Implementing a Face Recognition System for Media Companies

Arturs Sprogis[1(✉)], Karlis Freivalds[1], and Elita Cirule[2]

[1] Institute of Mathematics and Computer Science, University of Latvia, Raina blvd. 29, Riga 1459, Latvia
{arturs.sprogis,karlis.freivalds}@lumii.lv
[2] LETA, Marijas 2, Riga 1050, Latvia
elita.cirule@leta.lv

Abstract. During the past few years face recognition technologies have greatly benefited from the huge progress in machine learning and now have achieved precision rates that are even comparable with humans. This allows us to apply face recognition technologies more effectively for a number of practical problems in various businesses like media monitoring, security, advertising, entertainment that we previously were not able to do due to low precision rates of existing face recognition technologies. In this paper we discuss how to build a face recognition system for media companies and share our experience gained from implementing one for Latvian national news agency LETA. Our contribution is: which technologies to use, how to build a practical training dataset, how large should it be, how to deal with unknown persons.

Keywords: Face recognition · Media companies · System implementation

1 Introduction

A face recognition problem has been researched for decades and for a long time hand-crafted or machine assisted feature detection techniques were the dominant. The latest such face recognition techniques [1–3] use sophisticated algorithms to select up to tens of thousands of parameters to represent a face. But they achieve high accuracy only in constrained environments where faces are frontally positioned, and a variety of conditions like lighting, expression, and occlusion are restricted. Obviously in real life images these conditions are rarely satisfied, and, therefore, in unconstrained environment the accuracy rates are significantly lower.

In contrast to hand-crafted techniques, machine learning approaches let a computer (instead of human) figuring out which face parameters are important to measure. Since these parameters are extracted from the training data, the recognition accuracy depend more on the quality and variety of the training data and less on the environment constraints. In addition, machine learning models typically use 100 to 2000 dimensions to represent a face, in contrast to tens of thousands used by hand-crafted methods. As a result, machine learning techniques are more effective for representing and processing

A. Lupeikiene et al. (Eds.): DB&IS 2018, CCIS 838, pp. 328–341, 2018.
https://doi.org/10.1007/978-3-319-97571-9_26

faces, as well as they are more flexible and accurate to recognize people in real life images with no constraining conditions.

In this paper we are discussing how to apply the latest scientific and technological achievements in face recognition from specialized face recognition companies as well as tech giants like Google, Facebook, Baidu to implement a face recognition system for media companies utilizing state-of-the-art face recognition machine learning models. Typically, media companies own large archives of images and videos in which persons have to be tagged. Accomplishing this task requires executing a number of sub-tasks like extracting training datasets of sample images with a large number of identities (possibly thousands), additionally recognizing unknown people (that is, those who do not belong to our training dataset), selecting an appropriate algorithm for each step in the face recognition pipeline.

The structure of the paper is the following. In Sect. 2 we will explain the pipeline of modern face recognition technology. In Sect. 3 we will discuss how to put together all the building blocks to implement the actual face recognition system.

2 Face Recognition Pipeline

A typical modern face recognition pipeline consists of four steps: detection, alignment, representation and classification [4].

2.1 Detection and Alignment

The first step in the face recognition pipeline is to detect faces. To detect faces we have to locate face areas in images. As a result a list of locations of faces is produced in this step. An example of face detection is demonstrated in Fig. 1.

Fig. 1. An example of face detection

When we have detected faces, they most probably are turned in different directions. Obviously such faces look totally different to a computer. To account for this, faces are aligned so that the eyes and lips are always centered. This significantly simplifies the face recognition step for the computer.

2.2 Representation

In this step, an n-dimensional vector (also called "embedding" vector) is computed to represent each face. These vectors have a characteristic that vectors representing one and the same person are geometrically close in n-dimensional space, whereas vectors representing different persons are geometrically farther from each other. Traditionally, the L2 or cosine distance is used to measure the distance between vectors, and then some experimental threshold is determined to distinguish whether two vectors represent one and the same person or two different persons. An example of face recognition is demonstrated in Fig. 2.

