

Improving Deep Unconstrained Facial Recognition by Data Augmentation



Arnauld Fountsop Nzegha, Jean Louis Ebongue Fendji, Christopher Thron, and Clementin Djameni Tayou

1 Introduction

Facial recognition technology aims to recognize the face of a (human or animal) subject by using biometrics to map facial features from a picture. This recognition is usually achieved through artificial intelligence techniques, mainly deep learning.

Deep learning is a subfield of machine learning which is based on hierarchical learning architectures for data representation [25]. It includes a set of methods that represent data with a high level of abstraction through nonlinear transformations. One of the strengths of deep learning lies in its ability to exploit technological advances in computing power. Today, deep architectures based on convolutional neural networks (CNN) form the basis of most facial biometrics technology, and are very robust [27, 29, 34, 37, 38, 38–41].

Many CNN models can achieve close to 100% accuracy on standard benchmarks such as Labeled Faces in the Wild (LFW), MegaFace, or YouTubeFace (YTF). However, these models cannot guarantee the same performance in a realistic environment which is uncontrolled. Facial recognition remains a big challenge when it is in a non-constrained scenario or without the cooperation of the subject. Indeed, the face of a person can be very different observed on images taken in various poses, lighting conditions, at different ages, with makeup or different

A. F. Nzegha (✉) · C. D. Tayou

Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon

J. L. E. Fendji

Department of Computer Engineering, Institute of Technology, University of Ngaoundere, Ngaoundere, Cameroon

C. Thron

Department of Science and Mathematics, Texas A&M University-Central Texas, Killeen, TX, USA

© Springer Nature Switzerland AG 2020

S. Subair, C. Thron (eds.), *Implementations and Applications of Machine Learning*, Studies in Computational Intelligence 782,

https://doi.org/10.1007/978-3-030-37830-1_7

facial expressions. These hazards lead to a great intra-personal variability which is the major difficulty of facial recognition without constraint. In particular, lighting variations can significantly reduce the performance of face recognition systems. The present chapter tackles this issue by using data augmentation that makes it possible to overcome the problem of intra-personal variability by inserting more diversity in the facial recognition data. Lighting compensation allows facial recognition models to better capture the dynamics of lighting and greatly improves recognition.

This chapter is organized as follows. Section 2 describes elements of the system design, including the image processing techniques used for introducing lighting variation as well as techniques used to train the CNN. Section 3 describes the evaluation of the approach using a convolutional neural network model inspired by VGGNet-16 [35], that was trained on the LFW dataset and then tested on the YaleB and ORL databases. Section 4 presents and discusses the results of the experiments described in Sect. 3; and Sect. 5 gives some concluding remarks.

2 Facial Recognition System Design Elements

2.1 Overview

The implementation of our improved facial recognition system can be divided into two main tasks: data augmentation and CNN training. These two main tasks may be subdivided as follows.

Data augmentation is accomplished by modifying images in the training set so as to simulate the effects of different lighting situations on the appearance of the face in the image. This simulation requires first that a 3D reconstruction of the face be inferred from the image and then a lighting model applied to the 3D reconstruction to perform a pixel-by-pixel alteration of the apparent brightness of the image pixels.

CNN training involves using training sets to train the CNN to make a correct identification when presented with a facial image of one of a limited number of subjects that are known to the CNN. This training also proceeds in two stages. First, a large training set from a wide variety of faces is used to train the CNN to recognize distinguishing “features,” which help to uniquely identify the image. This dataset is augmented by the data augmentation procedures described above, to improve the CNN’s robustness to variations in lighting. After this, a smaller dataset containing only images of the specific subjects to be identified (in various poses and lighting conditions) is used to familiarize the CNN so that it can make specific identifications.

The techniques used in the steps outlined above are described in the following subsections.

2.2 *Data Augmentation*

2.2.1 **Data Augmentation Overview**

Data augmentation attacks the problem of lighting variation from a different angle. Rather than modifying the original image before attempting recognition, data augmentation uses the unmodified image but increases the variation in the training set. This makes the system training more rigorous, and the trained network is better able to cope with lighting variations.

Early versions of augmentation for lighting variations used 2-D methods. The 2012 AlexNet and 2014 VGGNet systems for image recognition used principal component analysis (PCA) on the set of RGB values for all pixels and all images in the training set. Based on the eigenvalues and eigenvectors from PCA, a (mean-zero multivariate Gaussian distribution) was constructed in RGB space, and additional images were generated from a given training image by selecting a single vector from this distribution and adding it to the RGB values for all pixels of the given image [22, 35].

More recently, data augmentation methods for lighting compensation have been based on 3-D reconstructions, which enable more accurate estimation of the effects of lighting variation. Examples include Paysan et al. [30] and Jiang et al. [19]. Sixt et al. [36] propose a new framework called RenderGAN that can generate large datasets of realistic labeled images by combining a 3-D model and a GAN model.

