataset consists of a large collection of labeled images of cats and dogs, with a total of 25,000 images [1]. The images vary in terms of size, breed, background, and poses of the animals. The dataset was divided into a training set containing 20,000 images and a testing set containing 5,000 images. Each image is labeled as either a cat or a dog, enabling supervised learning for classification purposes.
Access to the dataset can be obtained from the Kaggle competition link:
https://www.kaggle.com/competitions/dogs-vs-cats/data

Ps: To facilitate the learning step, take only a portion of the dataset (500 cat images and 500 dog images) and resize the images to 224x224 pixels.

4. Approach Description (Methodology)

4.1 Feature Extraction:
the ResNet50 architecture is utilized to extract discriminative features from the images. ResNet50 is a popular deep convolutional neural network architecture that has achieved remarkable performance in various computer vision tasks [2].

ResNet50 consists of 50 layers, including residual blocks, which allow for the training of deeper networks without suffering from the vanishing gradient problem. These residual blocks enable the network to learn more complex and abstract features from the images, making it particularly suitable for image classification tasks.

By employing the ResNet50 architecture, the model takes advantage of the pre-trained weights learned on the large-scale ImageNet dataset [2]. This pre-training enhances the model's ability to extract meaningful features from the input images.

During the feature extraction process, each image is resized to match the input size of ResNet50 (typically 224x224 pixels) and undergoes preprocessing steps, such as normalization, before being fed into the network [1]. The image is then passed through the ResNet50 model, and the output from the last convolutional layer or global average pooling layer is obtained as the feature representation of the image.

The extracted features form a high-dimensional vector that captures the distinctive characteristics of each image. These features can subsequently be used as inputs for downstream tasks such as classification, clustering, or similarity comparison [3].

It is worth noting that different layers of the ResNet50 architecture can be used for feature extraction, depending on the specific requirements of the task at hand. For instance, the last fully connected layer (often referred to as the "fc" layer or "fc7") can be used to obtain a feature vector

with a dimensionality of 2048 [1]. This dimensionality represents the number of in_features of the fully connected layer and can serve as a basis for subsequent analysis or classification.

In summary, the integration of the ResNet50 architecture in the feature extraction process enhances the model's capability to capture meaningful and discriminative features from the input images, facilitating accurate classification or further analysis.

4.2 SVM Model:

The extracted features are then used as inputs for an SVM classifier. We split the dataset into training and test sets using the train_test_split function from the scikit-learn library. The SVM model is trained using the training set with hyperparameters set as follows: C=1.0, kernel='rbf', and gamma='scale'. Once trained, the model is applied to the test set to make predictions.

5. Results and Discussion:

The performance of the SVM model is evaluated using several metrics: accuracy, precision, recall, F1-score, and the confusion matrix. The accuracy score indicates the overall classification performance, achieving an accuracy of 97.67%. Precision measures the proportion of correctly classified instances within a specific class, with values of 0.99 for cats and 0.96 for dogs. Recall, also known as sensitivity or true positive rate, indicates the proportion of correctly classified instances relative to the actual instances, yielding values of 0.96 for cats and 0.99 for dogs. The F1-score combines precision and recall, resulting in values of 0.98 for both classes. The confusion matrix provides detailed information on the classification results, revealing 144 true positives, 6 false positives, 1 false negative, and 150 true negatives.

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

         cat       0.99      0.96      0.98       150
         dog       0.96      0.99      0.98       151

    accuracy                           0.98       301
   macro avg       0.98      0.98      0.98       301
weighted avg       0.98      0.98      0.98       301

Accuracy: 0.9767441860465116
Confusion Matrix:
[[144   6]
 [  1 150]]
```

6. Conclusion:

In this TP, we successfully developed an SVM model for classifying images of cats and dogs using features extracted from the ResNet50 architecture. The model exhibited high accuracy and demonstrated excellent performance in distinguishing between the two classes. The precision, recall, F1-score, and confusion matrix analysis further validate the model's effectiveness. This TP contributes to the field of computer vision and image classification, showcasing the power of combining deep learning architectures and traditional machine learning techniques for solving practical image recognition problems.

7. References (4.1 Feature Extraction:):

[1] Stack Overflow. "Extract features from pretrained ResNet50 in PyTorch. Link

[2] Datagen.tech. "Resnet-50 Image Classification with Transfer Learning." Link

[3] AI Stack Exchange. "How is a ResNet-50 used for deep feature extraction?" Link