Surpassing Human-Level Face Verification Performance on LFW with GaussianFace

Chaochao Lu Xiaoou Tang
Department of Information Engineering, The Chinese University of Hong Kong
{lc013, xtang}@ie.cuhk.edu.hk

Abstract

Face verification remains a challenging problem in very complex conditions with large variations such as pose, illumination, expression, and occlusions. This problem is exacerbated when we rely unrealistically on a single training data source, which is often insufficient to cover the intrinsically complex face variations. This paper proposes a principled multi-task learning approach based on Discriminative Gaussian Process Latent Variable Model, named GaussianFace, to enrich the diversity of training data. In comparison to existing methods, our model exploits additional data from multiple source-domains to improve the generalization performance of face verification in an unknown target-domain. Importantly, our model can adapt automatically to complex data distributions, and therefore can well capture complex face variations inherent in multiple sources. Extensive experiments demonstrate the effectiveness of the proposed model in learning from diverse data sources and generalize to unseen domain. Specifically, the accuracy of our algorithm achieves an impressive accuracy rate of 98.52% on the well-known and challenging Labeled Faces in the Wild (LFW) benchmark [23]. For the first time, the human-level performance in face verification (97.53%) [28] on LFW is surpassed.

1. Introduction

Face verification, which is the task of determining whether a pair of face images are from the same person, has been an active research topic in computer vision for decades [28, 22, 46, 5, 47, 31, 14, 9]. It has many important applications, including surveillance, access control, image retrieval, and automatic log-on for personal computer or mobile devices. However, various visual complications deteriorate the performance of face verification, as shown by numerous studies on real-world face images from the wild [23]. The Labeled Faces in the Wild (LFW) dataset is well known as a challenging benchmark for face verification. The dataset provides a large set of relatively unconstrained face images with complex variations in pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters. Not surprisingly, LFW has proven difficult for automatic face verification methods [23, 28]. Although there has been significant work [22, 9, 5, 14, 47, 13, 59, 50, 51, 53] on LFW and the accuracy rate has been improved from 60.02% [56] to 97.25% [53] since LFW is established in 2007, these studies have not closed the gap to human-level performance [28] in face verification.

Why could not we surpass the human-level performance? Two possible reasons are found as follows:

1) Most existing face verification methods assume that the training data and the test data are drawn from the same feature space and follow the same distribution. When the distribution changes, these methods may suffer a large performance drop [58]. However, many practical scenarios involve cross-domain data drawn from different facial appearance distributions. Learning a model solely on a single source data often leads to overfitting due to dataset bias [55]. Moreover, it is difficult to collect sufficient and necessary training data to rebuild the model in new scenarios, for highly accurate face verification specific to the target domain. In such cases, it becomes critical to exploit more data from multiple source-domains to improve the generalization of face verification methods in the target-domain.

2) Modern face verification methods are mainly divided into two categories: extracting low-level features [36, 3, 34, 10, 24], and building classification models [62, 50, 13, 37, 31, 56, 5, 28, 47, 33]. Although these existing methods have made great progress in face verification, most of them are less flexible when dealing with complex data distributions. For the methods in the first category, for example, low-level features such as SIFT [36], LBP [3], and Gabor [34] are handcrafted. Even for features learned from data [10, 24], the algorithm parameters (such as the depth of random projection tree, or the number of centers in k-means) also need to be specified by users. Similarly,

for the methods in the second category, the architectures of deep networks in [62, 50, 63, 51] (for example, the number of layers, the number of nodes in each layer, etc.), and the parameters of the models in [31, 5, 28, 47] (for example, the number of Gaussians, the number of classifiers, etc.) must also be determined in advance. Since most existing methods require some assumptions to be made about the structures of the data, they cannot work well when the assumptions are not valid. Moreover, due to the existence of the assumptions, it is hard to capture the intrinsic structures of data using these methods.

To this end, we propose the Multi-Task Learning approach based on Discriminative Gaussian Process Latent Variable Model (DGPLVM) [57], named *GaussianFace*, for face verification. Unlike most existing studies [22, 5, 14, 47, 13] that rely on a single training data source, in order to take advantage of more data from multiple source-domains to improve the performance in the target-domain, we introduce the multi-task learning constraint to DGPLVM. Here, we investigate the asymmetric multi-task learning because we only focus on the performance improvement of the target task. From the perspective of information theory, this constraint aims to maximize the mutual information between the distributions of target-domain data and multiple source-domains data. Moreover, the GaussianFace model is a reformulation based on the Gaussian Processes (GPs) [42], which is a non-parametric Bayesian kernel method. Therefore, our model also can adapt its complexity flexibly to the complex data distributions in the real-world, without any heuristics or manual tuning of parameters.

Reformulating GPs for large-scale multi-task learning is non-trivial. To simplify calculations, we introduce a more efficient equivalent form of Kernel Fisher Discriminant Analysis (KFDA) to DGPLVM. Despite that the GaussianFace model can be optimized effectively using the Scaled Conjugate Gradient (SCG) technique, the inference is slow for large-scale data. We make use of GP approximations [42] and anchor graphs [35] to speed up the process of inference and prediction, so as to scale our model to large-scale data. Our model can be applied to face verification in two different ways: as a binary classifier and as a feature extractor. In the former mode, given a pair of face images, we can directly compute the posterior likelihood for each class to make a prediction. In the latter mode, our model can automatically extract high-dimensional features for each pair of face images, and then feed them to a classifier to make the final decision.

The main contributions of this paper are as follows:

- We propose a novel GaussianFace model for face verification by virtue of the multi-task learning constraint to DGPLVM. Our model can adapt to complex distributions, avoid over-fitting, exploit discriminative information, and take advantage of multiple source-domains data.
- We introduce a computationally more efficient equivalent form of KFDA to DGPLVM. This equivalent form reformulates KFDA to the kernel version consistent with the covariance function in GPs, which greatly simplifies calculations.
- We introduce approximation in GPs and anchor graphs to speed up the process of inference and prediction.
- We achieve superior performance on the challenging LFW benchmark [23], with an accuracy rate of 98.52%, beyond human-level performance reported in [28].

2. Related Work

Human and computer performance on face recognition has been compared extensively [40, 38, 2, 54, 41, 8]. These studies have shown that computer-based algorithms were more accurate than humans in well-controlled environments (e.g., frontal view, natural expression, and controlled illumination), whilst still comparable to humans in the poor condition (e.g., frontal view, natural expression, and uncontrolled illumination). However, the above conclusion is only verified on face datasets with controlled variations, where only one factor changes at a time [40, 38]. To date, there has been virtually no work showing that computer-based algorithms could surpass human performance on unconstrained face datasets, such as LFW, which exhibits natural (multifactor) variations in pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters.

There has been much work dealing with multifactor variations in face verification. For example, Simonyan *et al.* applied the Fisher vector to face verification and achieved a good performance [47]. However, the Fisher vector is derived from the Gaussian mixture model (GMM), where the number of Gaussians need to be specified by users, which means it cannot cover complex data automatically. Li *et al.* proposed a non-parametric subspace analysis [33, 32], but it is only a linear transformation and cannot cover the complex distributions. Besides, there also exist some approaches for utilizing plentiful source-domain data. Based on the Joint Bayesian algorithm [13], Cao *et al.* proposed a transfer learning approach [9] by merging source-domain data with limited target-domain data. Since this transfer learning approach is based on the joint Bayesian model of original visual features, it is not suitable for handling the complex nonlinear data and the data with complex manifold structures. Moreover, the transfer learning approach in [9] only considered two different domains, restricting its wider applications in large-scale data from multiple domains. More recently, Zhu *et al.* [63] learned the transformation from face images under various poses and lighting conditions to a canonical view with a deep convolutional network. Sun *et al.* [51] learned face representation with a deep model through face identification, which is a challenging multi-class prediction task. Taigman *et al.* [52] first utilized explicit 3D face modeling to apply a piecewise affine transformation, and then derived a face representation from a nine-layer deep neural network. Although these methods have achieved high performances on LFW, many parameters of them must be determined in advance so that they are less flexible when dealing with complex data distributions.

The core of our algorithm is GPs. To the best of our knowledge, GPs methods and Multi-task learning with related GPs methods (MTGP) have not been applied for face verification. Actually, MTGP/GPs have been extensively studied in machine learning and computer vision in recent years [6, 60, 11, 25, 30, 44, 49, 61, 26]. However, most of them [60, 11, 6, 44, 25, 49, 61] have only considered the symmetric multi-task learning, which means that all tasks have been assumed to be of equal importance, whereas our purpose is to enhance performance on a target task given all other source tasks. Leen *et al.* proposed a MTGP model in the asymmetric setting [30] to focus on improving performance on the target task, and Kim *et al.* developed a GP model for clustering [26], but their methods do not take the discriminative information of the covariance function into special account like DGPLVM. Although the discriminative information is considered in [57], it does not apply multi-task learning to improve its performance. Salakhutdinov *et al.* used a deep belief net to learn a good covariance kernel for GPs [45]. The limitation of such deep methods is that it is hard to determine which architecture for this network is optimal. Also, multi-task learning constraint was not considered in [45].

3. Preliminary

In this section, we briefly review Gaussian Processes (GPs) for classification and clustering [26], and Gaussian Process Latent Variable Model (GPLVM) [29]. We use GPs method mainly due to the following three notable advantages. Firstly, as mentioned previously, it is a non-parametric method, which means it adapts its complexity flexibly to the complex data distributions in the real-world, without any heuristics or manual tuning of parameters. Secondly, GPs method can be computed effectively because of its closed-form marginal probability computation. Furthermore, its hyper-parameters can be learned from data automatically without using model selection methods such as cross validation, thereby avoiding the high computational cost. Thirdly, the inference of GPs is based on Bayesian rules, resulting in robustness to overfitting. We recommend Rasmussen and Williams's excellent monograph for further reading [42].

3.1. Gaussian Processes for Binary Classification

Formally, for two-class classification, suppose that we have a training set $\mathcal{D}$ of N observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where the i-th input point $\mathbf{x}_i \in \mathbb{R}^D$ and its corresponding output y_i is binary, with $y = 1_i$ for one class and $y_i = -1$ for the other. Let $\mathbf{X}$ be the $N \times D$ matrix, where the row vectors represent all n input points, and $\mathbf{y}$ be the column vector of all n outputs. We define a latent variable f_i for each input point $\mathbf{x}_i$, and let $\mathbf{f} = [f_1, \ldots, f_N]^\top$. A sigmoid function $\pi(\cdot)$ is imposed to squash the output of the latent function into $[0, 1]$, $\pi(f_i) = p(y_i = 1|f_i)$. Assuming the data set is i.i.d, then the joint likelihood factorizes to

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^N p(y_i|f_i) = \prod_{i=1}^N \pi(y_i f_i). \tag{1}$$

Moreover, the posterior distribution over latent functions is

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{X}, \mathbf{y}|\boldsymbol{\theta})}. \tag{2}$$

Since neither $p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ nor $p(\mathbf{y}|\mathbf{f})$ can be computed analytically, the Laplace method is utilized to approximate the posterior

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}(\hat{\mathbf{f}}, (\mathbf{K}^{-1} + \mathbf{W})^{-1}), \tag{3}$$

where $\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ and $\mathbf{W} = -\nabla\nabla \log p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})|_{\mathbf{f}=\hat{\mathbf{f}}}$. Then, we can obtain

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\hat{\mathbf{f}}^\top \mathbf{K}^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log|\mathbf{B}|. \tag{4}$$

where $|\mathbf{B}| = |\mathbf{K}| \cdot |\mathbf{K}^{-1} + \mathbf{W}| = |\mathbf{I}_n + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}|$. The optimal value of $\boldsymbol{\theta}$ can be acquired by using the gradient method to maximize Equation (4). Given any unseen test point x_*, the probability of its latent function f_* is

$$f_*|\mathbf{X}, \mathbf{y}, x_* \sim \mathcal{N}(\mathbf{K}_*\mathbf{K}^{-1}\hat{\mathbf{f}}, \mathbf{K}_{**} - \mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^\top), \tag{5}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \mathbf{W}^{-1}$. Finally, we squash f_* to find the probability of class membership as follows

$$\bar{\pi}(f_*) = \int \pi(f_*)p(f_*|\mathbf{X}, \mathbf{y}, x_*)\mathrm{d}f_*. \tag{6}$$

3.2. Gaussian Processes for Clustering

The principle of GP clustering is based on the key observation that the variances of predictive values are smaller in dense areas and larger in sparse areas. The variances can be employed as a good estimate of the support of a probability density function, where each separate support domain can be considered as a cluster. This observation can be explained from the variance function of any predictive data point x_*

$$\sigma^2(x_*) = \mathbf{K}_{**} - \mathbf{K}_* \tilde{\mathbf{K}}^{-1} \mathbf{K}_*^\top. \tag{7}$$

If x_* is in a sparse region, then $\mathbf{K}_* \tilde{\mathbf{K}}^{-1} \mathbf{K}_*^\top$ becomes small, which leads to large variance $\sigma^2(x_*)$, and vice versa. Another good property of Equation (7) is that it does not depend on the labels, which means it can be applied to the unlabeled data.

To perform clustering, the following dynamic system associated with Equation (7) can be written as

$$F(x) = -\nabla \sigma^2(x). \tag{8}$$

The theorem in [26] guarantees that almost all the trajectories approach one of the stable equilibrium points detected from Equation (8). After each data point finds its corresponding stable equilibrium point, we can employ a complete graph [4, 26] to assign cluster labels to data points with the stable equilibrium points. Obviously, the variance function in Equation (7) completely determines the performance of clustering.

3.3. Gaussian Process Latent Variable Model

Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$ denote the matrix whose rows represent corresponding positions of $\mathbf{X}$ in latent space, where $\mathbf{z}_i \in \mathbb{R}^d$ ($d \ll D$). The Gaussian Process Latent Variable Model (GPLVM) can be interpreted as a Gaussian process mapping from a low dimensional latent space to a high dimensional data set, where the locale of the points in latent space is determined by maximizing the Gaussian process likelihood with respect to $\mathbf{Z}$. Given a covariance function for the Gaussian process, denoted by $k(\cdot, \cdot)$, the likelihood of the data given the latent positions is as follows,

$$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^{ND} |\mathbf{K}|^D}} \exp\Big(-\frac{1}{2} \mathrm{tr}(\mathbf{K}^{-1} \mathbf{X}\mathbf{X}^\top) \Big), \tag{9}$$

where $\mathbf{K}_{i,j} = k(\mathbf{z}_i, \mathbf{z}_j)$. Therefore, the posterior can be written as

$$p(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{X}) = \frac{1}{\mathcal{Z}_a} p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{Z}) p(\boldsymbol{\theta}), \tag{10}$$

where $\mathcal{Z}_a$ is a normalization constant, the uninformative priors over $\boldsymbol{\theta}$, and the simple spherical Gaussian priors over $\mathbf{Z}$ are introduced [57]. To obtain the optimal $\boldsymbol{\theta}$ and $\mathbf{Z}$, we need to optimize the above likelihood (10) with respect to $\boldsymbol{\theta}$ and $\mathbf{Z}$, respectively.