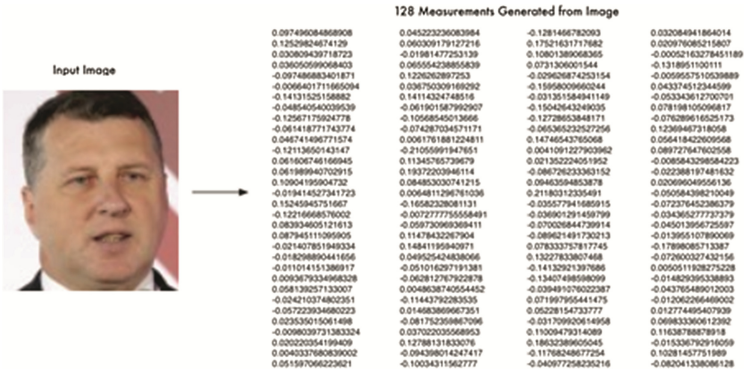

Fig. 2. An example of face representation as a 128 dimensional vector

The face representation step is the most important and complex of all steps in the face recognition pipeline because the overall accuracy rates mostly depend on the way we compute face representations. Currently, the best results are demonstrated by face recognition systems that perform the following two steps. Firstly, they transform faces to n-dimension vector using deep convolutional neural network (CNN) [5]. Secondly, they perform dimensionality reduction, if necessary. Commonly used methods for this task are PCA [6], Joint Bayesian [7] and Metric learning [8].

Next we will review some of the most accurate approaches for face representation computing which are selected by their performance on the popular LFW [9] dataset.

MMDFR. MMDFR [10] is a solution that achieves 99.02% accuracy rate on LFW dataset. In the first step the system aligns faces to 230 × 230 pixels and then transforms them to 3D model. Then the 3D model is cropped and passed to 8 different CNNs. The representation vector is computed by combining 8 vectors by applying Stacked Auto-Encoder method for dimensionality reduction. The system is trained on a data set containing more than 9,000 identities.

Face++. FACE++ [11] is a solution that achieves 99.50% accuracy rate on LFW dataset. The system is trained on a data set consisting of 500,000 images with 10,000

identities. The model consists of 4 different models each recognizing a certain face area. The representation vector is computed by combining 4 vectors by applying PCA method.

DeepID3. DeepID3 [12] is a solution that achieves 99.53% accuracy rate on LFW dataset. The system consists of two CNN models VGG net [13] and GoogLeNet [14] that are extended with Supervisory signals method [15]. For one model the original face is passed, for the second model the horizontally rotated faces. The representation vector is computed by combining 2 vectors in a single vector having 300,000 dimensions. By applying PCA method the dimensionality is reduced to 300. The system is trained on a data set consisting of 300,000 images.

Facenet. Facenet [16] is a solution that achieves 99.63% accuracy rate on LFW dataset. The system is implemented by Google. The model consists of 1 CNN with 140 million parameters and is trained on a data set consisting of hundreds of millions of images. The resulting representation vector has 128 dimensions.

Daream. Daream [17] is a commercial solution that achieves 99.68% accuracy rate on LFW dataset. The publicly available information tells that the system is trained on a data set consisting of 1.2 million images with 30,000 identities. The final model consists of 4 different models - Residual network [18], Wide-residual network [19], Highway path network [20] and Alexnet [21]. The representation vector is computed by combining 4 vectors by applying Joint Bayesian method for dimensionality reduction.

Baidu. Baidu [22] is a commercial solution that achieves 99.77% accuracy rate on LFW dataset. The publicly available information tells that the system is trained on a data set consisting of 1.2 million images with 18,000 identities. The final model consists of 9 different models each recognizing a certain face area. The representation vector is computed by combining 9 vectors by applying learning with triplet loss method to obtain a single 128-dimensional vector.

Dahua-FaceImage. Dahua-FaceImage [23] is a commercial solution that achieves 99.78% accuracy rate on LFW dataset. The publicly available information tells that the system is trained on a data set consisting of 2 million images with 20,000 identities. The final model consists of 30 different CNNs, and the representation vector is computed as a combination of 30 vectors by applying *Joint Bayesian* method for dimensionality reduction.

3 Implementation

So far we have seen an overview of a general face recognition pipeline and have reviewed various techniques to compute face representations. In this section we will discuss how to implement the actual face recognition system by applying these techniques.

3.1 Model

To compute face representations we must have a trained model. One option is to select one of the previously described approaches to train the model by ourselves, or another is to use the already trained model. To train a model from scratch we need a large dataset with labeled images as well as computing resources. On the other hand, there are available already trained models, for instance, [24, 25]. These models are trained on publicly available datasets CASIA-WebFace [26], FaceScrub [27] and MS-Celeb-1M [28]. Thus, if we have enough computing resources and our data set is larger than publicly available data sets, then it is reasonable to train a model by ourselves. Otherwise, we would recommend selecting an existing solution. In general, they have very decent precision rates. For [24] the accuracy rate is 97.3%, for [25] it is 99.2% on LFW datasets in comparison to 99.63% and 99.78% for solutions from Google and Baidu.