2.2.2 **3-D Face Reconstruction**

3-D face reconstruction from a 2-D image is an important, long-standing problem in facial recognition. Apart from lighting compensation, 3-D facial surface models have been used in other applications such as 3-D facial expression recognition [42] and facial animation [31].

Point Clouds

Point clouds are the simplest type of surface representation, and forms the basis for most 3-D object representation methods. A point cloud consists of an unordered set of 3-dimensional vectors that represent points lying on the surface. As described below, more descriptive surface representations (e.g., mesh representations) are typically constructed based on point cloud representations.

Polygonal Meshes

In 3-D research in general and 3-D facial recognition in particular, most researchers represent the surfaces of 3-D objects as meshes. A mesh is essentially an unordered

set of vertices (points), edges (connections between two vertices), and faces (sets of edges with shared vertices) that together represent the surface explicitly. Generally, the faces consist of triangles, quadrilaterals, or other simple convex polygons, which simplifies the rendering. The task of building a mesh from a cloud of points is commonly called the surface reconstruction problem. Several techniques exist, many of which are based on the classic method of Delaunay triangulation [26].

Polygonal mesh surface representations provide several distinct advantages over point clouds. They enable much clearer representations of the surface using 3-D plotting software. Numerous ray tracing, collision detection, and rigid body dynamics algorithms have been developed for polygonal meshes. They provide explicit knowledge of connectivity, which can be used to calculate the geodetic distance between points on the surface. This is particularly useful in face recognition, because geodetic distances are more invariant than Euclidean distances under changes of expression by the subject [7]. Some researchers have exploited this by directly comparing point to point distances, or by using isogeodesic curves [18].

Meshes do have a significant drawback in that they may have errors such as cracks, holes, T-junctions, overlapping polygons, duplicate geometry, and auto-intersections. These defects may impede the mesh path and affect the quality of rendering. To correct these problems, either a local or global approach may be used, depending on the seriousness of the defects[2]. The local approach locates each specific defect in the mesh and tries to fix it while conserving the model unchanged, and consequently preserves most of the mesh information unchanged. It is suitable when meshes have rare defects and consist of operations such as triangulation for cracks, or filling holes [15], etc. On the other hand, the global approach takes into account both individual defects and the mutual relations between defects, and requires some adjustment for most or all mesh elements.

In cases where lighting of the surface is being considered, the mesh must include orientation information of the mesh faces. The orientation of a flat face is given by a vector of unit length that is normal (i.e., perpendicular) to the face. Any face has two unit normal vectors that point in opposite directions. In cases when the mesh is describing a human face, one normal points “outward” and the other points “inward.” As we shall see in Sect. 2.2.3, the brightness of a surface depends on the angle between the surface’s outward normal and the lighting source. It follows that in order to correctly predict brightness, the outward normal must be correctly identified. Algorithms have been designed to consistently identify outward normals: for example, Borodin et al. [6] propose and verify an algorithm for consistently orienting the normals of a boundary representation, even in the presence of gaps, T-junctions, and intersections.

3D Morphable Models

3D morphable models (3DMM) were introduced in 1999 by Blanz and Vetter [5]. They first created a database of 200 3-D face scans in standardized position using Cyberware™ laser scan technology, which captures comprehensive 3-dimensional

geometric and textural data. The i 'th face in the database is represented as a pair of $3n$ -dimensional vectors $(\mathbf{S}_i, \mathbf{T}_i)$, where \mathbf{S}_i and \mathbf{T}_i capture the geometric and texture (RGB) information, respectively. Additional faces can then be modeled as ‘‘morphs’’ of the faces in this database, which represented as linear combinations of the face vectors in the database. The mathematical expression is

$$(\mathbf{S}_{morph}, \mathbf{T}_{morph}) = \left(\sum_{i=1}^m a_i \mathbf{S}_i, \sum_{i=1}^m b_i \mathbf{T}_i \right), \quad (1)$$

where m is the number of faces in the database, and $\{a_i\}$ and $\{b_i\}$ are sets of coefficients such that $\sum_{i=1}^m a_i = \sum_{i=1}^m b_i = 1$ (coefficients can be positive or negative). The two sets of coefficients reflect the fact that geometry and texture are modeled independently.