4. GaussianFace

In order to automatically learn discriminative features or covariance function, and to take advantage of source-domain data to improve the performance in face verification, we develop a principled GaussianFace model by including the multi-task learning constraint into Discriminative Gaussian Process Latent Variable Model (DGPLVM) [57].

4.1. DGPLVM Reformulation

The DGPLVM is an extension of GPLVM, where the discriminative prior is placed over the latent positions, rather than a simple spherical Gaussian prior. The DGPLVM uses the discriminative prior to encourage latent positions of the same class to be close and those of different classes to be far. Since face verification is a binary classification problem and the GPs mainly depend on the kernel function, it is natural to use Kernel Fisher Discriminant Analysis (KFDA) [27] to model class structures in kernel spaces. For simplicity of inference in the followings, we introduce another equivalent formulation of KFDA to replace the one in [57].

KFDA is a kernelized version of linear discriminant analysis method. It finds the direction defined by a kernel in a feature space, onto which the projections of positive and negative classes are well separated by maximizing the ratio of the between-class variance to the within-class variance. Formally, let $\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_+}\}$ denote the positive class and $\{\mathbf{z}_{N_++1}, \ldots, \mathbf{z}_N\}$ the negative class, where the numbers of positive and negative classes are N_+ and $N_- = N - N_+$, respectively. Let $\mathbf{K}$ be the kernel matrix. Therefore, in the feature space, the two sets $\{\phi_{\mathbf{K}}(\mathbf{z}_1), \ldots, \phi_{\mathbf{K}}(\mathbf{z}_{N_+})\}$ and $\{\phi_{\mathbf{K}}(\mathbf{z}_{N_++1}), \ldots, \phi_{\mathbf{K}}(\mathbf{z}_N)\}$ represent the positive class and the negative class, respectively. The optimization criterion of KFDA is to maximize the ratio of the between-class variance to the within-class variance

$$J(\omega, \mathbf{K}) = \frac{(\mathbf{w}^\top(\mu_{\mathbf{K}}^+ - \mu_{\mathbf{K}}^-))^2}{\mathbf{w}^\top(\boldsymbol{\Sigma}_{\mathbf{K}}^+ + \boldsymbol{\Sigma}_{\mathbf{K}}^- + \lambda \mathbf{I}_N)\mathbf{w}}, \tag{11}$$

where λ is a positive regularization parameter, $\mu_{\mathbf{K}}^+ = \frac{1}{N_+}\sum_{i=1}^{N_+} \phi_{\mathbf{K}}(\mathbf{z}_i)$, $\mu_{\mathbf{K}}^- = \frac{1}{N_-}\sum_{i=N_++1}^{N} \phi_{\mathbf{K}}(\mathbf{z}_i)$, $\boldsymbol{\Sigma}_{\mathbf{K}}^+ = \frac{1}{N_+}\sum_{i=1}^{N_+}(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^+)(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^+)^\top$, and $\boldsymbol{\Sigma}_{\mathbf{K}}^- = \frac{1}{N_-}\sum_{i=N_++1}^{N}(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^-)(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^-)^\top$.

In this paper, however, we focus on the covariance function rather than the latent positions. To simplify calculations, we represent Equation (11) with the kernel function, and let the kernel function have the same form as the covariance function. Therefore, it is natural to introduce a more efficient equivalent form of KFDA with certain assumptions as Kim *et al.* points out [27], i.e., maximizing Equation (11) is equivalent to maximizing the following equation

$$J^* = \frac{1}{\lambda}\Big(\mathbf{a}^\top \mathbf{K}\mathbf{a} - \mathbf{a}^\top \mathbf{K}\mathbf{A}(\lambda \mathbf{I}_n + \mathbf{A}\mathbf{K}\mathbf{A})^{-1}\mathbf{A}\mathbf{K}\mathbf{a}\Big), \tag{12}$$

where

$$\mathbf{a} =[\frac{1}{n_+}\mathbf{1}_{N_+}^\top, -\frac{1}{N_-}\mathbf{1}_{N_-}^\top]$$
$$\mathbf{A} =\text{diag}\Big(\frac{1}{\sqrt{N_+}}\big(\mathbf{I}_{N_+} - \frac{1}{N_+}\mathbf{1}_{N_+}\mathbf{1}_{N_+}^\top\big), \frac{1}{\sqrt{N_-}}\big(\mathbf{I}_{N_-} - \frac{1}{N_-}\mathbf{1}_{N_-}\mathbf{1}_{N_-}^\top\big)\Big).$$

Here, $\mathbf{I}_N$ denotes the $N \times N$ identity matrix and $\mathbf{1}_N$ denotes the length-N vector of all ones in $\mathbb{R}^N$.

Therefore, the discriminative prior over the latent positions in DGPLVM can be written as

$$p(\mathbf{Z}) = \frac{1}{\mathcal{Z}_b}\exp\Big(-\frac{1}{\sigma^2}J^*\Big), \tag{13}$$

where $\mathcal{Z}_b$ is a normalization constant, and σ^2 represents a global scaling of the prior.

The covariance matrix obtained by DGPLVM is discriminative and more flexible than the one used in conventional GPs for classification (GPC), since they are learned based on a discriminative criterion, and more degrees of freedom are estimated than conventional kernel hyper-parameters.

4.2. Multi-task Learning Constraint

From an asymmetric multi-task learning perspective, the tasks should be allowed to share common hyper-parameters of the covariance function. Moreover, from an information theory perspective, the information cost between target task and multiple source tasks should be minimized. A natural way to quantify the information cost is to use the mutual entropy, because it is the measure of the mutual dependence of two distributions. For multi-task learning, we extend the mutual entropy to multiple distributions as follows

$$\mathcal{M} = H(p_t) - \frac{1}{S}\sum_{i=1}^{S} H(p_t|p_i), \tag{14}$$

where $H(\cdot)$ is the marginal entropy, $H(\cdot|\cdot)$ is the conditional entropy, S is the number of source tasks, $\{p_i\}_{i=1}^S$, and p_t are the probability distributions of source tasks and target task, respectively.

4.3. GaussianFace Model

In this section, we describe our GaussianFace model in detail. Suppose we have S source-domain datasets $\{\mathbf{X}_1, \dots, \mathbf{X}_S\}$ and a target-domain data $\mathbf{X}_T$. For each source-domain data or target-domain data $\mathbf{X}_i$, according to Equation (9), we write its marginal likelihood

$$p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}}\exp\Big(-\frac{1}{2}\text{tr}(\mathbf{K}^{-1}\mathbf{X}_i\mathbf{X}_i^\top)\Big). \tag{15}$$

where $\mathbf{Z}_i$ represents the domain-relevant latent space. For each source-domain data and target-domain data, their covariance functions $\mathbf{K}$ have the same form because they share the same hyper-parameters $\boldsymbol{\theta}$. In this paper, we use a widely used kernel

$$\mathbf{K}_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) =\theta_0 \exp\Big(-\frac{1}{2}\sum_{m=1}^{d}\theta_m(\mathbf{x}_i^m - \mathbf{x}_j^m)^2\Big) + \theta_{d+1} + \frac{\delta_{\mathbf{x}_i,\mathbf{x}_j}}{\theta_{d+2}}, \tag{16}$$

where $\boldsymbol{\theta} = \{\theta_i\}_{i=0}^{d+2}$ and d is the dimension of the data point. Then, from Equations (10), learning the DGPLVM is equivalent to optimizing

$$p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i) = \frac{1}{\mathcal{Z}_a}p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta})p(\mathbf{Z}_i)p(\boldsymbol{\theta}), \tag{17}$$

where $p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta})$ and $p(\mathbf{Z}_i)$ are respectively represented in (15) and (13). According to the multi-task learning constraint in Equation (14), we can attain

$$\mathcal{M} =H(p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T)) - \frac{1}{S}\sum_{i=1}^{S} H(p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T)|p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i)). \tag{18}$$

From Equations (15), (17), and (18), we know that learning the GaussianFace model amounts to minimizing the following marginal likelihood

$$\mathcal{L}_{Model} = -\log p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T) - \beta\mathcal{M}, \tag{19}$$

where the parameter β balances the relative importance between the target-domain data and the multi-task learning constraint.

4.4. Optimization

For the model optimization, we first expand Equation (19) to obtain the following equation (ignoring the constant items)

$$\mathcal{L}_{Model} =-\log P_T + \beta P_T \log P_T + \frac{\beta}{S}\sum_{i=1}^{S}\big(P_{T,i}\log P_T - P_{T,i}\log P_{T,i}\big), \tag{20}$$

where $P_i = p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i)$ and $P_{i,j}$ means that its corresponding covariance function is computed on both $\mathbf{X}_i$ and $\mathbf{X}_j$. We can now optimize Equation (20) with respect to the hyper-parameters $\boldsymbol{\theta}$ and the latent positions $\mathbf{Z}_i$ by the Scaled Conjugate Gradient (SCG) technique. Since we focus on the covariance matrix in this paper, here we only present the derivations of hyper-parameters. It is easy to get

$$\begin{aligned}\frac{\partial \mathcal{L}_{Model}}{\partial \theta_j} =& \Big(\beta(\log P_T + 1) + \frac{\beta}{SP_T}\sum_{i=1}^{S} P_{T,i} - \frac{1}{P_T}\Big)\frac{\partial P_T}{\partial \theta_j} \\ &+ \frac{\beta}{S}\sum_{i=1}^{S}(\log P_T - \log P_{T,i} - 1)\frac{\partial P_{T,i}}{\partial \theta_j}.\end{aligned}$$

The above equation depends on the form $\frac{\partial P_i}{\partial \theta_j}$ as follows (ignoring the constant items)

$$\begin{aligned}\frac{\partial P_i}{\partial \theta_j} =& P_i \frac{\partial \log P_i}{\partial \theta_j} \\ \approx& P_i\Big(\frac{\partial \log p(\mathbf{X}_i|\mathbf{Z}_i,\boldsymbol{\theta})}{\partial \theta_j} + \frac{\partial \log p(\mathbf{Z}_i)}{\partial \theta_j} + \frac{\partial \log p(\boldsymbol{\theta})}{\partial \theta_j}\Big).\end{aligned}$$

The above three terms can be easily obtained (ignoring the constant items) by

$$\begin{aligned}\frac{\partial \log p(\mathbf{X}_i|\mathbf{Z}_i,\boldsymbol{\theta})}{\partial \theta_j} \approx& \frac{D}{2}\mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\Big) \\ &- \frac{1}{2}\mathbf{K}^{-\top}\mathbf{X}_i\mathbf{X}_i^{\top}\mathbf{K}^{-\top}\frac{\partial \mathbf{K}}{\partial \theta_j}, \\ \frac{\partial \log p(\mathbf{Z}_i)}{\partial \theta_j} \approx& -\frac{1}{\sigma^2}\frac{\partial J_i^*}{\partial \theta_j} \\ =& -\frac{1}{\lambda\sigma^2}\Big(\mathbf{a}^{\top}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{a} - \mathbf{a}^{\top}\frac{\partial \mathbf{K}}{\partial \theta_j}\tilde{\mathbf{A}}\mathbf{a} \\ &+ \mathbf{a}^{\top}\mathbf{K}\tilde{\mathbf{A}}\frac{\partial \mathbf{K}}{\partial \theta_j}\tilde{\mathbf{A}}\mathbf{K}\mathbf{a} - \mathbf{a}^{\top}\mathbf{K}\tilde{\mathbf{A}}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{a}\Big), \\ \frac{\partial \log p(\boldsymbol{\theta})}{\partial \theta_j} =& \frac{1}{\theta_j},\end{aligned}$$

where $\tilde{\mathbf{A}} = \mathbf{A}(\lambda\mathbf{I}_n + \mathbf{AKA})^{-1}\mathbf{A}$. Thus, the desired derivatives have been obtained.

4.5. Speedup

In the GaussianFace model, we need to invert the large matrix when doing inference and prediction. For large problems, both storing the matrix and solving the associated linear systems are computationally prohibitive. In this paper, we use the anchor graphs method [35] to speed up this process. To put it simply, we first select q ($q \ll n$) anchors to cover a cloud of n data points, and form an $n \times q$ matrix $\mathbf{Q}$, where $\mathbf{Q}_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{x}_i$ and $\mathbf{x}_j$ are from n training data points and q anchors, respectively. Then the original kernel matrix $\mathbf{K}$ can be approximated as $\mathbf{K} = \mathbf{QQ}^{\top}$. Using the Woodbury identity [21], computing the $n \times n$ matrix $\mathbf{QQ}^{\top}$ can be transformed into computing the $q \times q$ matrix $\mathbf{Q}^{\top}\mathbf{Q}$, which is more efficient.

Speedup on Inference When optimizing Equation (19), we need to invert the matrix $(\lambda\mathbf{I}_n + \mathbf{AKA})$. During inference, we take q k-means clustering centers as anchors to form $\mathbf{Q}$. Substituting $\mathbf{K} = \mathbf{QQ}^{\top}$ into $(\lambda\mathbf{I}_n + \mathbf{AKA})$, and then using the Woodbury identity, we get

$$\begin{aligned}(\lambda\mathbf{I}_n + \mathbf{AKA})^{-1} &= (\lambda\mathbf{I}_n + \mathbf{AQQ}^{\top}\mathbf{A})^{-1} \\ &= \lambda^{-1}\mathbf{I}_n - \lambda^{-1}\mathbf{AQ}(\lambda\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{AAQ})^{-1}\mathbf{Q}^{\top}\mathbf{A}.\end{aligned}$$

Speedup on Prediction When we compute the predictive variance $\sigma(\mathbf{x}_*)$, we need to invert the matrix $(\mathbf{K} + \mathbf{W}^{-1})$. At this time, we can use the method in Section 3.2 to calculate the accurate clustering centers that can be regarded as the anchors. Using the Woodbury identity again, we obtain

$$(\mathbf{K} + \mathbf{W}^{-1})^{-1} = \mathbf{W} - \mathbf{WQ}(\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{WQ})^{-1}\mathbf{Q}^{\top}\mathbf{W},$$

where $(\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{WQ})$ is only a $q \times q$ matrix, and its inverse matrix can be computed more efficiently.

5. GaussianFace Model for Face Verification

In this section, we describe two applications of the GaussianFace model to face verification: as a binary classifier and as a feature extractor.

Each face image is first normalized to 150×120 size by an affine transformation based on five landmarks (two eyes, nose, and two mouth corners). The image is then divided into overlapped patches of 25×25 pixels with a stride of 2 pixels. Each patch within the image is mapped to a vector by a certain descriptor, and the vector is regarded as the feature of the patch, denoted by $\{\mathbf{x}_p^A\}_{p=1}^{P}$ where P is the number of patches within the face image A. In this paper, the multi-scale LBP feature of each patch is extracted [14]. The difference is that the multi-scale LBP descriptors are extracted at the center of each patch instead of accurate landmarks.

