A face-orientation estimation system

acs23ab

University of Sheffield

Abstract

This assignment is centered around optimising face orientation estimation through the application of machine learning techniques. The objective is to exceed the performance of baseline models in accurately determining the orientation angle of facial images, while accommodating random rotations. The modeling approach involves the utilisation of MLP classifiers for the execution of this image classification task. The experiments conducted assess and compare the performance of MLP Classifier against baseline models, with a focus on improvements in accuracy. The findings of this assignment offer valuable insights into the effectiveness of the chosen machine learning methodology for image orientation tasks, paving the way for enhanced accuracy in real-world applications.

1. Introduction

In the field of image processing and computer vision, accurate face orientation estimation is significant for various applications, from facial recognition systems to augmented reality interfaces[1]. This assignment focuses onto the optimisation of face orientation estimation using advanced machine learning techniques, specially emphasising on surpassing the capabilities of baseline models, provided[2]. The primary objective is to enhance accuracy in determining the orientation angle of facial images, accommodating arbitrary rotations. Using Multi-Layer Perceptron (MLP) classifiers[3], the effectiveness of this modeling technique is explored in this image classification task. With the help of experiments and comparisons, this assignment aims to provide valuable insights that can contribute to improved accuracy in real-world scenarios.

After this introduction, a detailed System Description is delved into in Section 2, where a replicable system is outlined for face orientation estimation, extracting images, generating diverse datasets, and utilising tailored pipeline models with tuned hyperparameters, aiming to enhance accuracy. In Section 3, experiments are detailed, providing insights into the given image dataset, the experimental setup, and comparisons with baseline models. Section 4 presents Results and Analysis, giving a complete understanding of the system's performance. Section 5 involves Discussion and Conclusions, where findings are interpreted, and future directions are suggested. The report is concluded with Section 6; References,which lists the sources supporting this investigation. Now, the intricate landscape of face orientation estimation will be navigated, and the insights obtained from this examination will be resolved.

2. System Description

In this section, a detailed walk-through of the entire system is provided, designed to facilitate the replication of our results. The process is initiated by extracting images from the training data file 'train.full.joblib'. Subsequently, patches of varying sizes—specifically 30 pixels, 50 pixels, and 90 pixels—are extracted from these images. To differentiate the dataset, random

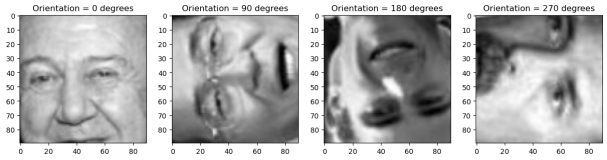

Figure 1: *Subimages of 90x90 size with different orientations.*

images are rotated at 0, 90, 180, and 270 degrees, and corresponding labels (0 for 0 degrees, 1 for 90 degrees, 2 for 180 degrees, and 3 for 270 degrees) are assigned to these sub-images.

The labeled data then enters a processing pipeline comprising two key estimators: Principal Component Analysis (PCA) and a Multi-Layer Perceptron (MLP) classifier. Three distinct pipeline models are constructed, each customised to the specific patch size (30 pixels, 50 pixels, and 90 pixels). These models are saved as joblib files for testing in the Python code, 'evaluate.py'.

A crucial aspect of the system involves the tuning of hyperparameters to optimise model performance. Specifically, the n_components[4] hyperparameter in PCA (for dimensionality reduction to that number of components) and the hidden_layer_sizes[5] hyperparameter in the MLP classifier (representing the number of layers and neurons in each layer) are tuned. This meticulous tuning process is undertaken to identify the best-performing pipeline model for each patch size.

By thoroughly detailing each step in this pipelining process and emphasising the tunable hyperparameters, it is aimed to provide a comprehensive guide for replicating our experimental setup and achieving comparable results.

3. Experiments

Efforts were invested in optimising the face orientation estimation system through experiments, focusing on training data selection, hyperparameter tuning, and adherence to model size constraints.

To select optimal training data, images were extracted from 'train.full.joblib,' and patches of varying sizes (30 pixels, 50 pixels, and 90 pixels) were diversified through random rotations (0, 90, 180, and 270 degrees). Labels were assigned to ensure a diverse and representative training dataset.