3.2 Classification

When we have computed face representations, we have to decide how to classify them, or in other words, how to attach the most appropriate name. In general, for many classification tasks SVM [29] algorithm is a popular choice. However, it does not fit well for our face classification task when there are known and unknown persons. To classify unknown persons we need not only the identity of the classified person but also the confidence score to determine the likelihood level of the classified person to be able to decide between the known and the unknown persons. Although SVM returns both, the identity and confidence score, the problem is that the confidence score is computed as a probability depending on the number of identities in the training dataset, and therefore it is not possible to have one particular threshold value because its values varies as a number of identities change (as a set of identities grows, the confidence score decreases).

We are suggesting to use an alternative approach: K-nearest neighbors (KNN) algorithm. In particular, the algorithm computes the k nearest points and the corresponding spatial distances for the given point. In context of face recognition, the algorithm for the given face selects the k most similar faces from the training dataset and their corresponding distances. Then we have to decide how to classify the known and the unknown persons. The algorithm we used is the following. First, we pick k most similar candidate faces (we used $k = 12$, but regarding the second step whether k is selected sufficiently large, it has minimal effect on the accuracy rate). Second, from the top k candidates we select those with distances smaller than some given distance threshold. This will give us the top similar faces, and then we have to decide whether it is a known person and give its name, or declare it as unknown. To achieve this we use a voting mechanism that attaches the identity having the majority of the votes or hits some threshold. To explain it in a greater detail, we will assume that after the second step we have selected the 10 top similar faces and the voting threshold is 4. Now we may have multiple cases. One case is that the majority of the faces belong to the same identity X (for instance, 6 of 10), and this number is greater than the voting threshold (6 is greater than 4), then we declare that the given face belongs to X. Another case is when there is not a majority for one particular identity, for instance, 4 faces belong to X and Y, and the rest belong

to Z. In that case we measure average distances for X and Y additionally to decide the identity. Finally, we may have a case when none of the identities hits the voting threshold, and in that case we declare the given face as unknown. For instance, if 2 faces belong to the five different identities, then none of them satisfy the voting threshold restriction (2 is not greater than 4).

Nevertheless, in practice it is wise to add some categories during the classification. For example, in LETA project we introduced four categories – very high (distance threshold minus 4x bias), high (distance threshold minus 3x bias), medium (distance threshold minus 2x bias) and low (distance threshold minus bias) where bias is equal 0.06.

3.3 Training Dataset

Until now we have discussed technologies handling the engineering complexity, but one of the key problems to successfully apply face recognition technologies is data. Thus before we can recognize someone in a picture or video, we have to build a training dataset containing some number of images per each person we want to recognize. To achieve applicable results we need at least 6 sample images per identity, however, the optimal number would be 25–45 (see Sect. 3.4). The rule of thumb is that the more samples with different variety we have, the more accurate results we will get. This is especially important to cover real life situations when faces are in different facial expressions, various poses, image resolutions, lighting, etc.

Thus to build a system being able to recognize thousands of identities, we need to collect tens or even hundreds of thousands of training images accordingly. Performing such a task manually is slow, labor-intensive and expensive, and therefore not scalable. While examining alternatives, we noticed that most of the images from LETA archive have an additional description explaining the location, event and the persons in the image (see Fig. 3) which we found very useful to provide an automated support for identity extraction.

The idea for the identity extraction algorithm is straightforward. First, we perform a face detection and check if there is exactly one face in the image. Then we perform a description analysis by applying advanced natural language processing algorithms [30] to extract person entities (it has to be noted that in Latvian language word endings change depending on context that makes an entity extraction more complicated) and then we check if there is exactly one entity. Thus, if there is exactly one face detected and there is exactly one entity extracted, then we assume that the detected face belongs to the extracted entity.

However, we found that there are scenarios when this assumption does not hold. The problem is that the entity extracted cannot always detect foreign names and therefore instead of two names sometimes only one is extracted causing face-entity mismatch (see Fig. 3).

To deal with face-entity mismatches, we extended the identity extraction process with post-processing step. The idea of this step is to iterate over the collected training dataset and find possibly higher number of face-entity mismatches. To accomplish the task we applied our face recognition algorithm for every image in the training dataset. Since we already know to which identity the image belongs, we can easily compare if

Fig. 3. An example of face-entity mismatch where Latvian president Raimonds Vejonis is incorrectly identified

the identity proposed by the face recognition algorithm is equal to the identity image belongs. Although the idea of the post-processing step is simple, it proved to be very effective since it detected that approximately 10% of images in training dataset were face-entity mismatches, thus helping to increase the quality of the training dataset.