5.1. GaussianFace Model as a Binary Classifier

For classification, our model can be regarded as an approach to learn a covariance function for GPC, as shown in Figure 1 (a). Here, for a pair of face images A and B from the same (or different) person, let the similarity vector $\mathbf{x}_i = [s_1, \ldots, s_p, \ldots, s_P]^{\top}$ be the input data point of the GaussianFace model, where s_p is the similarity of $\mathbf{x}_p^A$ and $\mathbf{x}_p^B$, and its corresponding output is $y_i = 1$ (or -1). With the learned hyper-parameters of covariance function from the training data, given any un-seen pair of face images, we first compute its similarity vector $\mathbf{x}_*$ using the above method, then predict whether the pair is from the same person through Equation (6). In this paper, we prescribe the sigmoid function $\pi(\cdot)$ to be the cumulative Gaussian distribution $\Phi(\cdot)$, which can be solved analytically as $\bar{\pi}_* = \Phi\Big(\frac{\bar{f}_*(\mathbf{x}_*)}{\sqrt{1+\sigma^2(\mathbf{x}_*)}}\Big)$, where $\sigma^2(\mathbf{x}_*) = \mathbf{K}_{**} - \mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^{\top}$ and $\bar{f}_*(\mathbf{x}_*) = \mathbf{K}_*\mathbf{K}^{-1}\hat{\mathbf{f}}$ from Equation (5) [42]. We call the method *GaussianFace-BC*.

5.2. GaussianFace Model as a Feature Extractor

As a feature extractor, our model can be regarded as an approach to automatically extract facial features, shown in Figure 1 (b). Here, for a pair of face images A and B from the same (or different) person, we regard the joint feature vector $\mathbf{x}_i = [(\mathbf{x}_i^A)^\top, (\mathbf{x}_i^B)^\top]^\top$ as the input data point of the GaussianFace model, and its corresponding output is $y_i = 1$ (or -1). To enhance the robustness of our approach, the flipped form of $\mathbf{x}_i$ is also included; for example, $\mathbf{x}_i = [(\mathbf{x}_i^B)^\top, (\mathbf{x}_i^A)^\top]^\top$. After the hyper-parameters of covariance function are learnt from the training data, we can use the method in Section 3.2 to group the input data points into different clusters automatically. Suppose that we finally obtain C clusters. The centers of these clusters are denoted by $\{\mathbf{c}_i\}_{i=1}^C$, the variances of these clusters by $\{\Sigma_i^2\}_{i=1}^C$, and their weights by $\{w_i\}_{i=1}^C$ where w_i is the ratio of the number of data points from the i-th cluster to the number of all data points. Then we refer to each $\mathbf{c}_i$ as the input of Equation (5), and we can obtain its corresponding probability p_i and variance σ_i^2. In fact, $\{\mathbf{c}_i\}_{i=1}^C$ can be regarded as a codebook generated by our model.

For any un-seen pair of face images, we also first compute its joint feature vector $\mathbf{x}_*$ for each pair of patches. Then we compute its first-order and second-order statistics to the centers. The statistics and variance of $\mathbf{x}_*$ are represented as its high-dimensional facial features, denoted by $\hat{\mathbf{x}}_* = [\Delta_1^1, \Delta_1^2, \Delta_1^3, \Delta_1^4, \ldots, \Delta_C^1, \Delta_C^2, \Delta_C^3, \Delta_C^4]^\top$, where $\Delta_i^1 = w_i\left(\frac{\mathbf{x}_* - \mathbf{c}_i}{\Sigma_i}\right)$, $\Delta_i^2 = w_i\left(\frac{\mathbf{x}_* - \mathbf{c}_i}{\Sigma_i}\right)^2$, $\Delta_i^3 = p_i$, and $\Delta_i^4 = \sigma_i^2$. We then concatenate all of the new high-dimensional features from each pair of patches to form the final new high-dimensional feature for the pair of face images. The new high-dimensional facial features not only describe how the distribution of features of an un-seen face image differs from the distribution fitted to the features of all training images, but also encode the predictive information including the probabilities of label and uncertainty. We call this approach *GaussianFace-FE*.

6. Experimental Settings

In this section, we conduct experiments on face verification. We start by introducing the source-domain datasets and the target-domain dataset in all of our experiments (see Figure 2 for examples). The source-domain datasets include four different types of datasets as follows:

Multi-PIE [19]. This dataset contains face images from 337 subjects under 15 view points and 19 illumination conditions in four recording sessions. These images are collected under controlled conditions.

MORPH [43]. The MORPH database contains 55,000 images of more than 13,000 people within the age ranges of 16 to 77. There are an average of 4 images per individual.

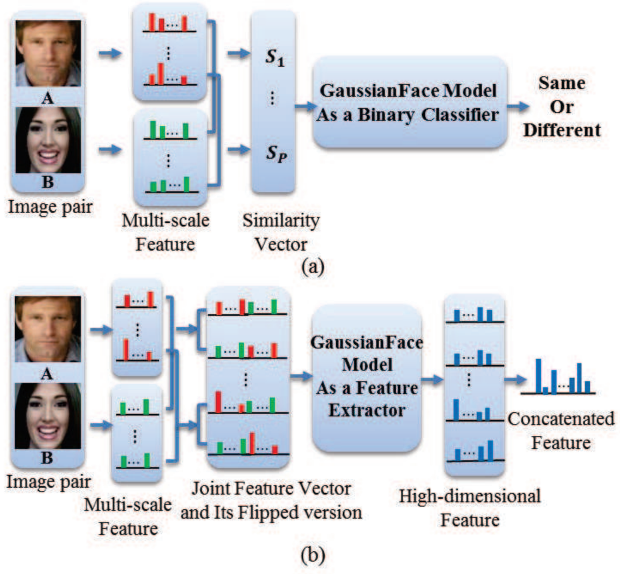


Figure 1. Two approaches based on GaussianFace model for face verification. (a) GaussianFace model as a binary classifier. (b) GaussianFace model as a feature extractor.

Web Images[2]. This dataset contains around 40,000 facial images from 3261 subjects; that is, approximately 10 images for each person. The images were collected from the Web with significant variations in pose, expression, and illumination conditions.

Life Photos[2]. This dataset contains approximately 5000 images of 400 subjects collected online. Each subject has roughly 10 images.

If not otherwise specified, the target-domain dataset is the benchmark of face verification as follows:

LFW [23]. This dataset contains 13,233 uncontrolled face images of 5749 public figures with variety of pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters. All of these images are collected from the Web.

We use the LFW dataset as the target-domain dataset because it is well known as a challenging benchmark. Using it also allows us to compare directly with other existing face verification methods [9, 5, 14, 47, 13, 59, 1, 20, 16]. Besides, this dataset provides a large set of relatively unconstrained face images with complex variations as described above, and has proven difficult for automatic face verification methods [23, 28]. In all the experiments conducted on LFW, we strictly follow the standard unrestricted protocol of LFW [23]. More precisely, during the training procedure, the four source-domain datasets are: Web Images, Multi-PIE, MORPH, and Life Photos, the target-domain dataset is

[2]These two datasets are collected by our own from the Web. It is guaranteed that these two datasets are mutually exclusive with the LFW dataset.

the training set in View 1 of LFW, and the validation set is the test set in View 1 of LFW. At the test time, we follow the standard 10-fold cross-validation protocol to test our model in View 2 of LFW.

For each one of the four source-domain datasets, we randomly sample 20,000 pairs of matched images and 20,000 pairs of mismatched images. The training partition and the testing partition in all of our experiments are mutually exclusive. In other words, there is no identity overlap among the two partitions.

For the experiments below, "The Number of SD" means "the Number of Source-Domain datasets that are fed into the GaussianFace model for training". By parity of reasoning, if "The Number of SD" is i, that means the first i source-domain datasets are used for model training. Therefore, if "The Number of SD" is 0, models are trained with the training data from target-domain data only.

Implementation details. Our model involves four important parameters: λ in (12), σ in (13), β in (19), and the number of anchors q in *Speedup on Inference* [3]. Following the same setting in [27], the regularization parameter λ in (12) is fixed to 10^{-8}. σ reflects the tradeoff between our method's ability to discriminate (small σ) and its ability to generalize (large σ), and β balances the relative importance between the target-domain data and the multi-task learning constraint. Therefore, the validation set (the test set in View 1 of LFW) is used for selecting σ and β. Each time we use different number of source-domain datasets for training, the corresponding optimal σ and β should be selected on the validation set.

Since we collected a large number of image pairs for training (20,000 matched pairs and 20,000 mismatched pairs from each source-domain dataset), and our model is based on the kernel method, thus an important consideration is how to efficiently approximate the kernel matrix using a low-rank method in the limited space and time. We adopt the anchor graphs method (see Section 4.5) for kernel approximation. In our experiments, we take two steps to determine the number of anchor points. In the first step, the optimal σ and β are selected on the validation set in each experiment. In the second step, we fix σ and β, and then tune the number of anchor points. We vary the number of anchor points to train our model on the training set, and test it on the validation set. We report the average accuracy for our model over 10 trials. After we consider the trade-off between memory and running time in practice, the number of anchor points with the best average accuracy is determined in each experiments.

[3]The other parameters, such as the hyper-parameters in the kernel function and the number of anchors in *Speedup on Prediction*, can be automatically learned from the data.

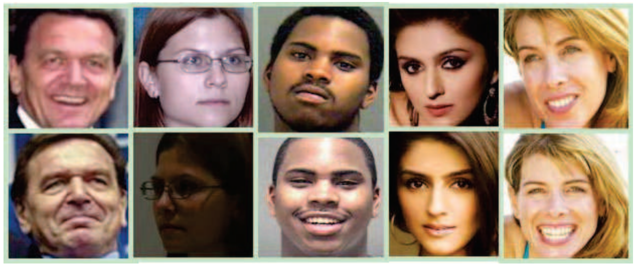


Figure 2. Samples of the datasets in our experiments. From left to right: LFW, Multi-PIE, MORPH, Web Images, and Life Photos.

7. Experimental Results

In this section, we conduct five experiments to demonstrate the validity of the GaussianFace model.

7.1. Comparisons with Other MTGP/GP Methods

Since our model is based on GPs, it is natural to compare our model with four popular GP models: GPC [42], MTGP prediction [6], GPLVM [29], and DGPLVM [57]. For fair comparisons, all these models are trained on multiple source-domain datasets using the same two methods as our GaussianFace model described in Section 5. After the hyper-parameters of covariance function are learnt for each model, we can regard each model as a binary classifier and a feature extractor like ours, respectively. Figure 3 shows that our model significantly outperforms the other four GPs models, and the superiority of our model becomes more obvious as the number of source-domain datasets increases.

7.2. Comparisons with Other Binary Classifiers

Since our model can be regarded as a binary classifier, we have also compared our method with other classical binary classifiers. For this paper, we chose three popular representatives: SVM [12], logistic regression (LR) [17], and Adaboost [18]. Table 1 demonstrates that the performance of our method GaussianFace-BC is much better than those of the other classifiers. Furthermore, these experimental results demonstrates the effectiveness of the multi-task learning constraint. For example, our GaussianFace-BC has about 7.5% improvement when all four source-domain datasets are used for training, while the best one of the other three binary classifiers has only around 4% improvement.

7.3. Comparisons with Other Feature Extractors

Our model can also be regarded as a feature extractor, which is implemented by clustering to generate a codebook. Therefore, we evaluate our method by comparing it with three popular clustering methods: K-means [24], Random Projection (RP) tree [15], and Gaussian Mixture Model (GMM) [47]. Since our method can determine the

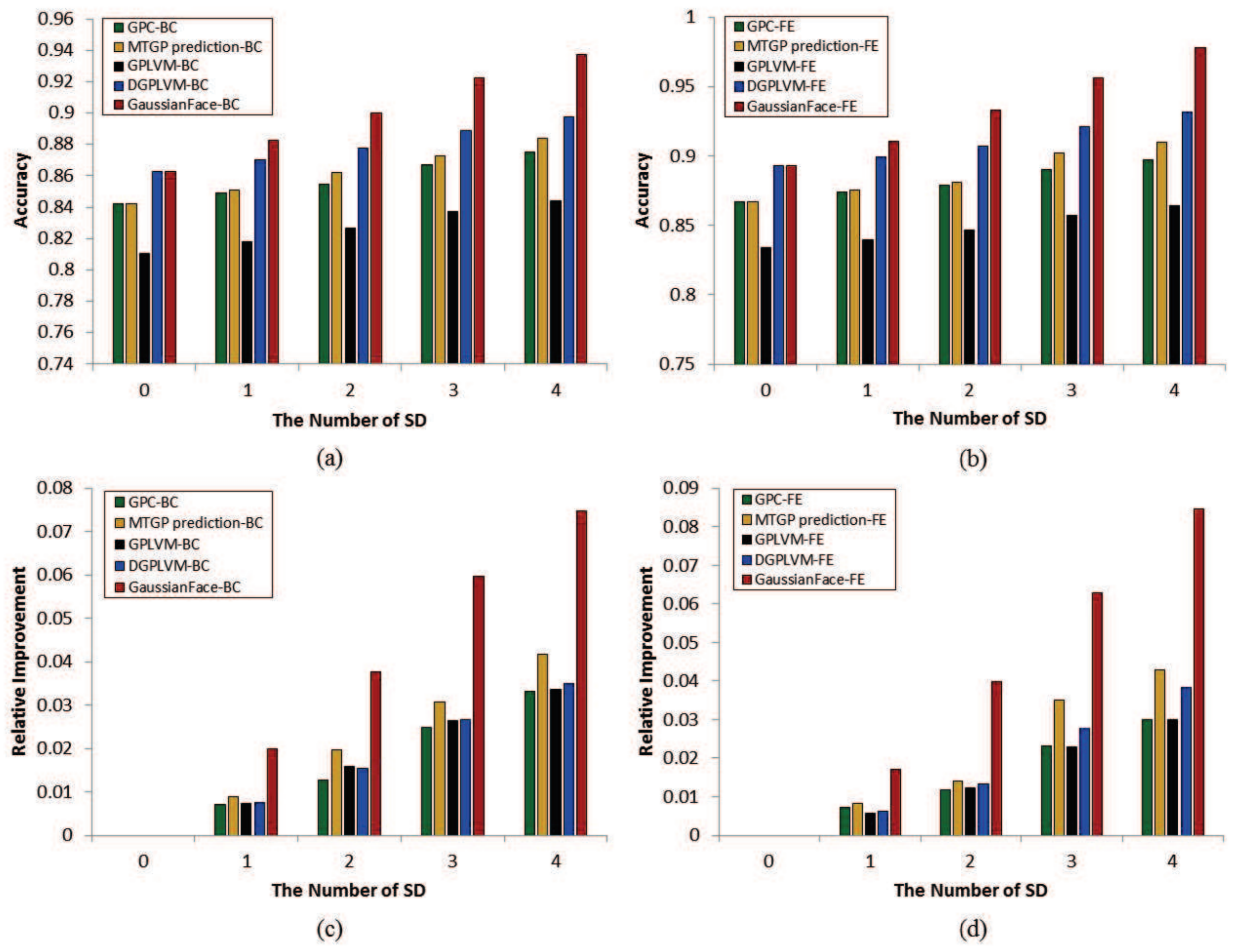


Figure 3. (a) The accuracy rate (%) of the GaussianFace-BC model and other competing MTGP/GP methods as a binary classifier. (b) The accuracy rate (%) of the GaussianFace-FE model and other competing MTGP/GP methods as a feature extractor. (c) The relative improvement of each method as a binary classifier with the increasing number of SD, compared to their performance when the number of SD is 0. (d) The relative improvement of each method as a feature extractor with the increasing number of SD, compared to their performance when the number of SD is 0.