UV-Mapping and 3D Face

3D models must contain both 3D shape and texture information. Typically, the shape information is given by a 3D mesh as described above. Since the texture information describes properties of a surface, the texture information is inherently two-dimensional and may be stored in a two-dimensional array. However, since the surface is not flat, a mapping from \mathbb{R}^2 to \mathbb{R}^3 is required to associate two-dimensional points with the actual positions in 3-dimensional space. This mapping parametrizes the facial surface in terms of a pair of coordinates, which are generally denoted as (U, V) . The texture values (R, G, B) are then stored as functions of (U, V) . Feng et al. [12] have developed a facial recognition system that uses a CNN to regress the (U, V) position map directly from unconstrained 2D images. Their approach attains state-of-the-art performance, with much lower processing times than other methods. In their system, the reconstructed 3D face is represented by:

- 43868 3D vertices, where each vertex has x, y, z coordinates.
- 43868 UV mapping coordinates corresponding to the 3D vertices.
- 86907 triangular faces, which form a triangulation of the 3D vertices.
- UV texture information of size $256 \times 256 \times 3$ (RGB values for a 256×256 grid in the UV plane).

2.2.3 Lighting Variation

In this section we describe the modification of images used to augment the dataset in order to account for lighting variations. We assume that our face surface is a Lambertian surface, so can reflect the light following the Lambert reflectance model [4]. The overall process for introducing lighting variation is shown in Fig. 1.

The first step in the process shown in the figure is to calculate the normal vectors at all vertices of the 3D triangulation, which may be used to calculate the light



Fig. 1 Process of illumination variation

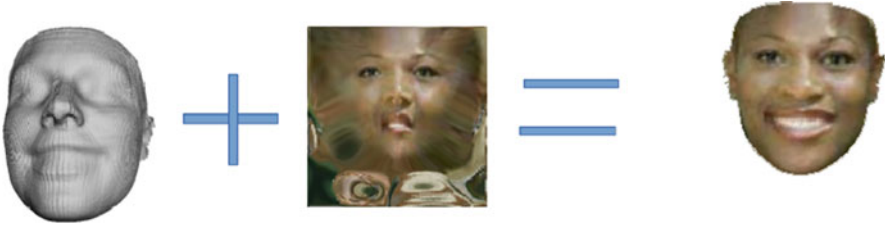


Fig. 2 Fusion of the mesh and the UV-texture

intensity at each vertex. Lighting values at surface points on the triangular faces can then be interpolated based on vertex values using barycentric coordinates. In order to be able to calculate the vertex normals, we first calculate unit normal vectors (i.e., perpendiculars) to the triangular surfaces that meet at the vertex. For a triangular surface F of vertices (v_1, v_2, v_3) , a normal vector to the plane of F may be calculated as a cross product of two edge vectors, $N_f = \overrightarrow{v_1v_2} \times \overrightarrow{v_1v_3}$ (the vertices in the mesh must be correctly ordered so that the outward rather than the inward normal is produced, as discussed in Sect. 2.2.2).

Given N_f , the unit outward normal vector to a surface is found as

$$\hat{N}_f = \frac{N_f}{|N_f|}. \quad (2)$$

We estimate the unit normal at the vertex v_i by the normalized sum of the normals to the surfaces containing v_i as follows:

$$N_{v_i} = N(v_i) = \frac{\sum \hat{N}_f}{|\sum \hat{N}_f|} \quad \text{for all } f \text{ such that } v_i \in f. \quad (3)$$

To calculate the intensities at the vertices, we merge the 3D shape and the UV-texture by using the UV-mapping. This consists in associating the vertex v_i to the corresponding pixel (U_i, V_i) in the UV plane (Fig. 2). The intensity information $I(U_i, V_i)$ is used as the initial intensity for vertex v_i .

Compute the Intensities

In 3D space, we randomly choose a point $L_{x,y,z}$ as a lighting source. The unit direction vector from L to the vertex v_i is denoted by $l_i = \frac{\overrightarrow{Lv_i}}{|\overrightarrow{Lv_i}|}$. We then

apply the Lambert reflectance model, so that the modified intensity of v_i is given by

$$I(v_i) = I(U_i, V_i) \cdot I_i \cos(\theta), \quad (4)$$

where I_i is the relative intensity of the lighting source and θ is the angle between l_i and N_{v_i} . In our model, I_i was chosen as the constant value 1.5 independent of i .

The cosine function is a decreasing function. If the angle between θ is too large, the intensity will vanish or even become negative. To avoid the loss of information, we add to the new intensity 50% of the initial intensity.

$$I(v_i) = I_0(v_i)I_i \cos(\theta) + \frac{1}{2}I_0(v_i) \quad (5)$$

This equation was used to compute the intensity for all vertices. For each triangle in the mesh, 13 additional points P_1, \dots, P_{13} are created within the triangle according to the equation:

$$P_i = \sum_{j=1}^3 w_{ij} v_j = 1, i = 1 \dots 13, \quad (6)$$

where v_1, v_2, v_3 are the vertices of the triangle and $\{w_{i1}, w_{i2}, w_{i3}\}_{i=1, \dots, 13}$ represent 13 sets of barycentric weights where $w_{ij} \geq 0$ and $\sum_{j=1}^3 w_{ij} = 1, i = 1 \dots 13$. The intensity at point P_i is then calculated as

$$I(P_i) = \sum_{j=1}^3 w_{ij} I(v_j). \quad (7)$$

The 3D face has three coordinates (x, y, z) , where z represents the depth. Given that the 3D face is aligned, the coordinates (x, y) give a 2D plane containing the face (Fig. 3).