As a result, by performing our identity extraction algorithm we managed to extract 4400 unique person names with a total of 95,000 images from LETA archive.

3.4 Experimental Evaluations

In practice we have to determine three things: how does the number of images per person affect the overall accuracy rate, what are the optimal KNN parameters and what the recognition speed is. To answer these questions we performed a number of experiments, and rated the results on a test dataset containing 1255 faces where 892 are faces of known persons of 186 identities and 363 faces are faces of unknown persons. All of the images were selected from LETA image archive.

To find an answer to the first question, we trained 50 different models having randomly selected 1 to 50 images per person in the training dataset accordingly. Figure 4 represents the obtained results, where X axis represent the number of images per person and Y axis represent the error rate on our test dataset. The error rate is computed as total correct images (known and unknown) divided by total images (892 known and 363 unknown). We can see that 6% error rate is reached when there are at least 6 images per person, 5% error rate when there are at least 24 images per person and 4.2% error rate (the lowest) when there are at least 45 images per person. Thus, the more images per person we have, the more accurate results we get.

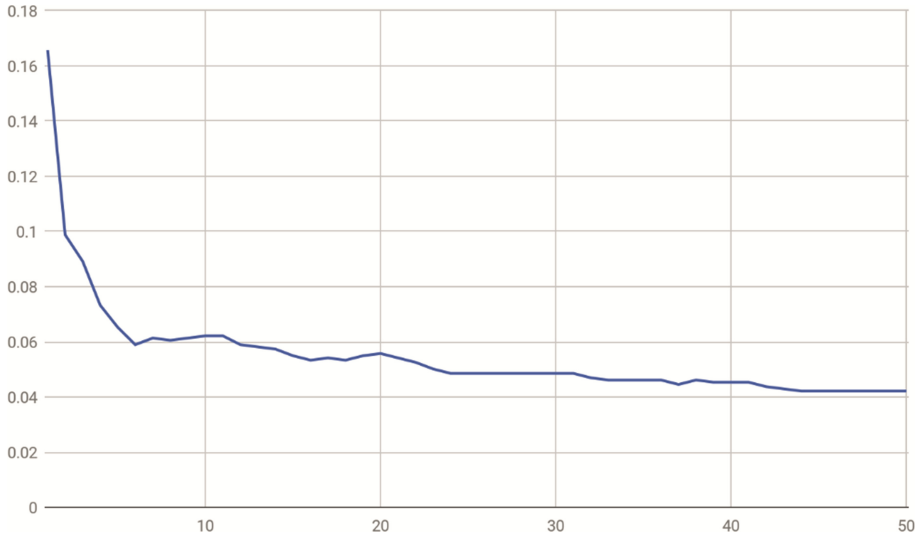

Fig. 4. An error rate depending on the images count per person

Figure 5 represents the error rate only among known persons.

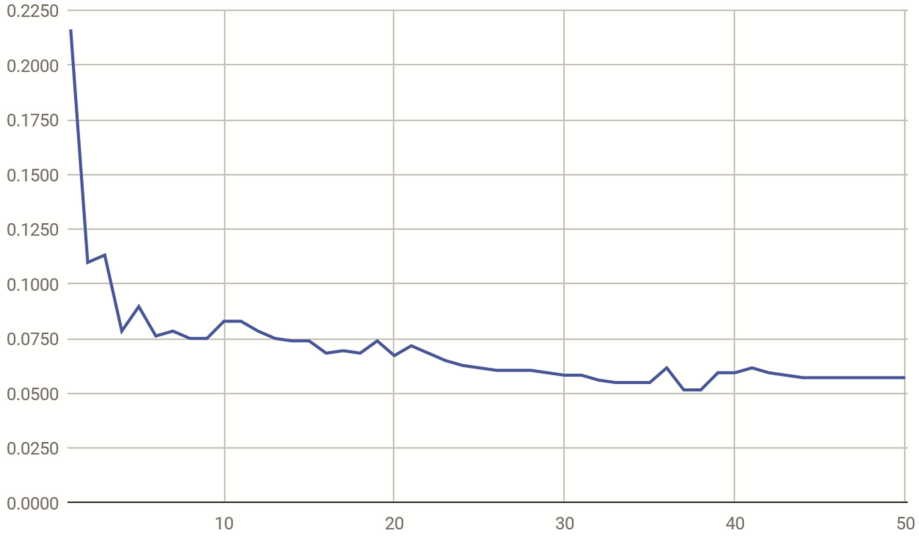

Fig. 5. An error rate of know persons depending on the images count per person

Figure 6 represents the error rate only among unknown persons.