The Number of SD	0	1	2	3	4
SVM [12]	83.21	84.32	85.06	86.43	87.31
LR [17]	81.14	81.92	82.65	83.84	84.75
Adaboost [18]	82.91	83.62	84.80	86.30	87.21
GaussianFace-BC	**86.25**	**88.24**	**90.01**	**92.22**	**93.73**

Table 1. The accuracy rate (%) of our methods as a binary classifier and other competing methods on LFW using the increasing number of source-domain datasets.

The Number of SD	0	1	2	3	4
K-means [24]	84.71	85.20	85.74	86.81	87.68
RP Tree [15]	85.11	85.70	86.45	87.52	88.34
GMM [47]	86.63	87.02	87.58	88.60	89.21
GaussianFace-FE	**89.33**	**91.04**	**93.31**	**95.62**	**97.79**

Table 2. The accuracy rate (%) of our methods as a feature extractor and other competing methods on LFW using the increasing number of source-domain datasets.

number of clusters automatically, for fair comparison, all the other methods generate the same number of clusters as ours. As shown in Table 2, our method GaussianFace-FE significantly outperforms all of the compared approaches, which verifies the effectiveness of our method as a feature extractor. The results have also proved that the multi-task learning constraint is effective. Each time one different type of source-domain dataset is added for training, the performance can be improved significantly. Our GaussianFace-FE model achieves over 8% improvement when the number of SD varies from 0 to 4, which is much higher than the $\sim$3% improvement of the other methods.

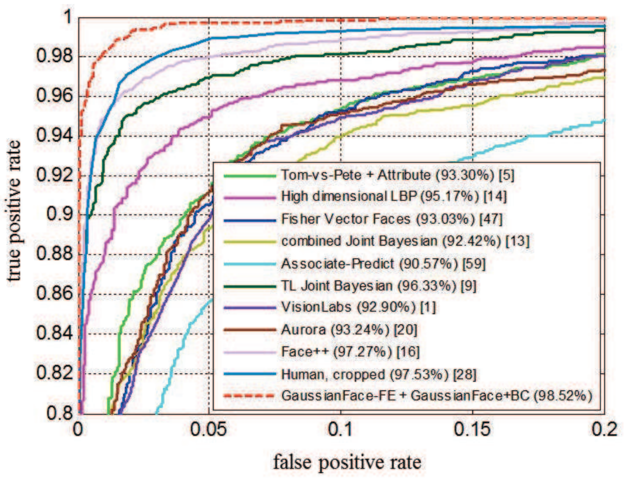


Figure 4. The ROC curve on LFW. Our method achieves the best performance, beating human-level performance.

7.4. Comparison with the state-of-art Methods

Motivated by the appealing performance of both GaussianFace-BC and GaussianFace-FE, we further combine them for face verification. Specifically, after facial features are extracted using GaussianFace-FE, GaussianFace-BC [4] is used to make the final decision. Figure 4 shows the results of this combination compared with state-of-the-art methods [9, 5, 14, 47, 13, 59, 1, 20, 16]. The best published result on the LFW benchmark is 96.33% [5], which is achieved by [9]. Our GaussianFace model can improve the accuracy to 98.52%, which for the first time beats the human-level performance (97.53%, cropped) [28]. Figure 5 presents some example pairs that were always incorrectly classified by our model. Obviously, even for humans, it is also difficult to verify some of them. Here, we emphasize that the centers of patches, instead of the accurate and dense facial landmarks like [9], are utilized to extract multi-scale features in our method. This makes our method simpler and easier to use.

7.5. Further Validations: Shuffling the Source-Target

To further prove the validity of our model, we also consider to treat Multi-PIE and MORPH respectively as the target-domain dataset and the others as the source-domain datasets. The target-domain dataset is split into two mutually exclusive parts: one consisting of 20,000 matched pairs and 20,000 mismatched pairs is used for training, the other is used for test. In the test set, similar to the protocol of LFW, we select 10 mutually exclusive subsets, where each subset consists of 300 matched pairs and 300 mismatched pairs. The experimental results are presented in Figure 6. Each time one dataset is added to the training set, the performance can be improved, even though the types of data are very different in the training set.

[4]Here, the GaussianFace BC is trained with the extracted high-dimensional features using GaussianFace-FE.

[5]In fact, [51] and [53] have achieved higher accuracies 97.15% and 97.25%, respectively. We do not report their performances in Figure 4, since they have not reported their ROC curves on the LFW website so that we cannot obtain the results to draw their ROC curves.

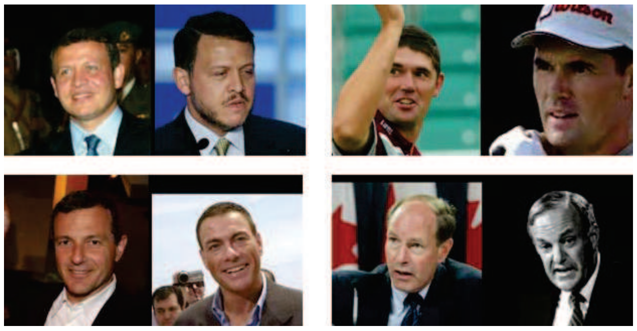
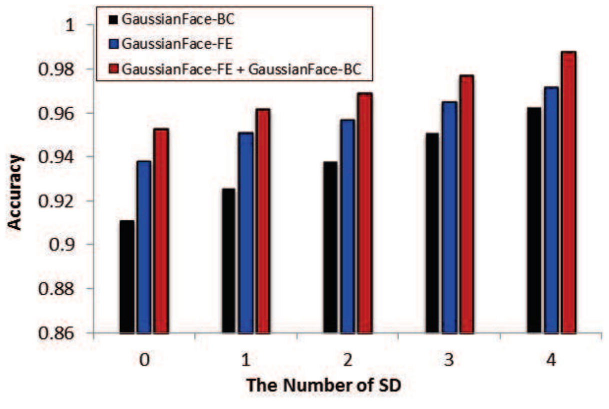


Figure 5. The two rows present examples of matched and mismatched pairs respectively from LFW that were incorrectly classified by the GaussianFace model.

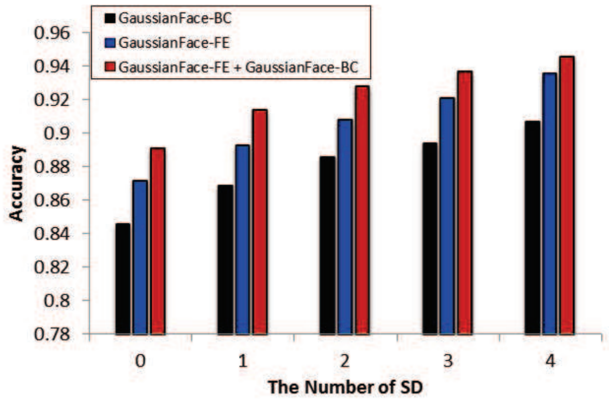
8. General Discussion

There is an implicit belief among many psychologists and computer scientists that human face verification abilities are currently beyond existing computer-based face verification algorithms [39]. This belief, however, is supported more by anecdotal impression than by scientific evidence. By contrast, there have already been a number of papers comparing human and computer-based face verification performance [2, 54, 40, 41, 38, 8]. It has been shown that the best current face verification algorithms perform better than humans in the good and moderate conditions. So, it is really not that difficult to beat human performance in some specific scenarios.

As pointed out by [38, 48], humans and computer-based algorithms have different strategies in face verification. Indeed, by contrast to performance with unfamiliar faces, human face verification abilities for familiar faces are relatively robust to changes in viewing parameters such as illumination and pose. For example, Bruce [7] found human recognition memory for unfamiliar faces dropped substantially when there were changes in viewing parameters. Besides, humans can take advantages of non-face configurable information from the combination of the face and body (e.g., neck, shoulders). It has also been examined in [28], where the human performance drops from 99.20% (tested using the original LFW images) to 97.53% (tested using the cropped LFW images). Hence, the experiments comparing human and computer performance may not show human face verification skill at their best, because humans were asked to match the cropped faces of people previously



(a)



(b)

Figure 6. (a) The accuracy rate (%) of the GaussianFace model on Multi-PIE. (b) The accuracy rate (%) of the GaussianFace model on MORPH.

unfamiliar to them. To the contrary, those experiments can fully show the performance of computer-based face verification algorithms. First, the algorithms can exploit information from enough training images with variations in all viewing parameters to improve face verification performance, which is similar to information humans acquire in developing face verification skills and in becoming familiar with individuals. Second, the algorithms might exploit useful, but subtle, image-based detailed information that give them a slight, but consistent, advantage over humans.

Therefore, surpassing the human-level performance may only be symbolically significant. In reality, a lot of challenges still lay ahead. To compete successfully with humans, more factors such as the robustness to familiar faces and the usage of non-face information, need to be considered in developing future face verification algorithms.

9. Conclusion and Future Work

This paper presents a principled Multi-Task Learning approach based on Discriminative Gaussian Process Latent Variable Model, named *GaussianFace*, for face verification by including a computationally more efficient equivalent form of KFDA and the multi-task learning constraint to the DGPLVM model. We use Gaussian Processes approximation and anchor graphs to speed up the inference and prediction of our model. Based on the GaussianFace model, we propose two different approaches for face verification. Extensive experiments on challenging datasets validate the efficacy of our model. The GaussianFace model finally surpassed human-level face verification accuracy, thanks to exploiting additional data from multiple source-domains to improve the generalization performance of face verification in the target-domain and adapting automatically to complex face variations.

Although several techniques such as the Laplace approximation and anchor graph are introduced to speed up the process of inference and prediction in our GaussianFace model, it still takes a long time to train our model for the high performance. In addition, large memory is also necessary. Therefore, for specific application, one needs to balance the three dimensions: memory, running time, and performance. Generally speaking, higher performance requires more memory and more running time. In the future, the issue of running time can be further addressed by the distributed parallel algorithm or the GPU implementation of large matrix inversion. To address the issue of memory, some online algorithms for training need to be developed. Another more intuitive method is to seek a more efficient sparse representation for the large covariance matrix.

Acknowledgements

We would like to thank Deli Zhao and Chen Change Loy for their insightful discussions. This work is partially supported by ”CUHK Computer Vision Cooperation” grant from Huawei, and by the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project No.CUHK 416510 and 416312) and Guangdong Innovative Research Team Program (No.201001D0104648280).

References

[1] Visionlabs. In *Website: http://www.visionlabs.ru/face-recognition*.

[2] A. Adler and M. E. Schuckers. Comparing human and automatic face recognition performance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(5):1248–1255, 2007.

[3] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *TPAMI*, 28(12):2037–2041, 2006.

[4] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *JMLR*, 2, 2002.

[5] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. In *BMVC*, volume 1, page 5, 2012.

[6] E. Bonilla, K. M. Chai, and C. Williams. Multi-task gaussian process prediction. In *NIPS*, 2008.

[7] V. Bruce. Changing faces: Visual and non-visual coding processes in face recognition. *British Journal of Psychology*, 73(1):105–116, 1982.

[8] V. Bruce, P. J. Hancock, and A. M. Burton. Comparisons between human and computer recognition of faces. In *Automatic Face and Gesture Recognition*, pages 408–413, 1998.

[9] X. Cao, D. Wipf, F. Wen, and G. Duan. A practical transfer learning algorithm for face verification. In *ICCV*. 2013.

[10] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. In *CVPR*, pages 2707–2714, 2010.

[11] K. M. Chai. Multi-task learning with gaussian processes. The University of Edinburgh, 2010.

[12] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM TIST*, 2(3):27, 2011.

[13] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *ECCV*, pages 566–579. 2012.

[14] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *CVPR*. 2013.

[15] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *IEEE Transactions on Information Theory*, 55(7):3229–3242, 2009.

[16] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou. Learning deep face representation. *arXiv preprint arXiv:1403.2802*, 2014.

[17] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[18] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[19] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010.

[20] T. Heseltine, P. Szeptycki, J. Gomes, M. Ruiz, and P. Li. Aurora face recognition technical report: Evaluation of algorithm aurora-c-2014-1 on labeled faces in the wild.

[21] N. J. Higham. *Accuracy and Stability of Numberical Algorithms*. Number 48. Siam, 1996.

[22] G. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *CVPR*, 2012.

[23] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[24] S. U. Hussain, T. Napoléon, F. Jurie, et al. Face recognition using local quantized patterns. In *BMVC*, 2012.

[25] H.-C. Kim, D. Kim, Z. Ghahramani, and S. Y. Bang. Appearance-based gender classification with gaussian processes. *Pattern Recognition Letters*, 27(6):618–626, 2006.

[26] H.-C. Kim and J. Lee. Clustering based on gaussian processes. *Neural computation*, 19(11), 2007.

[27] S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel fisher discriminant analysis. In *ICML*, pages 465–472, 2006.

[28] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, pages 365–372, 2009.

[29] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *NIPS*, volume 2, page 5, 2003.

[30] G. Leen, J. Peltonen, and S. Kaski. Focused multi-task learning using gaussian processes. In *Machine Learning and Knowledge Discovery in Databases*, pages 310–325. 2011.

[31] H. Li, G. Hua, Z. Lin, J. Brandt, and J. Yang. Probabilistic elastic matching for pose variant face verification. In *CVPR*. 2013.

[32] Z. Li, D. Lin, and X. Tang. Nonparametric discriminant analysis for face recognition. *TPAMI*, 31(4):755–761, 2009.

[33] Z. Li, W. Liu, D. Lin, and X. Tang. Nonparametric subspace analysis for face recognition. In *CVPR*, volume 2, pages 961–966, 2005.

[34] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *TIP*, 2002.

[35] W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[36] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[37] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782, 2000.

[38] A. J. O'Toole, X. An, J. Dunlop, V. Natu, and P. J. Phillips. Comparing face recognition algorithms to humans on challenging tasks. *ACM Transactions on Applied Perception*, 9(4):16, 2012.

[39] A. J. OToole, F. Jiang, D. Roark, and H. Abdi. Predicting human performance for face recognition. *Face Processing: Advanced Methods and Models. Elsevier, Amsterdam*, 2006.