2.3 CNN Training for Classification

2.3.1 Overview

Face recognition (and object classification in general) can be separated into two steps: features extraction and classification. The two tasks can be performed separately: using transfer learning, a pre-trained feature extractor can be used, associated with an MLP or another classifier for classification, as described in the following subsections.



Fig. 3 Illustration of illumination variation

2.3.2 Features Extraction

Features extraction consists of finding a set of quantities calculated from a dataset (called features) that capture the datasets' essential characteristics. Classical methods of feature extraction include PCA [20] and linear discriminant analysis (LDA) [3]. CNN has recently emerged as the most successful feature extractor for images [44], etc. With CNN, an image is passed through a succession of filters arranged in layers which successively transform and reduce the data, creating representations of the image called *feature maps*. These feature maps can be concatenated or "flattened" into a single vector, called a *features vector*.

Classification

The classification step assigns the image to a predefined class, based on the features vector that is calculated by the features extractor. Classifiers in the literature include k-nearest neighbors (KNN) [43], support vector machines (SVM) [10], decision trees [32], and multilayer perceptron (MLP) [11]. In this work we use the MLP, which is a classifier based on neural networks, where the neurons of one layer are fully connected to those of the next layer.

Transfer Learning

A challenge of image classification (and classification tasks in general) is to train a classifier for images from a particular source domain when representative training data is scarce. Sometimes it is possible to find a related domain where sufficient training data is available to develop an effective classifier. In such cases, it is

reasonable to suppose that a feature extractor that works well in a related domain will also work well in the source domain of interest. All that remains is to replace the classifier for the related domain with another that is particularly adapted to the source domain. This technique of reusing feature extractors is called *transfer learning* [28], and is a widely used strategy in the field of machine learning.

3 Experimental Setup

3.1 Computational Platform

For our system we used the Google Colaboratory platform [8], which is a cloud service based on Jupyter Notebooks designed support machine learning. It provides a fully configured runtime environment for deep learning, as well as free access to a robust graphics processor and TPU (tensor processing units). Currently it has NVIDIA GPUs with a preinstalled CUDA environment and several machine learning frameworks including TensorFlow, Theano, and scikit-learn. We used the TensorFlow framework [1] to train our model. TensorFlow is a programming framework for numerical computation that was made open source by Google Brain in November 2015. By 2017, TensorFlow was the most popular open source machine learning project on GitHub, even though it had only been available for little over a year [23]. To date it is one of the most powerful tools for Deep Learning, largely because of the ease of manipulation of tensors and their parallelization by tools such as CUDA.

Dlib-ml [21] is a state-of-the-art C++ toolkit containing machine-learning algorithms and data analysis tools, intended for both engineers in industry and researchers. We used Dlib to detect and extract the area of the face. This detector offers two methods for face detection: histogram of oriented gradients and support vector machine (HOG + SVM), and CNN-based detection. We used CNN-based detection: despite the heavy computing power required, since it is more suitable for non-frontal face detection. Results from [9] showed that the cascade algorithms with a CNN was superior to HOG + SVM or Haar Cascade methods in terms of both accuracy and speed criteria in unconstrained face detection problems.

PRNet (position map regression network) is an implementation of the system of Feng et al. [12] for regression of 3DMM model parameters. Initially this TensorFlow-based library was designed for pose estimation, facial alignment, and texture editing on the basis of 3D representation of the face. We use PRNet for 3D reconstruction of the different faces of the LFW dataset.

3.2 Description of CNN Model

We use a CNN for features extraction, therefore a stack of convolutional layers alternated with the pooling layers.

3.2.1 Inputs

We worked with grayscale images of size 100×100 . All outputs from all convolutional layers undergo linear rectification using the ReLU activation function.

3.2.2 Filters

We use 2D convolutional filters of size 3×3 , since the inputs are not very large. Small filters make it possible to detect highly localized patterns. The first convolutional layer has 64 filters applied to the input image, and the number of filters is doubled after each pooling layer. This process was inspired by the VGG16 [35] network of the group *Visual Geometric Group* which has a similar architecture. Unlike VGG16, we added a batch normalization [17] before every convolutional layer to avoid internal covariate shift which can slow down training and degrade performance. We also used dropout for regulation and greater robustness. Since [14] specifies that dropout has limited benefits when applied to convolutional layers, we applied dropout only on the first fully connected layer.

3.2.3 Subsampling (Pooling)

The feature extractor is subdivided into five blocks of convolutional layers separated by pooling layers. The first two blocks have two convolution layers followed by a pooling layer, because these layers produce larger feature maps and require more pooling. After this, subsequent blocks stack four convolutional layers before pooling. We chose to use 2×2 max pooling, which decreases each dimension of the feature maps by two (Fig. 4).

