Report

Hamza Rehan

October 1, 2023

1 Task 1: Yale Dataset Image Classification

In Task 1, we focused on classifying images from the Yale dataset using a Multilayer Perceptron (MLP) model. This section discusses the results of the experiment with and without hyperparameter tuning.

1.1 Data Preprocessing

To prepare the data for the MLP model, we performed the following steps:

- Loaded and displayed sample images.
- Resized and normalized pixel values.
- Preprocessed images for use with the ResNet50 model.

1.2 MLP Model Implementation

We implemented an MLP model with customizable architecture, including multiple hidden layers and activation functions. Backpropagation was used for training.

1.2.1 Experiment 1: MLP Model with Hyperparameter Tuning

We conducted an experiment with hyperparameter tuning using Scikit-learn's GridSearchCV. We fine-tuned various hyperparameters, including the number of neurons per layer, the number of hidden layers, activation functions, epochs, learning rate, and momentum.

1.2.2 Experiment 2: MLP Model without Hyperparameter Tuning

In this experiment, we trained the MLP model without hyperparameter tuning.

1.3 Results and Evaluation

We presented the results of both experiments, including accuracy, precision, recall, and F1-score metrics. Confusion matrices were also displayed.

1.4 Training and Validation Accuracy Plots

We plotted the training and validation accuracies across epochs for both experiments.

2 Task 3: Face Recognition System

This report provides an overview of the development and implementation of an Incremental Face Recognition System. The system is designed to recognize employees of FAST based on their facial features. It supports registration of new employees and real-time recognition of registered users through a web interface.

3 System Components

The Incremental Face Recognition System consists of the following key components:

3.1 Face Recognition Model

The face recognition model utilizes the ‘face$_r$*ecognition‘library, whichisbuiltontopof‘dlib‘.Knownfacesandthe*
timerecognitionofemployees.

3.2 Database Integration

The system integrates with an SQLite database to store employee information, including their name, email, and facial encodings. This ensures that the system recognizes registered employees.

3.3 Web Interface for Registration

The web interface is developed using Flask, a Python web framework. The ‘/register‘ route allows employees to register by providing their name and email. New user information is stored in the database, and the face recognition model is updated to include the new user’s data.

3.4 Face Recognition Endpoint

The ‘/recognize‘ route is responsible for recognizing registered users based on real-time facial features. Employees can use their cameras to show their faces, and the system detects and identifies them if they are registered.

4 Workflow

The workflow of the Incremental Face Recognition System is as follows:

4.1 Registration of New Employees

1. New employees access the registration page ('/register') through a web browser.
2. Employees provide their name and email for registration.
3. Upon registration, the system captures user images, generates a unique user ID, and stores this information in the SQLite database.
4. The face recognition model is updated with the new user's images to enable future recognition.

4.2 Real-time Face Recognition

1. Registered employees access the face recognition page ('/recognize') and provide access to their camera.
2. The system captures and processes real-time video frames.
3. Detected faces are compared to known faces using the face recognition model.
4. If a match is found, the system displays the employee's name; otherwise, it indicates "Unknown user."

5 Conclusion

The Incremental Face Recognition System offers a user-friendly solution for employee recognition at FAST. New employees can easily register, and the system ensures that the recognition model is continually updated to accommodate new users. This system has applications in security, access control, and attendance tracking.

6 Task 4: Skin Lesion Classification

In Task 4, we developed a skin lesion classification model using MLP. This section discusses the steps involved in data preprocessing, model training, and evaluation.

6.1 Data Preprocessing

For skin lesion classification, we followed these data preprocessing steps:

- Loaded and split data from a local directory.
- Encoded labels.
- Displayed sample images.

6.2 MLP Model Implementation

We implemented an MLP model for skin lesion classification, including input and hidden layers. The model was trained using backpropagation.

6.3 Model Training and Evaluation

We trained the MLP model with training data, calculated the training error, and evaluated the model's performance on the test data. This included displaying a confusion matrix and computing accuracy, precision, recall, and f1-score.

7 Conclusion

In this report, we conducted two tasks: Yale Dataset Image Classification and Skin Lesion Classification. Both tasks involved the use of MLP models, with Task 1 focusing on image classification and Task 4 on skin lesion classification. These models have the potential for various applications in healthcare and image analysis.