[40] A. J. O'Toole, P. J. Phillips, F. Jiang, J. Ayyad, N. Pénard, and H. Abdi. Face recognition algorithms surpass humans matching faces over changes in illumination. *TPAMI*, 29(9):1642–1646, 2007.

[41] P. J. Phillips and A. J. O'Toole. Comparison of human and computer performance across face recognition experiments. *Image and Vision Computing*, 32(1):74–85, 2014.

[42] C. E. Rasmussen and C. K. I. Williams. Gaussian processes for machine learning. 2006.

[43] K. Ricanek and T. Tesafaye. Morph: A longitudinal image database of normal adult age-progression. In *Automatic Face and Gesture Recognition*, pages 341–345, 2006.

[44] O. Rudovic, I. Patras, and M. Pantic. Coupled gaussian process regression for pose-invariant facial expression recognition. In *ECCV*. 2010.

[45] R. Salakhutdinov and G. E. Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. In *NIPS*, 2007.

[46] H. J. Seo and P. Milanfar. Face verification using the lark representation. *TIFS*, 6(4):1275–1286, 2011.

[47] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. *IJCV*, 60(2):91–110, 2004.

[48] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell. Face recognition by humans: 20 results all computer vision researchers should know about. *Department of Brain and Cognitive Sciences, MIT, Cambridge, MA*, 2005.

[49] G. Skolidis and G. Sanguinetti. Bayesian multitask classification with gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12), 2011.

[50] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for face verification. In *ICCV*. 2013.

[51] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *CVPR*, 2014.

[52] Y. Taigman, L. Wolf, and T. Hassner. Multiple one-shots for utilizing class label information. In *BMVC*, pages 1–12, 2009.

[53] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *CVPR, 2014*.

[54] X. Tang and X. Wang. Face sketch recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):50–57, 2004.

[55] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528, 2011.

[56] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *CVPR*, pages 586–591, 1991.

[57] R. Urtasun and T. Darrell. Discriminative gaussian process latent variable model for classification. In *ICML*, pages 927–934, 2007.

[58] J. Wright and G. Hua. Implicit elastic matching with random projections for pose-variant face recognition. In *CVPR*, pages 1502–1509, 2009.

[59] Q. Yin, X. Tang, and J. Sun. An associate-predict model for face recognition. In *CVPR*, pages 497–504, 2011.

[60] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML*, pages 1012–1019, 2005.

[61] Y. Zhang and D.-Y. Yeung. Multi-task warped gaussian process for personalized age estimation. In *CVPR*, pages 2622–2629, 2010.

[62] Z. Zhu, P. Luo, X. Wang, and X. Tang. Deep learning identity preserving face space. In *ICCV*. 2013.

[63] Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonical-view faces in the wild with deep neural networks. *arXiv:1404.3543*, 2014.

Surpassing Human-Level Face Verification Performance on LFW with GaussianFace

Chaochao Lu Xiaoou Tang
Department of Information Engineering, The Chinese University of Hong Kong
{lc013, xtang}@ie.cuhk.edu.hk

Abstract

*Face verification remains a challenging problem in very complex conditions with large variations such as pose, illumination, expression, and occlusions. This problem is exacerbated when we rely unrealistically on a single training data source, which is often insufficient to cover the intrinsically complex face variations. This paper proposes a principled multi-task learning approach based on Discriminative Gaussian Process Latent Variable Model, named GaussianFace, to enrich the diversity of training data. In comparison to existing methods, our model exploits additional data from multiple source-domains to improve the generalization performance of face verification in an unknown target-domain. Importantly, our model can adapt automatically to complex data distributions, and therefore can well capture complex face variations inherent in multiple sources. Extensive experiments demonstrate the effectiveness of the proposed model in learning from diverse data sources and generalize to unseen domain. Specifically, the accuracy of our algorithm achieves an impressive accuracy rate of 98.52% on the well-known and challenging Labeled Faces in the Wild (LFW) benchmark [**?**]. For the first time, the human-level performance in face verification (97.53%) [**?**] on LFW is surpassed.*

1. Introduction

Face verification, which is the task of determining whether a pair of face images are from the same person, has been an active research topic in computer vision for decades [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. It has many important applications, including surveillance, access control, image retrieval, and automatic log-on for personal computer or mobile devices. However, various visual complications deteriorate the performance of face verification, as shown by numerous studies on real-world face images from the wild [**?**]. The Labeled Faces in the Wild (LFW) dataset is well known as a challenging benchmark for face verification. The dataset provides a large set of relatively unconstrained face images with complex variations in pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters. Not surprisingly, LFW has proven difficult for automatic face verification methods [**?**, **?**]. Although there has been significant work [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**] on LFW and the accuracy rate has been improved from 60.02% [**?**] to 97.25% [**?**] since LFW is established in 2007, these studies have not closed the gap to human-level performance [**?**] in face verification.

Why could not we surpass the human-level performance? Two possible reasons are found as follows:

1) Most existing face verification methods assume that the training data and the test data are drawn from the same feature space and follow the same distribution. When the distribution changes, these methods may suffer a large performance drop [**?**]. However, many practical scenarios involve cross-domain data drawn from different facial appearance distributions. Learning a model solely on a single source data often leads to overfitting due to dataset bias [**?**]. Moreover, it is difficult to collect sufficient and necessary training data to rebuild the model in new scenarios, for highly accurate face verification specific to the target domain. In such cases, it becomes critical to exploit more data from multiple source-domains to improve the generalization of face verification methods in the target-domain.

2) Modern face verification methods are mainly divided into two categories: extracting low-level features [**?**, **?**, **?**, **?**, **?**], and building classification models [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. Although these existing methods have made great progress in face verification, most of them are less flexible when dealing with complex data distributions. For the methods in the first category, for example, low-level features such as SIFT [**?**], LBP [**?**], and Gabor [**?**] are handcrafted. Even for features learned from data [**?**, **?**], the algorithm parameters (such as the depth of random projection tree, or the number of centers in k-means) also need to be specified by users. Similarly, for the methods in the second category, the architectures of deep networks in

[**?**, **?**, **?**, **?**] (for example, the number of layers, the number of nodes in each layer, etc.), and the parameters of the models in [**?**, **?**, **?**, **?**] (for example, the number of Gaussians, the number of classifiers, etc.) must also be determined in advance. Since most existing methods require some assumptions to be made about the structures of the data, they cannot work well when the assumptions are not valid. Moreover, due to the existence of the assumptions, it is hard to capture the intrinsic structures of data using these methods.

To this end, we propose the Multi-Task Learning approach based on Discriminative Gaussian Process Latent Variable Model (DGPLVM) [**?**], named *GaussianFace*, for face verification. Unlike most existing studies [**?**, **?**, **?**, **?**, **?**] that rely on a single training data source, in order to take advantage of more data from multiple source-domains to improve the performance in the target-domain, we introduce the multi-task learning constraint to DGPLVM. Here, we investigate the asymmetric multi-task learning because we only focus on the performance improvement of the target task. From the perspective of information theory, this constraint aims to maximize the mutual information between the distributions of target-domain data and multiple source-domains data. Moreover, the GaussianFace model is a reformulation based on the Gaussian Processes (GPs) [**?**], which is a non-parametric Bayesian kernel method. Therefore, our model also can adapt its complexity flexibly to the complex data distributions in the real-world, without any heuristics or manual tuning of parameters.

Reformulating GPs for large-scale multi-task learning is non-trivial. To simplify calculations, we introduce a more efficient equivalent form of Kernel Fisher Discriminant Analysis (KFDA) to DGPLVM. Despite that the GaussianFace model can be optimized effectively using the Scaled Conjugate Gradient (SCG) technique, the inference is slow for large-scale data. We make use of GP approximations [**?**] and anchor graphs [**?**] to speed up the process of inference and prediction, so as to scale our model to large-scale data. Our model can be applied to face verification in two different ways: as a binary classifier and as a feature extractor. In the former mode, given a pair of face images, we can directly compute the posterior likelihood for each class to make a prediction. In the latter mode, our model can automatically extract high-dimensional features for each pair of face images, and then feed them to a classifier to make the final decision.

The main contributions of this paper are as follows:

- We propose a novel GaussianFace model for face verification by virtue of the multi-task learning constraint to DGPLVM. Our model can adapt to complex distributions, avoid over-fitting, exploit discriminative information, and take advantage of multiple source-domains data.
- We introduce a computationally more efficient equivalent form of KFDA to DGPLVM. This equivalent form reformulates KFDA to the kernel version consistent with the covariance function in GPs, which greatly simplifies calculations.
- We introduce approximation in GPs and anchor graphs to speed up the process of inference and prediction.
- We achieve superior performance on the challenging LFW benchmark [**?**], with an accuracy rate of 98.52%, beyond human-level performance reported in [**?**].

2. Related Work

Human and computer performance on face recognition has been compared extensively [**?**, **?**, **?**, **?**, **?**, **?**]. These studies have shown that computer-based algorithms were more accurate than humans in well-controlled environments (e.g., frontal view, natural expression, and controlled illumination), whilst still comparable to humans in the poor condition (e.g., frontal view, natural expression, and uncontrolled illumination). However, the above conclusion is only verified on face datasets with controlled variations, where only one factor changes at a time [**?**, **?**]. To date, there has been virtually no work showing that computer-based algorithms could surpass human performance on unconstrained face datasets, such as LFW, which exhibits natural (multifactor) variations in pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters.

There has been much work dealing with multifactor variations in face verification. For example, Simonyan *et al.* applied the Fisher vector to face verification and achieved a good performance [**?**]. However, the Fisher vector is derived from the Gaussian mixture model (GMM), where the number of Gaussians need to be specified by users, which means it cannot cover complex data automatically. Li *et al.* proposed a non-parametric subspace analysis [**?**, **?**], but it is only a linear transformation and cannot cover the complex distributions. Besides, there also exist some approaches for utilizing plentiful source-domain data. Based on the Joint Bayesian algorithm [**?**], Cao *et al.* proposed a transfer learning approach [**?**] by merging source-domain data with limited target-domain data. Since this transfer learning approach is based on the joint Bayesian model of original visual features, it is not suitable for handling the complex nonlinear data and the data with complex manifold structures. Moreover, the transfer learning approach in [**?**] only considered two different domains, restricting its wider applications in large-scale data from multiple domains. More recently, Zhu *et al.* [**?**] learned the transformation from face images under various poses and lighting conditions to a canonical view with a deep convolutional network. Sun *et al.* [**?**]

learned face representation with a deep model through face identification, which is a challenging multi-class prediction task. Taigman *et al.* [**?**] first utilized explicit 3D face modeling to apply a piecewise affine transformation, and then derived a face representation from a nine-layer deep neural network. Although these methods have achieved high performances on LFW, many parameters of them must be determined in advance so that they are less flexible when dealing with complex data distributions.

The core of our algorithm is GPs. To the best of our knowledge, GPs methods and Multi-task learning with related GPs methods (MTGP) have not been applied for face verification. Actually, MTGP/GPs have been extensively studied in machine learning and computer vision in recent years [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. However, most of them [**?**, **?**, **?**, **?**, **?**, **?**, **?**] have only considered the symmetric multi-task learning, which means that all tasks have been assumed to be of equal importance, whereas our purpose is to enhance performance on a target task given all other source tasks. Leen *et al.* proposed a MTGP model in the asymmetric setting [**?**] to focus on improving performance on the target task, and Kim *et al.* developed a GP model for clustering [**?**], but their methods do not take the discriminative information of the covariance function into special account like DGPLVM. Although the discriminative information is considered in [**?**], it does not apply multi-task learning to improve its performance. Salakhutdinov *et al.* used a deep belief net to learn a good covariance kernel for GPs [**?**]. The limitation of such deep methods is that it is hard to determine which architecture for this network is optimal. Also, multi-task learning constraint was not considered in [**?**].

3. Preliminary

In this section, we briefly review Gaussian Processes (GPs) for classification and clustering [**?**], and Gaussian Process Latent Variable Model (GPLVM) [**?**]. We use GPs method mainly due to the following three notable advantages. Firstly, as mentioned previously, it is a non-parametric method, which means it adapts its complexity flexibly to the complex data distributions in the real-world, without any heuristics or manual tuning of parameters. Secondly, GPs method can be computed effectively because of its closed-form marginal probability computation. Furthermore, its hyper-parameters can be learned from data automatically without using model selection methods such as cross validation, thereby avoiding the high computational cost. Thirdly, the inference of GPs is based on Bayesian rules, resulting in robustness to overfitting. We recommend Rasmussen and Williams's excellent monograph for further reading [**?**].

3.1. Gaussian Processes for Binary Classification

Formally, for two-class classification, suppose that we have a training set $\mathcal{D}$ of N observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where the i-th input point $\mathbf{x}_i \in \mathbb{R}^D$ and its corresponding output y_i is binary, with $y = 1_i$ for one class and $y_i = -1$ for the other. Let $\mathbf{X}$ be the $N \times D$ matrix, where the row vectors represent all n input points, and $\mathbf{y}$ be the column vector of all n outputs. We define a latent variable f_i for each input point $\mathbf{x}_i$, and let $\mathbf{f} = [f_1, \dots, f_N]^\top$. A sigmoid function $\pi(\cdot)$ is imposed to squash the output of the latent function into $[0, 1]$, $\pi(f_i) = p(y_i = 1|f_i)$. Assuming the data set is i.i.d, then the joint likelihood factorizes to

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|f_i) = \prod_{i=1}^{N} \pi(y_i f_i). \tag{1}$$

Moreover, the posterior distribution over latent functions is

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{X}, \mathbf{y}|\boldsymbol{\theta})}. \tag{2}$$

Since neither $p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ nor $p(\mathbf{y}|\mathbf{f})$ can be computed analytically, the Laplace method is utilized to approximate the posterior

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \mathcal{N}(\hat{\mathbf{f}}, (\mathbf{K}^{-1} + \mathbf{W})^{-1}), \tag{3}$$

where $\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ and $\mathbf{W} = -\nabla\nabla \log p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})|_{\mathbf{f}=\hat{\mathbf{f}}}$. Then, we can obtain

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\hat{\mathbf{f}}^\top \mathbf{K}^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log|\mathbf{B}|. \tag{4}$$

where $|\mathbf{B}| = |\mathbf{K}| \cdot |\mathbf{K}^{-1} + \mathbf{W}| = |\mathbf{I}_n + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}|$. The optimal value of $\boldsymbol{\theta}$ can be acquired by using the gradient method to maximize Equation (4). Given any unseen test point x_*, the probability of its latent function f_* is

$$f_*|\mathbf{X}, \mathbf{y}, x_* \sim \mathcal{N}(\mathbf{K}_*\mathbf{K}^{-1}\hat{\mathbf{f}}, \mathbf{K}_{**} - \mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^\top), \tag{5}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \mathbf{W}^{-1}$. Finally, we squash f_* to find the probability of class membership as follows

$$\bar{\pi}(f_*) = \int \pi(f_*)p(f_*|\mathbf{X}, \mathbf{y}, x_*)\mathrm{d}f_*. \tag{6}$$

3.2. Gaussian Processes for Clustering

The principle of GP clustering is based on the key observation that the variances of predictive values are smaller in dense areas and larger in sparse areas. The variances can be employed as a good estimate of the support of a probability density function, where each separate support domain can be considered as a cluster. This observation can be explained from the variance function of any predictive data point x_*

$$\sigma^2(x_*) = \mathbf{K}_{**} - \mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^\top. \tag{7}$$

If x_* is in a sparse region, then $\mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^\top$ becomes small, which leads to large variance $\sigma^2(x_*)$, and vice versa. Another good property of Equation (7) is that it does not depend on the labels, which means it can be applied to the unlabeled data.