System performance optimisation included tuning hyperparameters, specifically n_components in PCA and hidden_layer_sizes in the MLP classifier. Hyperparameter optimisation used a 'For' loop instead of GridSearchCV()[6] to manage complexity and keep model file sizes below 20 MB. Different sets of n_components and hidden_layer_sizes values were employed for different patch sizes.

Meticulous management of hyperparameter optimisation considered the constraints on model size, ensuring optimal values while adhering to prescribed limits of file size.

The experiments incorporated both the k-NN and MLP

classifiers, involving the tuning of hyperparameters such as the number of nearest neighbors for k-NN and the number of components for PCA in k-NN. The outcomes were generated through the execution of the Python script 'evaluate.py'. Subsequently, the Results and Analysis section furnishes a comprehensive comparison between the MLP and k-NN classifiers, providing valuable insights into the distinctions in their modeling approaches.

These experiments demonstrated a significant accuracy enhancement in the face orientation estimation system with the chosen hyperparameter values. The subsequent Results and Analysis section delves into detailed findings, including a comparison between MLP and k-NN classifiers, providing insights into distinct modeling approaches.

4. Results and Analysis

4.1. Results

A comparative analysis between the MLP classifier and the k-NN classifier revealed intriguing insights into their respective performances. The table below outlines the accuracy scores achieved by both classifiers and the baseline models, providing a basis for understanding the effectiveness of different modeling techniques and demonstrating the advancement in accuracy with respect to the baseline models[2]:

Patch_Size	MLP (%)	KNN_Optimised (%)	KNN_Baseline (%)
30	58.3	51.7	47.6
50	77.45	75.8	71.7
90	97.3	95.6	93.2

Table 1: *The system performance for MLP classifier, Optimised K-NN classifier, and baseline model.*

4.2. Analysis

The results give us a good basis to thoroughly analyse the findings:

- **Accuracy Dominance:** Higher accuracy is consistently achieved by MLP compared to KNN across all evaluated patch sizes (30, 50, and 90 pixels). The improvement in accuracy is particularly pronounced for larger patch sizes, highlighting the effective capturing of complex patterns by MLP.
- **Performance Consistency:** Consistently strong performance is maintained by MLP, even with increasing patch sizes. This indicates MLP's ability to handle larger and more varied datasets, showing its adaptability to different complexities in image data.
- **Optimisation Impact:** Although the KNN classifier improves with optimisation over its baseline performance, MLP maintains a distinct advantage. MLP's accuracy, particularly for larger patches, exceeds both the optimised and baseline configurations of KNN.
- **Task Suitability:** The MLP classifier excels in face orientation estimation by effectively understanding intricate image relationships. Its hierarchical structure with hidden layers and neurons proves effective in learning complex facial features, enhancing orientation accuracy.

5. Discussion and Conclusions

5.1. Conclusions

In summary, the results of the experiments showcase the effectiveness of the Multi-Layer Perceptron (MLP) classifier in face orientation estimation. Across varying patch sizes (30, 50, and 90 pixels), the MLP consistently outperforms the k-NN classifier and baseline models, achieving higher accuracy percentages. Notably, the accuracy improvement is more pronounced with larger patch sizes, indicating the MLP's ability to capture complex patterns effectively.

5.2. Discussion

The following future directions aim to build upon the current findings and advance the face orientation estimation system for more effective applications.

1. **Fine-Tuning Hyperparameters:** Further exploration and fine-tuning of hyperparameters could enhance the performance of both the MLP and KNN classifiers. Trying out various combinations of hyperparameters could help discover the best settings for better accuracy.
2. **Dataset Expansion:** Expanding the dataset by adding more diverse and larger samples could help the model generalise better and improve its overall performance. Including a wider variety of facial orientations and expressions can enhance the model's robustness.
3. **Exploring Alternative Architectures:** Exploring different types of neural network structures other than MLP might uncover models better tailored for face orientation tasks. Looking into architectures like convolutional neural networks (CNNs)[7], designed specifically for image-related tasks, could be advantageous.

6. References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson, 2008.

[2] J. Barker, "Com6018 assignment 2," University of Sheffield, 2023, this document provides instructions for assignment 2.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] scikit-learn. (2023) scikit-learn pca documentation. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[5] ——. (2023) scikit-learn mlpclassifier documentation. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.