COM6018 Assignment 2

4th Dec 2024. v1.0.1 (fixed typo: train.full.joblib -> train.joblib)
15th November 2024. v1.0.0

Face Verification

Due: 15:00, Wednesday 18th December 2024

Table of Contents

1. Introduction

Your task is to build a face verification system that can take a pair of face images and determine whether they are of the same person or not. The system will be evaluated on a test set of face pairs labelled as either 'same' or 'different'. The performance of your system will be measured in terms of accuracy.

Further details on the data, task, assessment criteria, and submission process are provided below.

2. The Data

You will be provided with training and evaluation datasets constructed from the 'Labeled Faces in the Wild' dataset. This is the same data used in the lab for the person identification task.

2.1. Training data

The training set consists of 2,200 examples of pairs of images and corresponding labels to indicate whether the images are of the same person or not. The data is stored in the file train.joblib. The training data has an equal proportion of 'same' and 'different' pairs.

```python
import joblib

data = joblib.load('train.joblib')
```

The data variable is a dictionary with the following structure:

```
data = {
    'data': <2200 x 5828 numpy array of pixels for pairs of face images>,
    'target': <2200 element 1D numpy array of class labels>
    },
}
```

The data can be reshaped into pairs of 62 x 47 images with:

```
images = data['data'].reshape(-1, 2, 62, 47)
```

The target class labels are either 0 or 1, representing 'different person' or 'same person', respectively.

2.2. Evaluation data

The evaluation set consists of 1,000 examples of pairs of images and corresponding labels. The data is stored in the file eval1.joblib. The evaluation data has an equal proportion of 'same' and 'different' pairs. Note that none of the people in the evaluation set appear in the training set.

The evaluation data can be loaded similarly to the training data:

```
import joblib

data = joblib.load('eval1.joblib')
```

The data variable is a dictionary with the following structure:

```
data = {
    'data': <1000 x 5828 numpy array of pixels for pairs of face images>,
    'target': <1000 element 1D numpy array of class labels>
    },
}
```

The target class labels have the same interpretation as for the training data.

2.3. The baseline models

You are provided with an example model that uses a 1-nearest neighbour classifier on the raw pixel data. This model is called baseline_model.joblib.

The performance of this model on the evaluation data is 56.3%. While this is not particularly high, it is significantly better than random guessing (50%). Your goal is to outperform this baseline.

3. The Task

3.1 Developing a Classification Model

You are asked to develop your own model, which should be saved using joblib to a file called model.joblib. You will submit this file along with the code used to train the model.

The code that you write to train and save the model should be in a Python script called train.py, which you will also submit.

You are provided with a script evaluate.py, which you can use to check that your models work. This script will also be used to assess your system.

To run the evaluation script:

```
python evaluate.py <MODEL_FILE_NAME>
```

For example, to test your model:

```
python evaluate.py model.joblib
```

Your model files should be created by a Python script called train.py that you will need to write and submit. It will take the training data as input and save the trained model to a joblib file. The train.py script should be run with

```
python train.py <TRAINING_DATA_FILE_NAME> <MODEL_FILE_NAME>
```

So, for example, you can train your model files with

```
python train.py train.joblib model.joblib
```

3.2. Writing a Report

You are also required to write a report, which you will submit as a PDF file named report.pdf. The report should be no more than two pages long and include the following sections:

- **Introduction**: A brief description of the face verification problem.
- **System Description**: A complete description of your verification system pipeline, highlighting critical hyper-parameters to optimise.
- **Experiments**: A description of the experiments you conducted to tune your system's hyper-parameters, including the construction of your training dataset. Be explicit about which hyper-parameters you experimented with and how they influenced the performance of your model. Include results that justify your final choices.
- **Results and Analysis**: Provide an analysis of the performance of your final system on the evaluation data, including a comparison to the baseline system. Include a table that reports the accuracy of your model. Additionally, provide a brief discussion of any observed trends or insights, highlighting factors that may have influenced the results.
- **Conclusions**: A summary of your work, including suggestions for further improvements.
- **References**: A list of any references cited in your report.

The report should be written clearly and concisely, using LaTeX. A LaTeX template has been provided.

4. Additional Rules

Some additional rules have been set to ensure that the assignment is fair for everyone. Please read carefully,

- **Your model file should be named 'model.joblib' and must not exceed 80 MB in size.**
- **You can only train your model using the provided training data (augmentation is allowed).**
- **You cannot use any pre-trained models.**
- **You may only use the standard Python libraries and the following: numpy, matplotlib, seaborn, pandas, scikit-learn, joblib.**

5. Assessment

This assignment is worth 60% of the module mark and will be graded out of 60.