To perform clustering, the following dynamic system associated with Equation (7) can be written as

$$F(x) = -\nabla\sigma^2(x). \tag{8}$$

The theorem in [**?**] guarantees that almost all the trajectories approach one of the stable equilibrium points detected from Equation (8). After each data point finds its corresponding stable equilibrium point, we can employ a complete graph [**?**, **?**] to assign cluster labels to data points with the stable equilibrium points. Obviously, the variance function in Equation (7) completely determines the performance of clustering.

3.3. Gaussian Process Latent Variable Model

Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$ denote the matrix whose rows represent corresponding positions of $\mathbf{X}$ in latent space, where $\mathbf{z}_i \in \mathbb{R}^d$ ($d \ll D$). The Gaussian Process Latent Variable Model (GPLVM) can be interpreted as a Gaussian process mapping from a low dimensional latent space to a high dimensional data set, where the locale of the points in latent space is determined by maximizing the Gaussian process likelihood with respect to $\mathbf{Z}$. Given a covariance function for the Gaussian process, denoted by $k(\cdot,\cdot)$, the likelihood of the data given the latent positions is as follows,

$$p(\mathbf{X}|\mathbf{Z},\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}} \exp\Big(-\frac{1}{2}\mathrm{tr}(\mathbf{K}^{-1}\mathbf{X}\mathbf{X}^\top)\Big), \tag{9}$$

where $\mathbf{K}_{i,j} = k(\mathbf{z}_i, \mathbf{z}_j)$. Therefore, the posterior can be written as

$$p(\mathbf{Z},\boldsymbol{\theta}|\mathbf{X}) = \frac{1}{\mathcal{Z}_a} p(\mathbf{X}|\mathbf{Z},\boldsymbol{\theta})p(\mathbf{Z})p(\boldsymbol{\theta}), \tag{10}$$

where $\mathcal{Z}_a$ is a normalization constant, the uninformative priors over $\boldsymbol{\theta}$, and the simple spherical Gaussian priors over $\mathbf{Z}$ are introduced [**?**]. To obtain the optimal $\boldsymbol{\theta}$ and $\mathbf{Z}$, we need to optimize the above likelihood (10) with respect to $\boldsymbol{\theta}$ and $\mathbf{Z}$, respectively.

4. GaussianFace

In order to automatically learn discriminative features or covariance function, and to take advantage of source-domain data to improve the performance in face verification, we develop a principled GaussianFace model by including the multi-task learning constraint into Discriminative Gaussian Process Latent Variable Model (DGPLVM) [**?**].

4.1. DGPLVM Reformulation

The DGPLVM is an extension of GPLVM, where the discriminative prior is placed over the latent positions, rather than a simple spherical Gaussian prior. The DGPLVM uses the discriminative prior to encourage latent positions of the same class to be close and those of different classes to be far. Since face verification is a binary classification problem and the GPs mainly depend on the kernel function, it is natural to use Kernel Fisher Discriminant Analysis (KFDA) [**?**] to model class structures in kernel spaces. For simplicity of inference in the followings, we introduce another equivalent formulation of KFDA to replace the one in [**?**].

KFDA is a kernelized version of linear discriminant analysis method. It finds the direction defined by a kernel in a feature space, onto which the projections of positive and negative classes are well separated by maximizing the ratio of the between-class variance to the within-class variance. Formally, let $\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_+}\}$ denote the positive class and $\{\mathbf{z}_{N_++1}, \ldots, \mathbf{z}_N\}$ the negative class, where the numbers of positive and negative classes are N_+ and $N_- = N - N_+$, respectively. Let $\mathbf{K}$ be the kernel matrix. Therefore, in the feature space, the two sets $\{\phi_{\mathbf{K}}(\mathbf{z}_1), \ldots, \phi_{\mathbf{K}}(\mathbf{z}_{N_+})\}$ and $\{\phi_{\mathbf{K}}(\mathbf{z}_{N_++1}), \ldots, \phi_{\mathbf{K}}(\mathbf{z}_N)\}$ represent the positive class and the negative class, respectively. The optimization criterion of KFDA is to maximize the ratio of the between-class variance to the within-class variance

$$J(\omega, \mathbf{K}) = \frac{(\mathbf{w}^\top(\mu_{\mathbf{K}}^+ - \mu_{\mathbf{K}}^-))^2}{\mathbf{w}^\top(\boldsymbol{\Sigma}_{\mathbf{K}}^+ + \boldsymbol{\Sigma}_{\mathbf{K}}^- + \lambda\mathbf{I}_N)\mathbf{w}}, \tag{11}$$

where λ is a positive regularization parameter, $\mu_{\mathbf{K}}^+ = \frac{1}{N_+}\sum_{i=1}^{N_+}\phi_{\mathbf{K}}(\mathbf{z}_i)$, $\mu_{\mathbf{K}}^- = \frac{1}{N_-}\sum_{i=N_++1}^{N}\phi_{\mathbf{K}}(\mathbf{z}_i)$, $\boldsymbol{\Sigma}_{\mathbf{K}}^+ = \frac{1}{N_+}\sum_{i=1}^{N_+}(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^+)(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^+)^\top$, and $\boldsymbol{\Sigma}_{\mathbf{K}}^- = \frac{1}{N_-}\sum_{i=N_++1}^{N}(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^-)(\phi_{\mathbf{K}}(\mathbf{z}_i) - \mu_{\mathbf{K}}^-)^\top$.

In this paper, however, we focus on the covariance function rather than the latent positions. To simplify calculations, we represent Equation (11) with the kernel function, and let the kernel function have the same form as the covariance function. Therefore, it is natural to introduce a more efficient equivalent form of KFDA with certain assumptions as Kim *et al.* points out [**?**], i.e., maximizing Equation (11) is equivalent to maximizing the following equation

$$J^* = \frac{1}{\lambda}\big(\mathbf{a}^\top\mathbf{K}\mathbf{a} - \mathbf{a}^\top\mathbf{K}\mathbf{A}(\lambda\mathbf{I}_n + \mathbf{A}\mathbf{K}\mathbf{A})^{-1}\mathbf{A}\mathbf{K}\mathbf{a}\big), \tag{12}$$

where

$$\begin{aligned}\mathbf{a} =&[\frac{1}{n_+}\mathbf{1}_{N_+}^\top, -\frac{1}{N_-}\mathbf{1}_{N_-}^\top]\\ \mathbf{A} =&\text{diag}\Big(\frac{1}{\sqrt{N_+}}\big(\mathbf{I}_{N_+} - \frac{1}{N_+}\mathbf{1}_{N_+}\mathbf{1}_{N_+}^\top\big),\\ &\frac{1}{\sqrt{N_-}}\big(\mathbf{I}_{N_-} - \frac{1}{N_-}\mathbf{1}_{N_-}\mathbf{1}_{N_-}^\top\big)\Big).\end{aligned}$$

Here, $\mathbf{I}_N$ denotes the $N \times N$ identity matrix and $\mathbf{1}_N$ denotes the length-N vector of all ones in $\mathbb{R}^N$.

Therefore, the discriminative prior over the latent positions in DGPLVM can be written as

$$p(\mathbf{Z}) = \frac{1}{\mathcal{Z}_b}\exp\Big(-\frac{1}{\sigma^2}J^*\Big), \tag{13}$$

where $\mathcal{Z}_b$ is a normalization constant, and σ^2 represents a global scaling of the prior.

The covariance matrix obtained by DGPLVM is discriminative and more flexible than the one used in conventional GPs for classification (GPC), since they are learned based on a discriminative criterion, and more degrees of freedom are estimated than conventional kernel hyper-parameters.

4.2. Multi-task Learning Constraint

From an asymmetric multi-task learning perspective, the tasks should be allowed to share common hyper-parameters of the covariance function. Moreover, from an information theory perspective, the information cost between target task and multiple source tasks should be minimized. A natural way to quantify the information cost is to use the mutual entropy, because it is the measure of the mutual dependence of two distributions. For multi-task learning, we extend the mutual entropy to multiple distributions as follows

$$\mathcal{M} = H(p_t) - \frac{1}{S}\sum_{i=1}^{S} H(p_t|p_i), \tag{14}$$

where $H(\cdot)$ is the marginal entropy, $H(\cdot|\cdot)$ is the conditional entropy, S is the number of source tasks, $\{p_i\}_{i=1}^S$, and p_t are the probability distributions of source tasks and target task, respectively.

4.3. GaussianFace Model

In this section, we describe our GaussianFace model in detail. Suppose we have S source-domain datasets $\{\mathbf{X}_1, \ldots, \mathbf{X}_S\}$ and a target-domain data $\mathbf{X}_T$. For each source-domain data or target-domain data $\mathbf{X}_i$, according to Equation (9), we write its marginal likelihood

$$p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}}\exp\Big(-\frac{1}{2}\text{tr}(\mathbf{K}^{-1}\mathbf{X}_i\mathbf{X}_i^\top)\Big). \tag{15}$$

where $\mathbf{Z}_i$ represents the domain-relevant latent space. For each source-domain data and target-domain data, their covariance functions $\mathbf{K}$ have the same form because they share the same hyper-parameters $\boldsymbol{\theta}$. In this paper, we use a widely used kernel

$$\begin{aligned}\mathbf{K}_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) =&\theta_0\exp\Big(-\frac{1}{2}\sum_{m=1}^{d}\theta_m(\mathbf{x}_i^m - \mathbf{x}_j^m)^2\Big)\\ &+\theta_{d+1} + \frac{\delta_{\mathbf{x}_i,\mathbf{x}_j}}{\theta_{d+2}},\end{aligned} \tag{16}$$

where $\boldsymbol{\theta} = \{\theta_i\}_{i=0}^{d+2}$ and d is the dimension of the data point. Then, from Equations (10), learning the DGPLVM is equivalent to optimizing

$$p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i) = \frac{1}{\mathcal{Z}_a}p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta})p(\mathbf{Z}_i)p(\boldsymbol{\theta}), \tag{17}$$

where $p(\mathbf{X}_i|\mathbf{Z}_i, \boldsymbol{\theta})$ and $p(\mathbf{Z}_i)$ are respectively represented in (15) and (13). According to the multi-task learning constraint in Equation (14), we can attain

$$\begin{aligned}\mathcal{M} =&H(p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T))\\ &-\frac{1}{S}\sum_{i=1}^{S}H(p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T)|p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i)).\end{aligned} \tag{18}$$

From Equations (15), (17), and (18), we know that learning the GaussianFace model amounts to minimizing the following marginal likelihood

$$\mathcal{L}_{Model} = -\log p(\mathbf{Z}_T, \boldsymbol{\theta}|\mathbf{X}_T) - \beta\mathcal{M}, \tag{19}$$

where the parameter β balances the relative importance between the target-domain data and the multi-task learning constraint.

4.4. Optimization

For the model optimization, we first expand Equation (19) to obtain the following equation (ignoring the constant items)

$$\begin{aligned}\mathcal{L}_{Model} =& -\log P_T + \beta P_T\log P_T\\ &+\frac{\beta}{S}\sum_{i=1}^{S}\big(P_{T,i}\log P_T - P_{T,i}\log P_{T,i}\big),\end{aligned} \tag{20}$$

where $P_i = p(\mathbf{Z}_i, \boldsymbol{\theta}|\mathbf{X}_i)$ and $P_{i,j}$ means that its corresponding covariance function is computed on both $\mathbf{X}_i$ and $\mathbf{X}_j$. We can now optimize Equation (20) with respect to the hyper-parameters $\boldsymbol{\theta}$ and the latent positions $\mathbf{Z}_i$ by the Scaled Conjugate Gradient (SCG) technique. Since we focus on the covariance matrix in this paper, here we only present the derivations of hyper-parameters. It is easy to get

$$\frac{\partial \mathcal{L}_{Model}}{\partial \theta_j} = \Big(\beta(\log P_T + 1) + \frac{\beta}{SP_T}\sum_{i=1}^{S} P_{T,i} - \frac{1}{P_T}\Big)\frac{\partial P_T}{\partial \theta_j} + \frac{\beta}{S}\sum_{i=1}^{S}(\log P_T - \log P_{T,i} - 1)\frac{\partial P_{T,i}}{\partial \theta_j}.$$

The above equation depends on the form $\frac{\partial P_i}{\partial \theta_j}$ as follows (ignoring the constant items)

$$\begin{aligned}\frac{\partial P_i}{\partial \theta_j} =& P_i \frac{\partial \log P_i}{\partial \theta_j}\\ \approx& P_i\Big(\frac{\partial \log p(\mathbf{X}_i|\mathbf{Z}_i,\boldsymbol{\theta})}{\partial \theta_j} + \frac{\partial \log p(\mathbf{Z}_i)}{\partial \theta_j} + \frac{\partial \log p(\boldsymbol{\theta})}{\partial \theta_j}\Big).\end{aligned}$$

The above three terms can be easily obtained (ignoring the constant items) by

$$\begin{aligned}\frac{\partial \log p(\mathbf{X}_i|\mathbf{Z}_i,\boldsymbol{\theta})}{\partial \theta_j} \approx& \frac{D}{2}\mathrm{tr}\Big(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\Big)\\ &- \frac{1}{2}\mathbf{K}^{-\top}\mathbf{X}_i\mathbf{X}_i^{\top}\mathbf{K}^{-\top}\frac{\partial \mathbf{K}}{\partial \theta_j},\\ \frac{\partial \log p(\mathbf{Z}_i)}{\partial \theta_j} \approx& -\frac{1}{\sigma^2}\frac{\partial J_i^*}{\partial \theta_j}\\ =& -\frac{1}{\lambda\sigma^2}\Big(\mathbf{a}^{\top}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{a} - \mathbf{a}^{\top}\frac{\partial \mathbf{K}}{\partial \theta_j}\tilde{\mathbf{A}}\mathbf{a}\\ &+ \mathbf{a}^{\top}\mathbf{K}\tilde{\mathbf{A}}\frac{\partial \mathbf{K}}{\partial \theta_j}\tilde{\mathbf{A}}\mathbf{K}\mathbf{a} - \mathbf{a}^{\top}\mathbf{K}\tilde{\mathbf{A}}\frac{\partial \mathbf{K}}{\partial \theta_j}\mathbf{a}\Big),\\ \frac{\partial \log p(\boldsymbol{\theta})}{\partial \theta_j} =& \frac{1}{\theta_j},\end{aligned}$$

where $\tilde{\mathbf{A}} = \mathbf{A}(\lambda\mathbf{I}_n + \mathbf{AKA})^{-1}\mathbf{A}$. Thus, the desired derivatives have been obtained.