We have also tested SVM algorithm on a dataset containing only images of known persons (see Fig. 7), and we can see that SVM results are slightly better than KNN results, still KNN is competitive, but with SVM we cannot detect unknown persons.

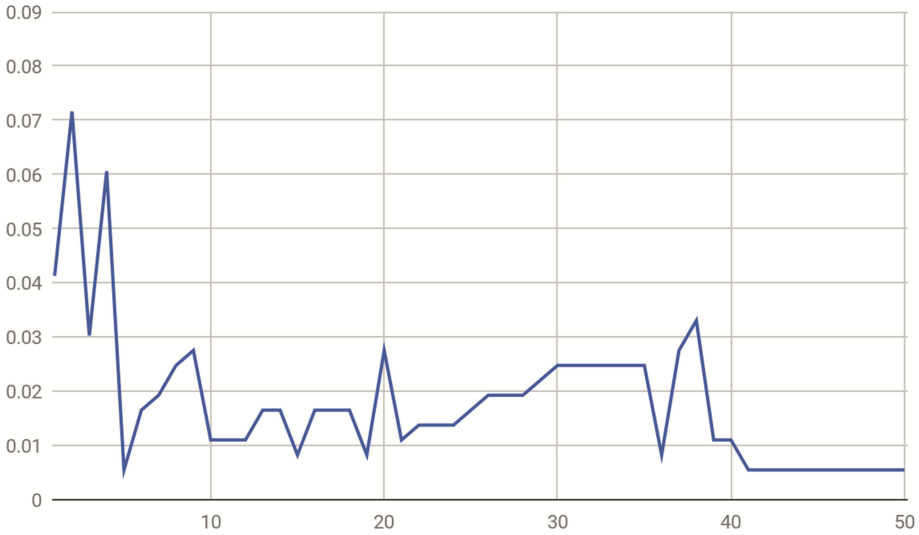

Fig. 6. An error rate of unknown persons depending on the images count per person

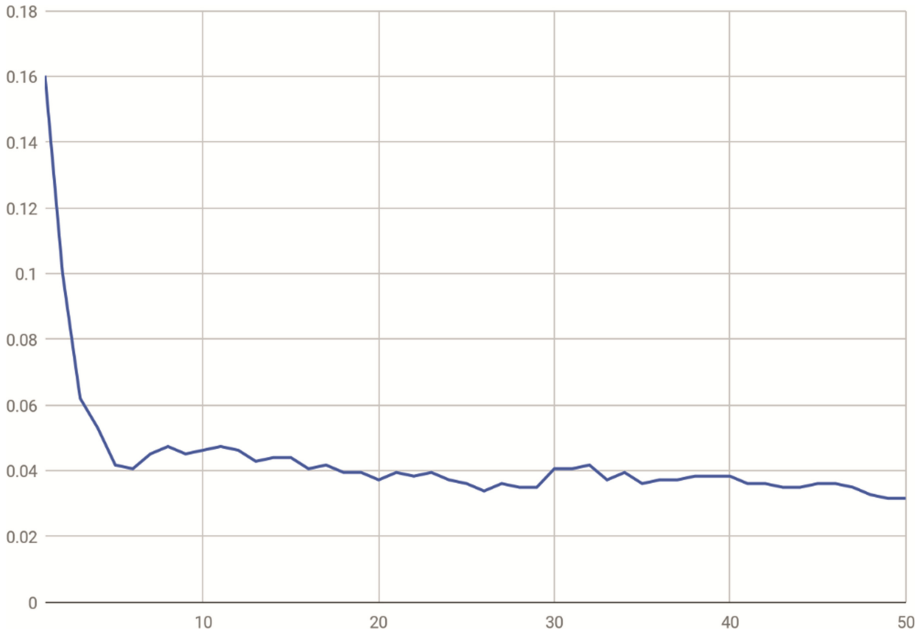

Fig. 7. An error rate of known persons depending on the images count per person using SVM algorithm

To find the optimal KNN parameters, we also performed the same 50 experiments having 1 to 50 images per person where count thresholds were from 1 to 4, and the

distance thresholds in range from 0.75 to 1 with a step size 0.05. Figures 8, 9, 10 and 11 represent how the accuracy changes depending on different voting and distance thresholds. Figures show that if the training dataset contains less than 10 images, then the distance threshold have to be relatively high to achieve the best performance. Whereas, if the size of the training dataset is getting larger than 10, then the tendency is that the best performance is achieved when the distance threshold is relatively small.