For identification, we use two fully connected layers of 1024 units each. A final softmax layer produces outputs in the interval $[0, 1]$: the number of neurons in the final layer is equal to the number of labels in the dataset. The outputs are interpreted as posterior probabilities of each individual. After the first fully connected layer we insert a dropout function that randomly eliminates 25% of the layer's outputs.

System hyperparameters are summarized in Table 1.

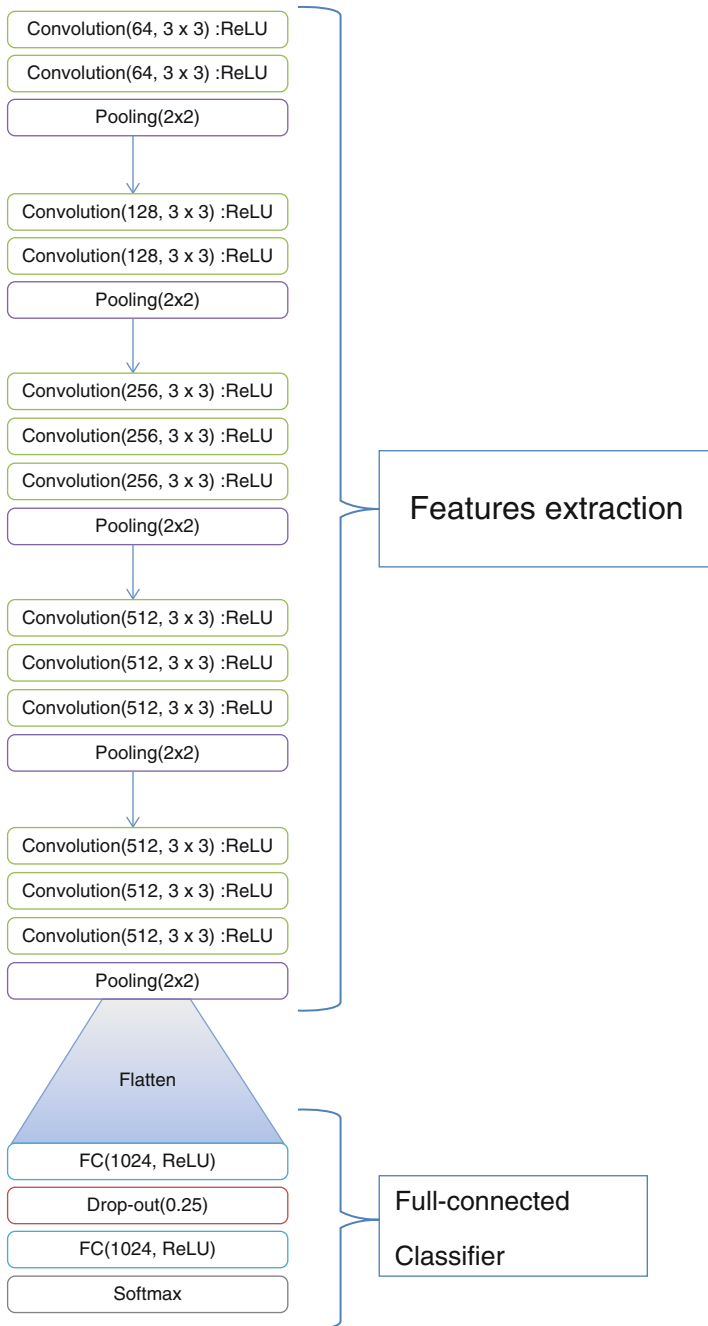


Fig. 4 Proposed model of convolutional network. The number of softmax outputs is set equal to the number of distinct individuals to be identified

Table 1 Setting of CNN

Hyperparameters	Details
Activation	ReLU
Initialization of weights	Random uniform
Optimizer	Adam
Number of iteration on LWF	1000
Number of iteration on YaleB and ORL	500

3.3 Datasets Used

3.3.1 Labeled Faces in the Wild (LFW)

This dataset is a collection of JPEG images of the faces of famous people collected on the internet [16, 24]. These faces were extracted from various online websites by face detectors based on the Viola–Jones model. LFW is commonly used to evaluate the performance of facial recognition systems. It contains 13,233 250×250 images of 5749 tagged celebrities, where each image is centered on a single face. In our experiments, the LFW dataset was used for pretraining the model. The data augmentation described in Sect. 2.2 was applied to these images to improve robustness of the feature extractor to lighting variations. The feature extractor was retained, and the classifier was replaced with classifiers for the smaller datasets described below. Since the smaller datasets have grayscale images, the LFW images were converted to grayscale using the function `cv2.cvtColor` from openCV.