4.5. Speedup

In the GaussianFace model, we need to invert the large matrix when doing inference and prediction. For large problems, both storing the matrix and solving the associated linear systems are computationally prohibitive. In this paper, we use the anchor graphs method [?] to speed up this process. To put it simply, we first select q ($q \ll n$) anchors to cover a cloud of n data points, and form an $n \times q$ matrix $\mathbf{Q}$, where $\mathbf{Q}_{i,j} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{x}_i$ and $\mathbf{x}_j$ are from n training data points and q anchors, respectively. Then the original kernel matrix $\mathbf{K}$ can be approximated as $\mathbf{K} = \mathbf{QQ}^{\top}$. Using the Woodbury identity [?], computing the $n \times n$ matrix $\mathbf{QQ}^{\top}$ can be transformed into computing the $q \times q$ matrix $\mathbf{Q}^{\top}\mathbf{Q}$, which is more efficient.

Speedup on Inference When optimizing Equation (19), we need to invert the matrix $(\lambda\mathbf{I}_n + \mathbf{AKA})$. During inference, we take q k-means clustering centers as anchors to form $\mathbf{Q}$. Substituting $\mathbf{K} = \mathbf{QQ}^{\top}$ into $(\lambda\mathbf{I}_n + \mathbf{AKA})$, and then using the Woodbury identity, we get

$$\begin{aligned}(\lambda\mathbf{I}_n + \mathbf{AKA})^{-1} &= (\lambda\mathbf{I}_n + \mathbf{AQQ}^{\top}\mathbf{A})^{-1}\\ &= \lambda^{-1}\mathbf{I}_n - \lambda^{-1}\mathbf{AQ}(\lambda\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{AAQ})^{-1}\mathbf{Q}^{\top}\mathbf{A}.\end{aligned}$$

Speedup on Prediction When we compute the predictive variance $\sigma(\mathbf{x}_*)$, we need to invert the matrix $(\mathbf{K} + \mathbf{W}^{-1})$. At this time, we can use the method in Section 3.2 to calculate the accurate clustering centers that can be regarded as the anchors. Using the Woodbury identity again, we obtain

$$(\mathbf{K} + \mathbf{W}^{-1})^{-1} = \mathbf{W} - \mathbf{WQ}(\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{WQ})^{-1}\mathbf{Q}^{\top}\mathbf{W},$$

where $(\mathbf{I}_q + \mathbf{Q}^{\top}\mathbf{WQ})$ is only a $q \times q$ matrix, and its inverse matrix can be computed more efficiently.

5. GaussianFace Model for Face Verification

In this section, we describe two applications of the GaussianFace model to face verification: as a binary classifier and as a feature extractor.

Each face image is first normalized to 150×120 size by an affine transformation based on five landmarks (two eyes, nose, and two mouth corners). The image is then divided into overlapped patches of 25×25 pixels with a stride of 2 pixels. Each patch within the image is mapped to a vector by a certain descriptor, and the vector is regarded as the feature of the patch, denoted by $\{\mathbf{x}_p^A\}_{p=1}^P$ where P is the number of patches within the face image A. In this paper, the multi-scale LBP feature of each patch is extracted [?]. The difference is that the multi-scale LBP descriptors are extracted at the center of each patch instead of accurate landmarks.

5.1. GaussianFace Model as a Binary Classifier

For classification, our model can be regarded as an approach to learn a covariance function for GPC, as shown in Figure 1 (a). Here, for a pair of face images A and B from the same (or different) person, let the similarity vector $\mathbf{x}_i = [s_1, \ldots, s_p, \ldots, s_P]^{\top}$ be the input data point of the GaussianFace model, where s_p is the similarity of $\mathbf{x}_p^A$ and $\mathbf{x}_p^B$, and its corresponding output is $y_i = 1$ (or -1). With the learned hyper-parameters of covariance function from the training data, given any un-seen pair of face images, we first compute its similarity vector $\mathbf{x}_*$ using the above method, then predict whether the pair is from the same person through Equation (6). In this paper, we prescribe the sigmoid function $\pi(\cdot)$ to be the cumulative Gaussian distribution $\Phi(\cdot)$, which can be solved analytically as $\bar{\pi}_* = \Phi\Big(\frac{\bar{f}_*(\mathbf{x}_*)}{\sqrt{1+\sigma^2(\mathbf{x}_*)}}\Big)$, where $\sigma^2(\mathbf{x}_*) = \mathbf{K}_{**} - \mathbf{K}_*\tilde{\mathbf{K}}^{-1}\mathbf{K}_*^{\top}$ and $\bar{f}_*(\mathbf{x}_*) = \mathbf{K}_*\mathbf{K}^{-1}\hat{\mathbf{f}}$ from Equation (5) [?]. We call the method *GaussianFace-BC*.

5.2. GaussianFace Model as a Feature Extractor

As a feature extractor, our model can be regarded as an approach to automatically extract facial features, shown in Figure 1 (b). Here, for a pair of face images A and B from the same (or different) person, we regard the joint feature vector $\mathbf{x}_i = [(\mathbf{x}_i^A)^\top, (\mathbf{x}_i^B)^\top]^\top$ as the input data point of the GaussianFace model, and its corresponding output is $y_i = 1$ (or -1). To enhance the robustness of our approach, the flipped form of $\mathbf{x}_i$ is also included; for example, $\mathbf{x}_i = [(\mathbf{x}_i^B)^\top, (\mathbf{x}_i^A)^\top]^\top$. After the hyper-parameters of covariance function are learnt from the training data, we can use the method in Section 3.2 to group the input data points into different clusters automatically. Suppose that we finally obtain C clusters. The centers of these clusters are denoted by $\{\mathbf{c}_i\}_{i=1}^C$, the variances of these clusters by $\{\Sigma_i^2\}_{i=1}^C$, and their weights by $\{w_i\}_{i=1}^C$ where w_i is the ratio of the number of data points from the i-th cluster to the number of all data points. Then we refer to each $\mathbf{c}_i$ as the input of Equation (5), and we can obtain its corresponding probability p_i and variance σ_i^2. In fact, $\{\mathbf{c}_i\}_{i=1}^C$ can be regarded as a codebook generated by our model.

For any un-seen pair of face images, we also first compute its joint feature vector $\mathbf{x}_*$ for each pair of patches. Then we compute its first-order and second-order statistics to the centers. The statistics and variance of $\mathbf{x}_*$ are represented as its high-dimensional facial features, denoted by $\hat{\mathbf{x}}_* = [\Delta_1^1, \Delta_1^2, \Delta_1^3, \Delta_1^4, \ldots, \Delta_C^1, \Delta_C^2, \Delta_C^3, \Delta_C^4]^\top$, where $\Delta_i^1 = w_i\left(\frac{\mathbf{x}_*-\mathbf{c}_i}{\Sigma_i}\right)$, $\Delta_i^2 = w_i\left(\frac{\mathbf{x}_*-\mathbf{c}_i}{\Sigma_i}\right)^2$, $\Delta_i^3 = p_i$, and $\Delta_i^4 = \sigma_i^2$. We then concatenate all of the new high-dimensional features from each pair of patches to form the final new high-dimensional feature for the pair of face images. The new high-dimensional facial features not only describe how the distribution of features of an un-seen face image differs from the distribution fitted to the features of all training images, but also encode the predictive information including the probabilities of label and uncertainty. We call this approach *GaussianFace-FE*.

6. Experimental Settings

In this section, we conduct experiments on face verification. We start by introducing the source-domain datasets and the target-domain dataset in all of our experiments (see Figure 2 for examples). The source-domain datasets include four different types of datasets as follows:

Multi-PIE [**?**]. This dataset contains face images from 337 subjects under 15 view points and 19 illumination conditions in four recording sessions. These images are collected under controlled conditions.

MORPH [**?**]. The MORPH database contains 55,000 images of more than 13,000 people within the age ranges of 16 to 77. There are an average of 4 images per individual.

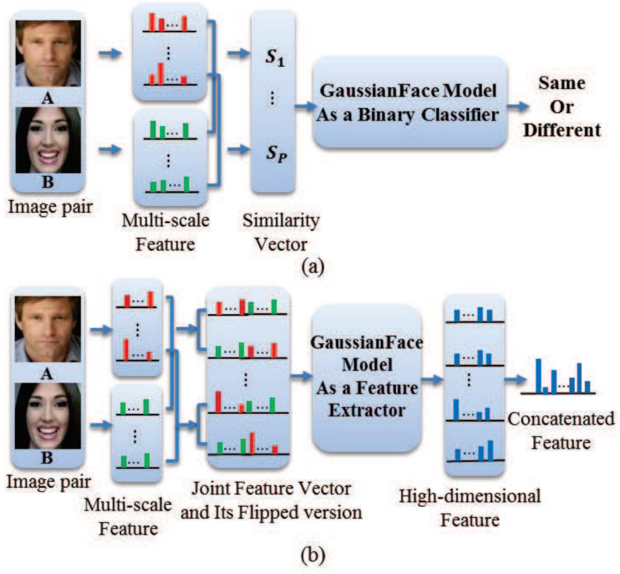


Figure 1. Two approaches based on GaussianFace model for face verification. (a) GaussianFace model as a binary classifier. (b) GaussianFace model as a feature extractor.

Web Images[2]. This dataset contains around 40,000 facial images from 3261 subjects; that is, approximately 10 images for each person. The images were collected from the Web with significant variations in pose, expression, and illumination conditions.

Life Photos[2]. This dataset contains approximately 5000 images of 400 subjects collected online. Each subject has roughly 10 images.

If not otherwise specified, the target-domain dataset is the benchmark of face verification as follows:

LFW [**?**]. This dataset contains 13,233 uncontrolled face images of 5749 public figures with variety of pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters. All of these images are collected from the Web.

We use the LFW dataset as the target-domain dataset because it is well known as a challenging benchmark. Using it also allows us to compare directly with other existing face verification methods [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. Besides, this dataset provides a large set of relatively unconstrained face images with complex variations as described above, and has proven difficult for automatic face verification methods [**?**, **?**]. In all the experiments conducted on LFW, we strictly follow the standard unrestricted protocol of LFW [**?**]. More precisely, during the training procedure, the four source-domain datasets are: Web Images, Multi-PIE, MORPH, and Life Photos, the target-domain dataset is the training set

[2]These two datasets are collected by our own from the Web. It is guaranteed that these two datasets are mutually exclusive with the LFW dataset.

in View 1 of LFW, and the validation set is the test set in View 1 of LFW. At the test time, we follow the standard 10-fold cross-validation protocol to test our model in View 2 of LFW.

For each one of the four source-domain datasets, we randomly sample 20,000 pairs of matched images and 20,000 pairs of mismatched images. The training partition and the testing partition in all of our experiments are mutually exclusive. In other words, there is no identity overlap among the two partitions.

For the experiments below, "The Number of SD" means "the Number of Source-Domain datasets that are fed into the GaussianFace model for training". By parity of reasoning, if "The Number of SD" is i, that means the first i source-domain datasets are used for model training. Therefore, if "The Number of SD" is 0, models are trained with the training data from target-domain data only.

Implementation details. Our model involves four important parameters: λ in (12), σ in (13), β in (19), and the number of anchors q in *Speedup on Inference* [3]. Following the same setting in [**?**], the regularization parameter λ in (12) is fixed to 10^{-8}. σ reflects the tradeoff between our method's ability to discriminate (small σ) and its ability to generalize (large σ), and β balances the relative importance between the target-domain data and the multi-task learning constraint. Therefore, the validation set (the test set in View 1 of LFW) is used for selecting σ and β. Each time we use different number of source-domain datasets for training, the corresponding optimal σ and β should be selected on the validation set.

Since we collected a large number of image pairs for training (20,000 matched pairs and 20,000 mismatched pairs from each source-domain dataset), and our model is based on the kernel method, thus an important consideration is how to efficiently approximate the kernel matrix using a low-rank method in the limited space and time. We adopt the anchor graphs method (see Section 4.5) for kernel approximation. In our experiments, we take two steps to determine the number of anchor points. In the first step, the optimal σ and β are selected on the validation set in each experiment. In the second step, we fix σ and β, and then tune the number of anchor points. We vary the number of anchor points to train our model on the training set, and test it on the validation set. We report the average accuracy for our model over 10 trials. After we consider the trade-off between memory and running time in practice, the number of anchor points with the best average accuracy is determined in each experiments.

[3]The other parameters, such as the hyper-parameters in the kernel function and the number of anchors in *Speedup on Prediction*, can be automatically learned from the data.

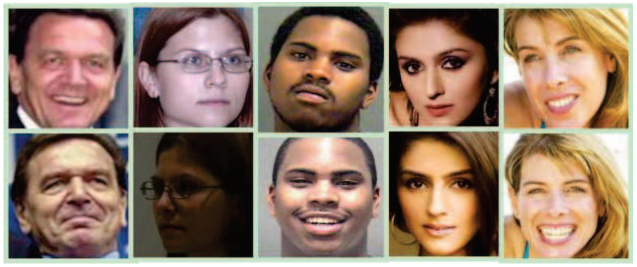


Figure 2. Samples of the datasets in our experiments. From left to right: LFW, Multi-PIE, MORPH, Web Images, and Life Photos.

The Number of SD	0	1	2	3	4
SVM [**?**]	83.21	84.32	85.06	86.43	87.31
LR [**?**]	81.14	81.92	82.65	83.84	84.75
Adaboost [**?**]	82.91	83.62	84.80	86.30	87.21
GaussianFace-BC	**86.25**	**88.24**	**90.01**	**92.22**	**93.73**

Table 1. The accuracy rate (%) of our methods as a binary classifier and other competing methods on LFW using the increasing number of source-domain datasets.

7. Experimental Results

In this section, we conduct five experiments to demonstrate the validity of the GaussianFace model.

7.1. Comparisons with Other MTGP/GP Methods

Since our model is based on GPs, it is natural to compare our model with four popular GP models: GPC [**?**], MTGP prediction [**?**], GPLVM [**?**], and DGPLVM [**?**]. For fair comparisons, all these models are trained on multiple source-domain datasets using the same two methods as our GaussianFace model described in Section 5. After the hyper-parameters of covariance function are learnt for each model, we can regard each model as a binary classifier and a feature extractor like ours, respectively. Figure 3 shows that our model significantly outperforms the other four GPs models, and the superiority of our model becomes more obvious as the number of source-domain datasets increases.

