Assignment: Image Classification using k-NN with Euclidean (L2) Distance and 5-Fold Cross Validation

Objective

This assignment focuses on using the k-Nearest Neighbors (k-NN) algorithm with Euclidean (L2) distance to classify images of wild animals. You will apply 5-fold cross-validation to evaluate the model's performance and plot the relationship between the number of neighbors KKK and the average accuracy across folds. The final goal is to find the centroid (mean) of the 5-fold accuracies for each KKK value and draw a line through these centroids to visualize the accuracy trend.

Dataset

- **Dataset Link**: Wild Animals Dataset on Kaggle
- This dataset contains images of various wild animals. It will be used for multi-class classification.

Requirements

1. **Dataset**: Download and load the dataset.
2. **Preprocessing**: Convert each image to grayscale before calculating distances.
3. **Distance Metric**: Use Euclidean (L2) distance to determine similarity.
4. **Classification Model**: Implement the k-Nearest Neighbors (k-NN) classifier with a range of KKK values from 1 to 30.
5. **Cross-Validation**: Apply 5-fold cross-validation, where for each K value, you will calculate accuracy across 5 folds.
6. **Visualization**: Plot the average accuracy for each K, with the centroid points of the 5-fold accuracies connected by a line.

Instructions

1. **Download and Load Dataset**
    - Download the dataset from Kaggle and load it into your Jupyter notebook.
    - Ensure that each image has the correct class label.
2. **Image Preprocessing**
    - Resize or reshape images to ensure consistent dimensions across the dataset.
    - Convert each image to grayscale to simplify the distance calculations.
3. **5-Fold Cross Validation**
    - Split your dataset into 5 folds. For each fold:
        - Use one fold as the validation set and the remaining 4 folds as the training set.
        - Repeat this process so each fold is used as the validation set once.
    - For each test image, calculate the Euclidean (L2) distance to all training images in that fold.

4. **k-NN Classification with Euclidean Distance**
    - Implement the k-NN classifier for each fold. For each test image, find the K nearest neighbors (smallest L2 distances) and assign the most common label among the neighbors as the predicted class.
    - Calculate the accuracy of each fold for each K value.
    - Repeat for K values from 1 to 30.
5. **Average Accuracy and Centroid Calculation**
    - For each K value, calculate the average accuracy across the 5 folds.
    - Find the centroid of the 5 accuracies for each K by computing their mean.
6. **Plotting the Results**
    - Plot a graph with K values on the x-axis and the average accuracy (centroids) on the y-axis.
    - Draw a line connecting these centroids to illustrate the trend in accuracy as K changes.

Deliverables

1. **Code**: Submit your Jupyter notebook (.ipynb) file containing:
    - Code for dataset loading, preprocessing, 5-fold cross-validation, and k-NN classification.
    - Code for calculating and plotting the average accuracy with centroids.
2. **Graph**: The plot should include:
    - K values (1 to 30) on the x-axis.
    - Average accuracy for each K on the y-axis, with a line connecting the centroids.
3. **Submission**: Upload your Jupyter notebook with outputs to your GitHub repository in the **CVPR/MID** folder. Jupyter notebook should be named "**knn.ipynb".**

Evaluation Criteria

- Proper implementation of 5-fold cross-validation.
- Correct computation of Euclidean distance and k-NN classification.
- Accurate calculation of average accuracy and plotting of centroids.
- Clear, organized, and well-documented code in the notebook.

This assignment will give you a hands-on understanding of k-NN classification, cross-validation, and performance analysis. Good luck!

SUBMISSION DATE: 16th November 20204.