3.3.2 ORL Database

The ORL Database [33] from the University of Cambridge contains 400 112×92 pixel images of 40 people, with 10 images per person. The images were taken at different times, with differing lighting and facial expressions. All the images were taken on a dark and homogeneous background, the subjects being in a frontal position, with a tolerance for certain lateral movements. In our experiment, the ORL database, like the YaleB database, was used for retraining the classifier.

3.3.3 Yale Face Database B

YaleB [13] was developed to allow systematic testing of facial recognition methods under wide variations in lighting and pose. It contains 5760 640×480 pixel images of 10 subjects viewed each in 576 viewing conditions (9 poses \times 64 lighting conditions). For each subject in a particular pose, an image with ambient lighting (background) was also captured. In our experiment, a selection of images from YaleB was used for retraining the classifier, according to the transfer learning methodology described in Sect. 2.3.

3.4 Experimental Training and Testing Configurations

3.4.1 Experiment 1: LFW Without Data Augmentation

For the first experiment, we randomly selected 12,600 images from the LFW database for training, corresponding to 5749 subjects. For the initial feature extractor and classifier training, a classifier with 5749 softmax outputs was trained over 1000 iterations. Of the training set images, 10% (1260 images) were used for cross-validation.

After this, the classifier was removed and replaced by a classifier for ORL data and then for YaleB data. The new classifiers had the same number of layers as the original, but fewer softmax outputs (40 outputs for ORL, 10 outputs for YaleB). For training the ORL classifier, from the 400 images in the database 300 images were used for training and 100 for validation over 500 iterations. For training the YaleB classifier, 8 images per subject were used for training and 3 images per subject were used for validation, over 500 iterations. The three images chosen for evaluation had very uneven light distributions: one illuminated only on the left, one only on the right, and a third image with a central lighting.

3.4.2 Experiment 2: LFW with Data Augmentation

The second experiment differed from the first only in the training set used for the initial feature extractor + classifier training. The training set used 700 images from LFW of 28 subjects (25 images per subject). Each of these images was subjected to 18 different lighting conditions (representing a range of lighting directions, from left to right), for a total of 12,600 images. Since 28 subjects were used, the classifier in this case had 28 outputs. As in Experiment 1, after pretraining the classifier was replaced with a classifier of 40 outputs for ORL and with a classifier of 10 outputs for YaleB.

4 Results and Interpretation

4.1 Evaluation on ORL

Figures 5 and 6 show the training and testing accuracy curves for ORL data in Experiment 1 (without augmentation) and Experiment 2 (with augmentation), respectively. The figures show roughly 10% improvement in accuracy when augmented data is used for pretraining.

Fig. 5 Accuracy curve using ORL data for the model pretrained with plain LFW data (Experiment 1)

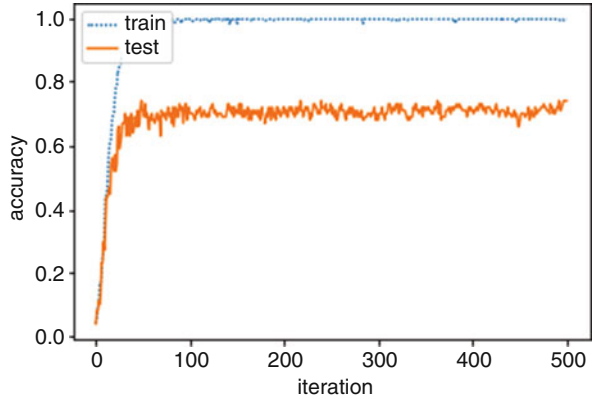
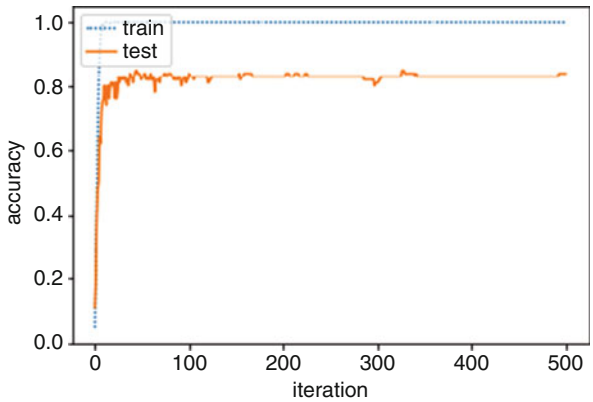


Fig. 6 Accuracy curve using ORL data for the model pretrained with augmented LFW data (Experiment 2)



4.2 Evaluation on YaleB

Figures 7 and 8 show the training and testing accuracy curves for YaleB data in Experiment 1 (without augmentation) and Experiment 2 (with augmentation), respectively. In this case, the improvement in test accuracy when augmented data is used for pretraining is about 20%.