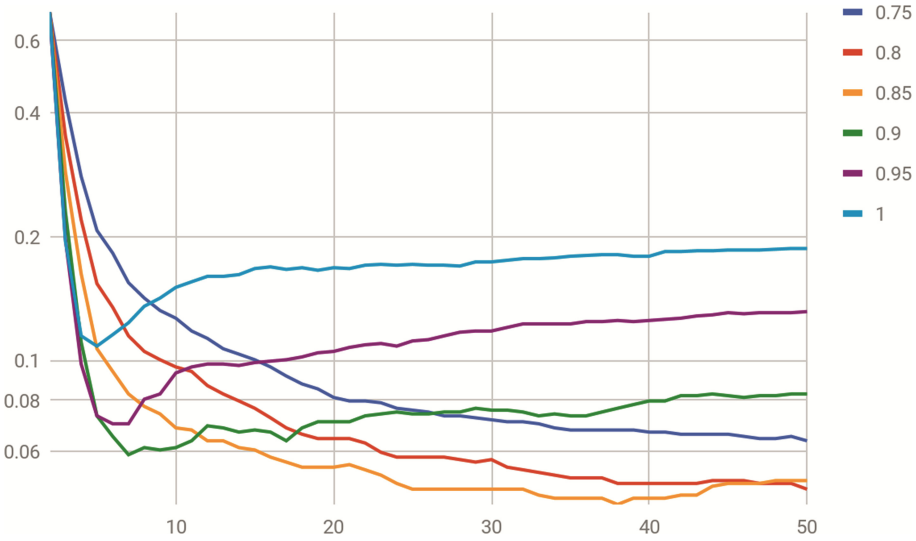

Fig. 8. Error rates with voting threshold 1 and various distance thresholds

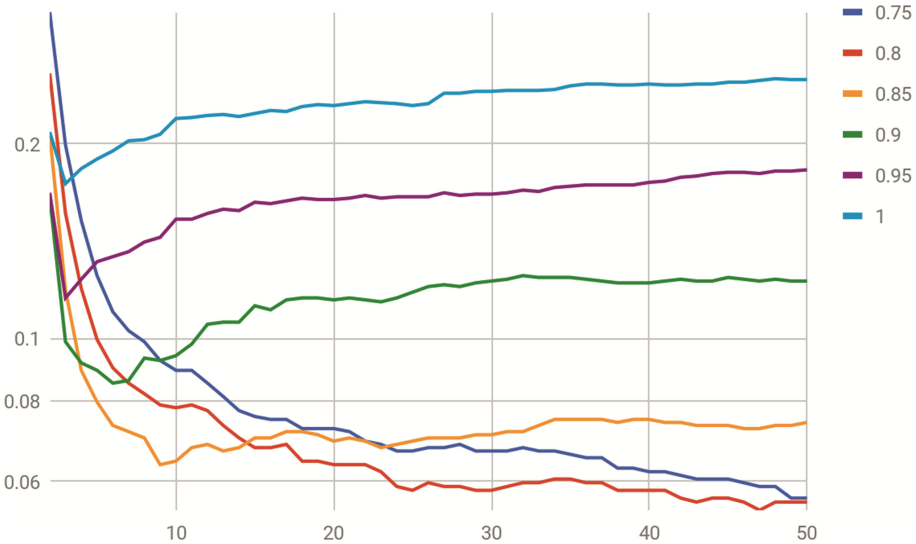

Fig. 9. Error rates with voting threshold 2 and various distance thresholds

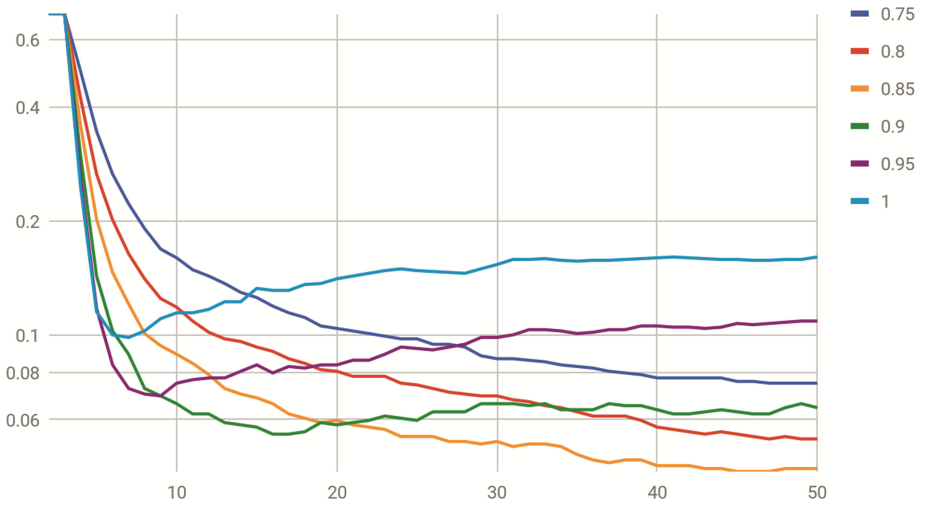

Fig. 10. Error rates with voting threshold 3 and various distance thresholds

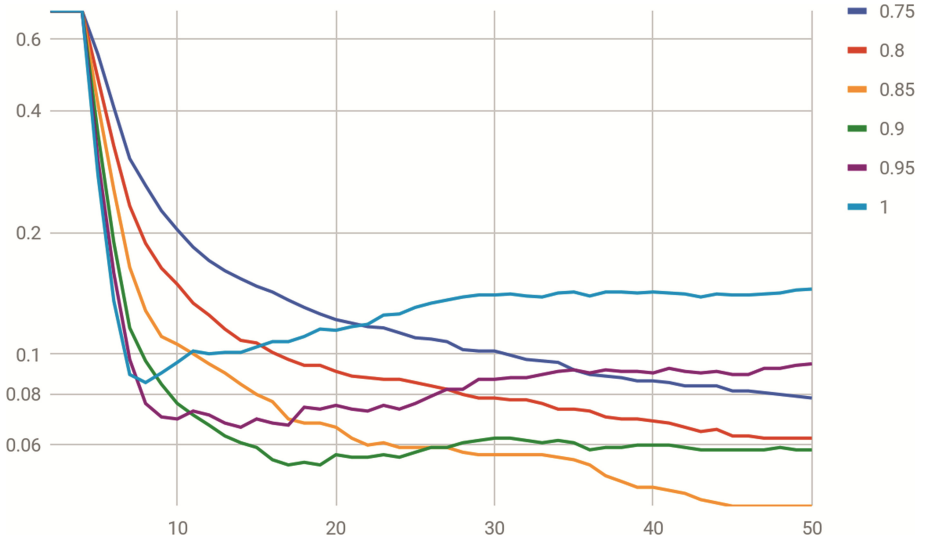

Fig. 11. Error rates with voting threshold 4 and various distance thresholds