7.2. Comparisons with Other Binary Classifiers

Since our model can be regarded as a binary classifier, we have also compared our method with other classical binary classifiers. For this paper, we chose three popular representatives: SVM [**?**], logistic regression (LR) [**?**], and Adaboost [**?**]. Table 1 demonstrates that the performance of our method GaussianFace-BC is much better than those of the other classifiers. Furthermore, these experimental results demonstrates the effectiveness of the multi-task learning constraint. For example, our GaussianFace-BC has about 7.5% improvement when all four source-domain datasets are used for training, while the best one of the other three binary classifiers has only around 4% improvement.

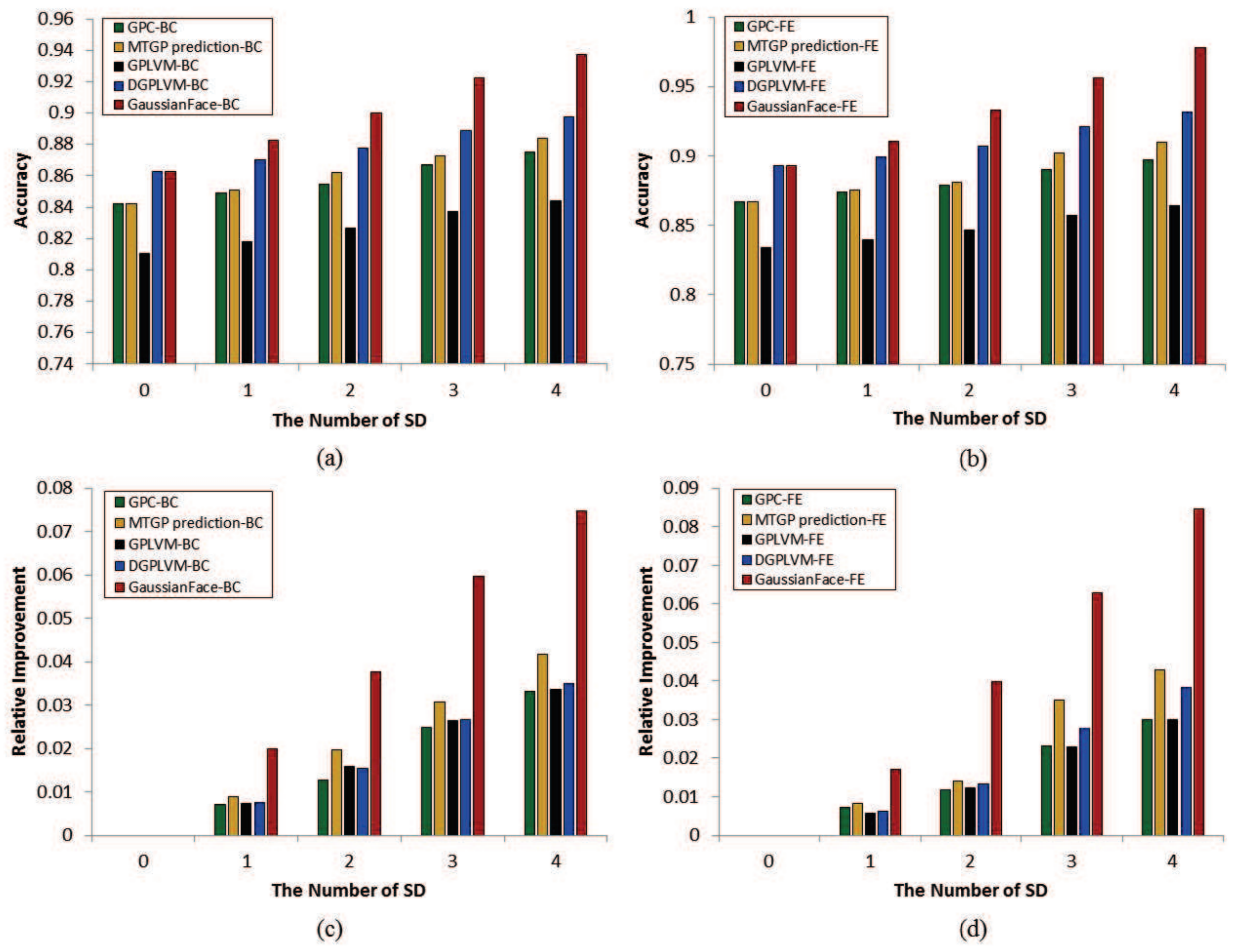


Figure 3. (a) The accuracy rate (%) of the GaussianFace-BC model and other competing MTGP/GP methods as a binary classifier. (b) The accuracy rate (%) of the GaussianFace-FE model and other competing MTGP/GP methods as a feature extractor. (c) The relative improvement of each method as a binary classifier with the increasing number of SD, compared to their performance when the number of SD is 0. (d) The relative improvement of each method as a feature extractor with the increasing number of SD, compared to their performance when the number of SD is 0.

The Number of SD	0	1	2	3	4
K-means [**?**]	84.71	85.20	85.74	86.81	87.68
RP Tree [**?**]	85.11	85.70	86.45	87.52	88.34
GMM [**?**]	86.63	87.02	87.58	88.60	89.21
GaussianFace-FE	**89.33**	**91.04**	**93.31**	**95.62**	**97.79**

Table 2. The accuracy rate (%) of our methods as a feature extractor and other competing methods on LFW using the increasing number of source-domain datasets.

7.3. Comparisons with Other Feature Extractors

Our model can also be regarded as a feature extractor, which is implemented by clustering to generate a codebook. Therefore, we evaluate our method by comparing it with three popular clustering methods: K-means [**?**], Random Projection (RP) tree [**?**], and Gaussian Mixture Model (GMM) [**?**]. Since our method can determine the number of clusters automatically, for fair comparison, all the other methods generate the same number of clusters as ours. As shown in Table 2, our method GaussianFace-FE significantly outperforms all of the compared approaches, which verifies the effectiveness of our method as a feature extractor. The results have also proved that the multi-task learning constraint is effective. Each time one different type of source-domain dataset is added for training, the performance can be improved significantly. Our GaussianFace-FE model achieves over 8% improvement when the number of SD varies from 0 to 4, which is much higher than the $\sim$3% improvement of the other methods.

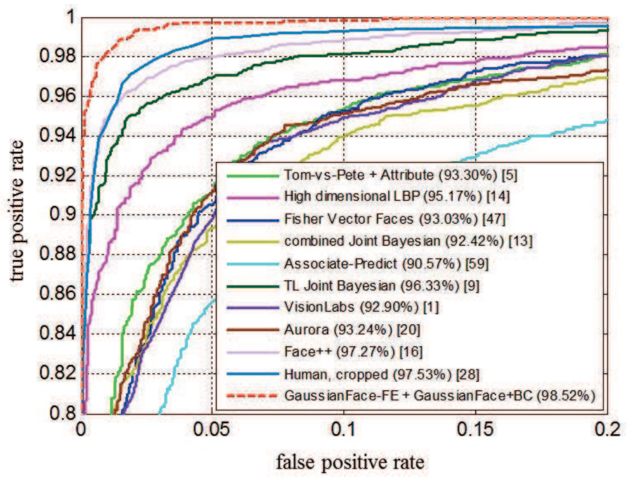


Figure 4. The ROC curve on LFW. Our method achieves the best performance, beating human-level performance.

7.4. Comparison with the state-of-art Methods

Motivated by the appealing performance of both GaussianFace-BC and GaussianFace-FE, we further combine them for face verification. Specifically, after facial features are extracted using GaussianFace-FE, GaussianFace-BC [4] is used to make the final decision. Figure 4 shows the results of this combination compared with state-of-the-art methods [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. The best published result on the LFW benchmark is 96.33% [5], which is achieved by [**?**]. Our GaussianFace model can improve the accuracy to 98.52%, which for the first time beats the human-level performance (97.53%, cropped) [**?**]. Figure 5 presents some example pairs that were always incorrectly classified by our model. Obviously, even for humans, it is also difficult to verify some of them. Here, we emphasize that the centers of patches, instead of the accurate and dense facial landmarks like [**?**], are utilized to extract multi-scale features in our method. This makes our method simpler and easier to use.

7.5. Further Validations: Shuffling the Source-Target

To further prove the validity of our model, we also consider to treat Multi-PIE and MORPH respectively as the target-domain dataset and the others as the source-domain datasets. The target-domain dataset is split into two mutually exclusive parts: one consisting of 20,000 matched pairs and 20,000 mismatched pairs is used for training, the other is used for test. In the test set, similar to the protocol of LFW, we select 10 mutually exclusive subsets, where each subset consists of 300 matched pairs and 300 mismatched pairs. The experimental results are presented in Figure 6. Each time one dataset is added to the training set, the performance can be improved, even though the types of data are very different in the training set.

[4]Here, the GaussianFace BC is trained with the extracted high-dimensional features using GaussianFace-FE.

[5]In fact, [**?**] and [**?**] have achieved higher accuracies 97.15% and 97.25%, respectively. We do not report their performances in Figure 4, since they have not reported their ROC curves on the LFW website so that we cannot obtain the results to draw their ROC curves.

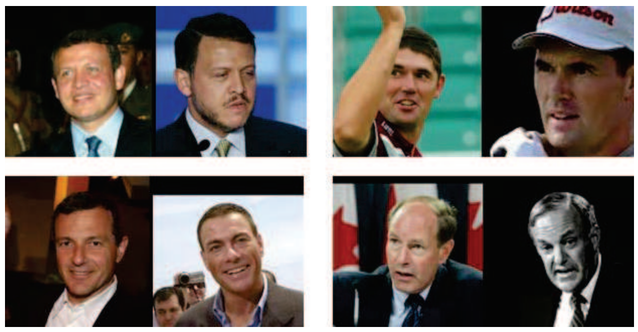


Figure 5. The two rows present examples of matched and mismatched pairs respectively from LFW that were incorrectly classified by the GaussianFace model.

8. General Discussion

There is an implicit belief among many psychologists and computer scientists that human face verification abilities are currently beyond existing computer-based face verification algorithms [**?**]. This belief, however, is supported more by anecdotal impression than by scientific evidence. By contrast, there have already been a number of papers comparing human and computer-based face verification performance [**?**, **?**, **?**, **?**, **?**, **?**]. It has been shown that the best current face verification algorithms perform better than humans in the good and moderate conditions. So, it is really not that difficult to beat human performance in some specific scenarios.

As pointed out by [**?**, **?**], humans and computer-based algorithms have different strategies in face verification. Indeed, by contrast to performance with unfamiliar faces, human face verification abilities for familiar faces are relatively robust to changes in viewing parameters such as illumination and pose. For example, Bruce [**?**] found human recognition memory for unfamiliar faces dropped substantially when there were changes in viewing parameters. Besides, humans can take advantages of non-face configurable information from the combination of the face and body (e.g., neck, shoulders). It has also been examined in [**?**], where the human performance drops from 99.20% (tested using the original LFW images) to 97.53% (tested using the cropped LFW images). Hence, the experiments comparing human and computer performance may not show human face verification skill at their best, because humans were asked to match the cropped faces of people previously unfamiliar to them. To the contrary, those experiments

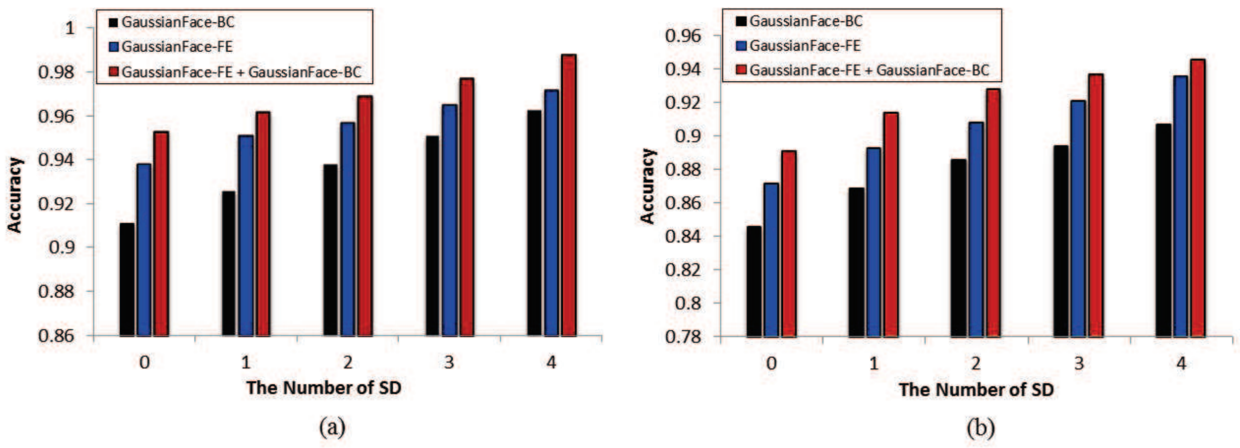


Figure 6. (a) The accuracy rate (%) of the GaussianFace model on Multi-PIE. (b) The accuracy rate (%) of the GaussianFace model on MORPH.

can fully show the performance of computer-based face verification algorithms. First, the algorithms can exploit information from enough training images with variations in all viewing parameters to improve face verification performance, which is similar to information humans acquire in developing face verification skills and in becoming familiar with individuals. Second, the algorithms might exploit useful, but subtle, image-based detailed information that give them a slight, but consistent, advantage over humans.

Therefore, surpassing the human-level performance may only be symbolically significant. In reality, a lot of challenges still lay ahead. To compete successfully with humans, more factors such as the robustness to familiar faces and the usage of non-face information, need to be considered in developing future face verification algorithms.

9. Conclusion and Future Work

This paper presents a principled Multi-Task Learning approach based on Discriminative Gaussian Process Latent Variable Model, named *GaussianFace*, for face verification by including a computationally more efficient equivalent form of KFDA and the multi-task learning constraint to the DGPLVM model. We use Gaussian Processes approximation and anchor graphs to speed up the inference and prediction of our model. Based on the GaussianFace model, we propose two different approaches for face verification. Extensive experiments on challenging datasets validate the efficacy of our model. The GaussianFace model finally surpassed human-level face verification accuracy, thanks to exploiting additional data from multiple source-domains to improve the generalization performance of face verification in the target-domain and adapting automatically to complex face variations.

Although several techniques such as the Laplace approximation and anchor graph are introduced to speed up the process of inference and prediction in our GaussianFace model, it still takes a long time to train our model for the high performance. In addition, large memory is also necessary. Therefore, for specific application, one needs to balance the three dimensions: memory, running time, and performance. Generally speaking, higher performance requires more memory and more running time. In the future, the issue of running time can be further addressed by the distributed parallel algorithm or the GPU implementation of large matrix inversion. To address the issue of memory, some online algorithms for training need to be developed. Another more intuitive method is to seek a more efficient sparse representation for the large covariance matrix.

Acknowledgements

We would like to thank Deli Zhao and Chen Change Loy for their insightful discussions. This work is partially supported by ”CUHK Computer Vision Cooperation” grant from Huawei, and by the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project No.CUHK 416510 and 416312) and Guangdong Innovative Research Team Program (No.201001D0104648280).