Evaluation Tips

- Training Data Augmentation: Consider augmenting your training data by applying transformations such as small rotations, scaling, or flipping to increase the diversity of your dataset. This can help your model generalize better and avoid overfitting.

- Feature Engineering: Try different ways to preprocess the images to extract more meaningful features beyond raw pixel values.

- Model Complexity: Experiment with different model architectures and complexity levels, but be careful not to overfit to the training data. Cross-validation can help you assess generalizability.

The final mark will be based on the following criteria:

- The quality and clarity of your code (20/60)
- The quality and clarity of the written report (30/60)
- The performance of your classifier on a hidden evaluation data (10/60)

Further details of the assessment criteria are given below.

5.1. Code (20/60)

Marks will be awarded based on the following rubric:

Score	Description
Fail (0-9)	The code lacks clarity, organization, and proper documentation. It may have significant errors, making it challenging to understand or run. Essential machine learning concepts are not demonstrated.
Pass (10-11)	The code meets the minimum requirements and demonstrates a basic understanding of scikit-learn. However, the organization and documentation could be improved for better clarity.
Merit (12-13)	The code meets all requirements and shows a good understanding of scikit-learn. It is well-organized, with clear documentation that aids in understanding the implementation.
Distinction (14-15)	The code exceeds expectations, demonstrating a thorough understanding of scikit-learn. It is well-organized, follows best practices, and includes comprehensive documentation that enhances clarity.
Distinction (16-17)	The code is of exceptional quality, showcasing a high level of mastery. It not only meets all requirements but does so with exceptional clarity, well-structured organization, and detailed documentation.
Exceptional (18-20)	The code is flawless. It demonstrates outstanding clarity, organization, and documentation. The implementation goes beyond the requirements, showcasing creativity and innovation in classifier design.

5.2. Report (30/60)

Marks will be awarded based on the following rubric:

Score	Description
Fail (0-15)	The report lacks a clear and coherent description of the verification system. It does not provide sufficient details for someone to reproduce the work. There are significant issues with the presentation of figures, tables, and referencing.
Pass (16-18)	The report offers a basic description of the verification system, but it lacks clarity and coherence. There are areas that could be more detailed. Reproduction of the work is possible with effort. LaTeX is used, but there are some issues with figures, tables, or referencing.

Merit (19-21)	The report effectively describes the verification system with clarity and organization. The details provided make it reasonably easy to reproduce the work. LaTeX is used appropriately, but there may be some minor issues with figures, tables, or referencing.
Distinction (22-24)	The report exceeds expectations in clarity and detail, making it easy to reproduce the work. The document has well-designed figures, tables, and accurate referencing.
Distinction (25-27)	The report is of exceptional quality, providing a clear and highly detailed description of the verification system. Reproducing the work is straightforward. The document has well-designed figures, tables, and accurate referencing.
Exceptional (28-30)	The report is flawless. It excels in clarity, detail, and reproducibility. The document has well-designed figures, tables, and accurate referencing. The document is of a publishable quality.

5.3. Performance (10/60)

Marks will be awarded based on the following rubric. Note: The hidden evaluation dataset is similar in structure to the provided evaluation set, but it includes different individuals to test your model's generalization capabilities:

Score	Description
Fail (0-4)	The system does not run
Pass (5)	The system runs and produces results.
Merit (6)	The system runs and produces results that are better than the baseline system.
Distinction (7)	The system runs and produces results that are better than the baseline system and you have clearly documented some results showing hyper-parameter tuning.
Distinction (8)	The system runs and produces results that are better than the baseline system and you have thoroughly investigated the effect of hyper-parameter tuning.
Exceptional (9 or 10)	As above but with bonus marks for exceptional performance (e.g., top 2 or 3 scores in the class) and/or particularly thorough experimentation.

6. Submission

You will need to submit the following files:

- **train.py**: The Python script that trains your classifier. Include clear comments explaining the training process and key steps.
- **report.pdf**: PDF report describing your system and experiments. Ensure all sections are included and the report is formatted according to LaTeX standards.
- **model.joblib**: A joblib file containing your trained model. This should be the output of your trained model and must not exceed 80 MB.

A submission page will appear on the module's Blackboard site where you will be able to upload the above three files.

The assignment is due by 15:00 on Wednesday, 18th December. Standard lateness penalties apply.

Academic Misconduct

The University takes academic misconduct very seriously. Academic misconduct includes plagiarism, collusion and fabrication of data. You should read the University's policy on academic misconduct to ensure you understand what is and what is not acceptable. The policy is available from https://www.sheffield.ac.uk/new-students/unfair-means (https://www.sheffield.ac.uk/new-students/unfair-means). In particular, this is an individual assignment and any suspected collusion will be investigated. Do not share your code with anyone else.