Figure 12 represents average error rates by various distance thresholds and we can see that the smallest error is achieved having 0.85 distance threshold value.

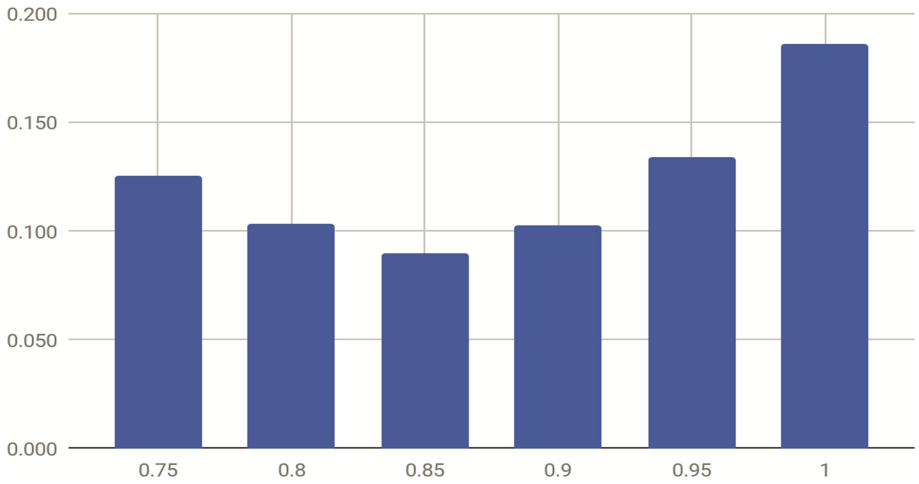

Fig. 12. Average error rates by various distance thresholds

Our experiments show that it takes on average 0.3 s to detect and recognize one face. The experiments were performed on NVIDIA GPU TITAN X 12 GB graphical processor.