Table 2 summarizes the results from the two experiments. For the evaluation of the model on test data, we observe a gain of accuracy of 9% on ORL. This finding confirms the enhanced effectiveness of pretraining that uses more images of fewer subjects taken under variable lighting conditions. The 18% improvement on YaleB shows that the performance improvement is further amplified in cases where the system is applied to images that also have variable lighting. In both experiments, the accuracy on YaleB is lower than the corresponding accuracy on ORL: this is probably due to the greater variability of both position and lighting for images in YaleB, as well as the smaller size of the dataset used in our experiments.

Fig. 7 Accuracy curve using YaleB for the model pretrained with plain LFW data (Experiment 1)

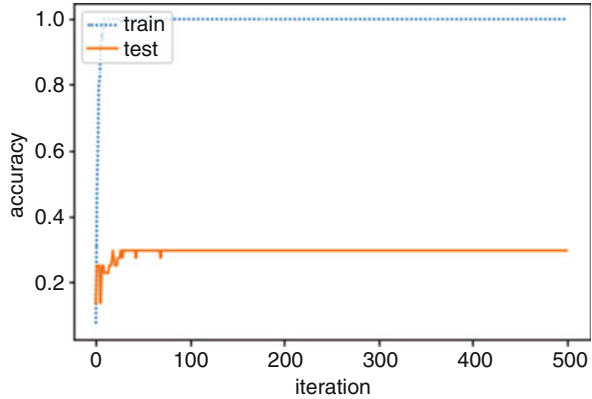


Fig. 8 Accuracy curve using YaleB for the model pretrained with augmented LFW data (Experiment 2)

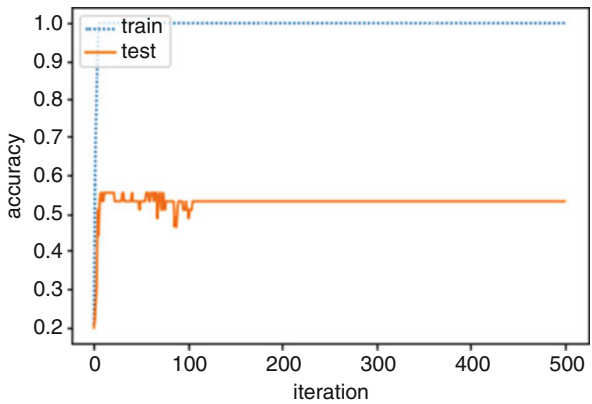


Table 2 Summary of accuracies and average errors observed for the ORL and YaleB data sets, with and without data augmentation

	ORL		YaleB	
	Accuracy	Average error	Accuracy	Average error
Simple data	74%	1.78	35.56%	6.03
Augmented data	83%	0.72	53.33%	3.12

5 Conclusion

This chapter demonstrates a practical method for improving deep facial recognition using data augmentation. Specifically, we employed 3D lighting variation as a method of data augmentation, using the Lambert reflectance model to model the dynamics of ambient lighting in 3D space. The improvement was verified on two different datasets possessing different degrees of image variability. Accuracy gain due to data augmentation ranged from 9% to 18%, with the greater gain observed in the dataset that showed more variability. We conclude that lighting variation using the Lambert reflectance model is well suited to data augmentation for deep facial recognition under unconstrained lighting conditions.

References

1. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., TensorFlow: a system for large-scale machine learning, in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016), pp. 265–283
2. M. Attene, M. Campen, L. Kobbelt, Polygon mesh repairing: an application perspective. *ACM Comput. Surv. (CSUR)* **45**(2), 15 (2013)
3. S. Balakrishnama, A. Ganapathiraju, Linear discriminant analysis—a brief tutorial. *Inst. Signal Inf. Process.* **18**, 1–8 (1998)
4. R. Basri, D.W. Jacobs, Lambertian reflectance and linear subspaces. *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(2), 218–233 (2003)
5. V. Blanz, T. Vetter et al., A morphable model for the synthesis of 3d faces, in *SIGGRAPH '99* (1999), pp. 187–194
6. P. Borodin, G. Zachmann, R. Klein, Consistent normal orientation for polygonal meshes, in *Proceedings Computer Graphics International, 2004* (IEEE, Piscataway, 2004), pp. 18–25
7. A.M. Bronstein, M.M. Bronstein, R. Kimmel, Expression-invariant 3d face recognition, in *International Conference on Audio- and video-based Biometric Person Authentication* (Springer, Berlin, 2003), pp. 62–70
8. T. Carneiro, R.V.M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V.H.C. De Albuquerque, P.P. Reboucas Filho, Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications. *IEEE Access* **6**, 61677–61685 (2018)
9. E. Cengil, A. Çınar, Comparison of Hog (histogram of oriented gradients) and Haar Cascade algorithms with a convolutional neural network based face detection approach. *Comput. Sci.* **3**(5), 244–255 (2017)
10. C. Cortes, V. Vapnik, Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
11. Q.-K. Do, A. Allauzen, F. Yvon, Modèles de langue neuronaux: une comparaison de plusieurs stratégies d'apprentissage, in *Actes de la 21e conférence sur le traitement automatique des langues naturelles (TALN)* (2014), pp. 256–267
12. Y. Feng, F. Wu, X. Shao, Y. Wang, X. Zhou, Joint 3d face reconstruction and dense alignment with position map regression network, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 534–551
13. A.S. Georghiadis, P.N. Belhumeur, D.J. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(6), 643–660 (2001)
14. G. Ghiasi, T.-Y. Lin, Q.V. Le, DropBlock: a regularization method for convolutional networks, in *Advances in Neural Information Processing Systems* (2018), pp. 10727–10737
15. X. Guo, J. Xiao, Y. Wang, A survey on algorithms of hole filling in 3d surface reconstruction. *Vis. Comput.* **34**(1), 93–103 (2018)
16. G.B. Huang, M. Mattar, T. Berg, E. Learned-Miller, Labeled faces in the wild: a database for studying face recognition in unconstrained environments, in *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition* (2008)
17. S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). Preprint. arXiv: 1502.03167
18. S. Jahanbin, H. Choi, Y. Liu, A.C. Bovik, Three dimensional face recognition using iso-geodesic and iso-depth curves, in *2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems* (IEEE, Piscataway, 2008), pp. 1–6
19. D. Jiang, Y. Hu, S. Yan, L. Zhang, H. Zhang, W. Gao, Efficient 3d reconstruction for face recognition. *Pattern Recogn.* **38**(6), 787–798 (2005)
20. I. Jolliffe, *Principal Component Analysis* (Springer, New York, 2011)
21. D.E. King, Dlib-ml: a machine learning toolkit. *J. Mach. Learn. Res.* **10**(Jul), 1755–1758 (2009)

22. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
23. J. Lawrence, J. Malmsten, A. Rybka, D.A. Sabol, K. Triplin, Comparing TensorFlow deep learning performance using CPUs, GPUs, local PCs and cloud, in *Proceedings of Student-Faculty Research Day, CSIS, Pace University* (2017)
24. E. Learned-Miller, G.B. Huang, A. RoyChowdhury, H. Li, G. Hua, Labeled faces in the wild: a survey, in *Advances in Face Detection and Facial Image Analysis* (Springer, Cham, 2016), pp. 189–248
25. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**(7553), 436 (2015)
26. S.P. Lim, H. Haron, Surface reconstruction techniques: a review. *Artif. Intell. Rev.* **42**(1), 59–78 (2014)
27. X. Liu, M. Kan, W. Wu, S. Shan, X. Chen, VIPLFaceNet: an open source deep face recognition SDK. *Front. Comput. Sci.* **11**(2), 208–218 (2017)
28. S.J. Pan, Q. Yang, A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **22**(10), 1345–1359 (2009)
29. O.M. Parkhi, A. Vedaldi, A. Zisserman et al., Deep face recognition, in *BMVC*, vol. 1 (2015), p. 6
30. P. Paysan, R. Knothe, B. Amberg, S. Romdhani, T. Vetter, A 3d face model for pose and illumination invariant face recognition, in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance* (IEEE, Piscataway, 2009), pp. 296–301
31. J. Roth, Y. Tong, X. Liu, Unconstrained 3d face reconstruction, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2606–2615
32. S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **21**(3), 660–674 (1991)
33. F.S. Samaria, Face recognition using hidden Markov models. PhD thesis, University of Cambridge, Cambridge, UK, 1994
34. F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 815–823
35. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2015). Preprint. arXiv: 1409.1556v6
36. L. Sixt, B. Wild, T. Landgraf, RenderGAN: generating realistic labeled data. *Front. Robot. AI* **5**, 66 (2018)
37. Y. Sun, Y. Chen, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, in *Advances in Neural Information Processing Systems* (2014), pp. 1988–1996
38. Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1891–1898
39. Y. Sun, D. Liang, X. Wang, X. Tang, Deepid3: face recognition with very deep neural networks (2015). Preprint. arXiv: 1502.00873
40. Y. Sun, X. Wang, X. Tang, Deeply learned face representations are sparse, selective, and robust, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2892–2900
41. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: closing the gap to human-level performance in face verification, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708
42. J. Wang, L. Yin, X. Wei, Y. Sun, 3d facial expression recognition based on primitive surface feature distribution, in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2 (IEEE, Piscataway, 2006), pp. 1399–1406
43. K.Q. Weinberger, J. Blitzer, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, in *Advances in Neural Information Processing Systems* (2006), pp. 1473–1480
44. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *European Conference on Computer Vision* (Springer, Cham, 2014), pp. 818–833