4 Conclusions

In this paper we have discussed a general pipeline to implement a face recognition system for media companies and shared our experience implementing one for the Latvian national news agency LETA. To implement a system, we have to make a number of decisions, and the stack we have implemented in LETA project is as following. To compute face representation we selected the pre-trained model [25], for classification we used KNN algorithm with parameters depending on a number of images per person in the training dataset. To build a training dataset we used our custom made algorithm which automatically extracted 4400 identities with the total of 95,000 sample images from LETA image archive. We tested the implemented system on the dataset with 1255 faces where 892 were faces of known persons of 186 identities and 363 faces were of unknown persons and the accuracy rate we achieved was 95.78% when there are at least 45 images per person in the training dataset. We have also tested the algorithm on a number of YouTube videos where Latvian politicians participated, and the accuracy rate stayed the same. Thus the implemented system is applicable on different datasets as well.

While implementing the system we performed experimental evaluations regarding optimal training dataset size and optimal classification algorithm parameters. Experiments show that we have to build a training dataset of images that contain at least 6 images per person but optimally they are 25–45 images per person, whereas KNN parameters have to be adjusted according to the image count per person in the training dataset.

Acknowledgement. The research leading to these results has received funding from the research project "Competence Centre of Information and Communication Technologies" of the EU Structural funds, contract No. 1.2.1.1/16/A/007 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 2.5 "Automated visual material recognition and annotation system for LETA archive".

References

1. Cao, X., Wipf, D., Wen, F., Duan, G., Sun, J.: A practical transfer learning algorithm for face verification. In: Proceedings of ICCV (2013)
2. Barkan, O., Weill, J., Wolf, L., Aronowitz, H.: Fast high dimensional vector multiplication face recognition. In: Proceedings of ICCV (2013)
3. Phillips, P.J., et al.: An introduction to the good, the bad, & the ugly face recognition challenge problem. In: FG (2011)
4. Taigman, Y., Yang, M., Ranzato, M.A., Wolf, L.: DeepFace: closing the gap to human-level performance in face verification. In: Computer Vision and Pattern Recognition (2014)
5. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
6. Joliffe, I.T.: Principal Component Analysis. Springer, New York (2002). https://doi.org/10.1007/b98835
7. Chen, D., Cao, X., Wang, L., Wen, F., Sun, J.: Bayesian face revisited: a joint formulation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7574, pp. 566–579. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33712-3_41
8. Xing, E.P., Ng, A.Y., Jordan, M., Russell, S.: Distance metric learning with application to clustering with side-information. In: Proceedings of the 15th Advances in Neural Information Processing Systems (NIPS 2002), pp. 521–528 (2002)
9. Huang, G.B., Learned-Miller, E.: Labeled faces in the wild: updates and new reporting procedures. University of Massachusetts, Amherst, Technical report UM-CS-2014-003 (2014)
10. Ding, C., Tao, D.: Robust face recognition via multimodal deep face representation. IEEE Trans. Multimed. **17**(11), 2049–2058 (2015)
11. FACE++. https://arxiv.org/pdf/1501.04690v1.pdf
12. Sun, Y., Liang, D., Wang, X., Tang, X.: DeepID3. face recognition with very deep neural networks (2014)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint: arXiv:1409.1556
14. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions (2014). arXiv:1409.4842 [cs.CV]
15. Sun, Y., Wang, X., Tang, X.: Hybrid deep learning for face verification. In: Proceedings of ICCV (2013)
16. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering (2015). arXiv:1503.03832 [cs.CV]
17. Daream. http://www.daream.com/
18. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38

19. Zagoruyko, S., Komodakis, N.: Wide Residual Networks (2016). arXiv:1605.07146 [cs.CV]
20. Srivastava, R.K., Greff, K., Schmidhuber, J.: Deep Learning Workshop (ICML 2015) (2015)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 1, pp. 1097–1105 (2012)
22. Liu, J., Deng, Y., Bai, T., Wei, Z., Huang, C.: Targeting ultimate accuracy: Face recognition via deep embedding (2015). arXiv:1506.07310v4 [cs.CV]
23. Dahua-FaceImage. http://www.dahuatech.com/recognition/index.php
24. OpenFace. http://cmusatyalab.github.io/openface/
25. Face recognition using Tensorflow. https://github.com/davidsandberg/facenet
26. CASIA-WebFace. http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html
27. FaceScrub. http://vintage.winklerbros.net/facescrub.html
28. MS-Celeb-1M. https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/
29. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
30. Paikens, P.: Latvian newswire information extraction system and entity knowledge base. In: Human Language Technologies – The Baltic Perspective. Frontiers in Artificial Intelligence and Applications, vol. 268, pp. 119–125. IOS Press (2014)