Comparing the Performance of Deep Learning Face Recognition Models when used on Images of Varying Quality

Bas van 't Spijker (s1497944) Keanu Nurherbyanto (s2430681)
Vitomir Pavlov (s2420457) Akshay Prasad (s2433583)

***Abstract*—In this paper, we evaluated the performance of several CNN-based face recognition models on facial images that we distorted in terms of either resolution, compression, brightness, or noise levels. We handpicked images of 30 individuals at 9 different angles from the PUT dataset and applied the distortion, then used the Multi-task Convolutional Neural Network (MTCNN) to extract the face from each image before calculating feature vectors and determining similarity scores using the cosine distance. To compare the performance of each model we presented the outcome of the experiments in Area Under the ROC Curve (AUC) graphs and tables. Our results show that the ResNet-50 model trained on the VGG-Face2 dataset has the best overall performance.**

I. Introduction

Recently, government agencies have adopted the use of biometric systems such as face recognition for use in law enforcement and other security areas [1]. *Face recognition* or *face verification* is the process of finding a match between an individual's face image with a set of face images from a database [2], [3]. Individual face images are fed into feature extractors to be converted into their vector representations. Vector representations between two face images are compared to produce a similarity score, usually in terms of its *cosine distance*. State-of-the-art feature extractors, or face recognition models, implement deep learning. They are reported to possess high recognition performance, and we intend to investigate whether their performance decreases when subjected to low quality images, since face recognition models tend to be prone to this [1], [3]–[5].

The authors of most of the face recognition models discussed in this study presented a performance comparison of their novel approaches against previous methods [6]–[12]. Other studies such as Golla [5] only evaluated the performance of a single CNN-based face recognition model with varying low-quality images. To the best of our knowledge, comparative studies on the performance of all these face recognition models next to each other are still scarce. One recent study compared the face recognition performance of DeepFace, FaceNet, OpenFace, and VGGFace on masked face images [13]. These face recognition models were tested on images taken from the Real World Masked Face Recognition Database (RMFD). It was not mentioned how each model was implemented, and the authors refrained from elaborating on the performance metrics used. They did present a performance analysis in the form of the error rate of each model, the precision of each model, and the elapsed time required to process a single pair of test images.

In this report, we evaluate the performance of several state-of-the-art face recognition models that are based on deep learning networks, using reference and test images of varying quality. The four quality parameters chosen are compression, resolution, brightness, and noise. To streamline the process, the evaluations are done using a Python library called `deepface` [14] that provides easy access to different face recognition models. Performance benchmarks are represented by Area under ROC (AUC) graphs and tables. In section II, we present a literature review of relevant works, including short descriptions about each face recognition models. Then in section III, we describe test image dataset and our evaluation setup to measure the performance of each face recognition model using the `deepface` Python library. Afterwards, we present our benchmarks and evaluation in section IV. Finally, we discuss the limitations of this study in section V and discuss avenues for further studies in section VI before concluding.

II. Literature Review

A. Biometrics System Performance Metrics

The performance of a biometric system can be represented by a Receiver Operating Characteristic (ROC) curve. An ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) of a biometric system at different thresholds [3]. To visualize multiple ROC benchmarks of a biometrics system in one image, Area Under the (ROC) Curve (AUC) values can be used, such as in Dutta's research [4]: we take a very similar research approach to the one in that paper.

The performance of a face recognition system can be affected by the quality variability of images [3]. Some of the most common challenges that can negatively affect the performance of face recognition systems are illumination and pose variation [1]. However, this does not rule out other possible parameters such as image resolution and compression.

B. Convolutional Neural Networks

Neural networks are used in a deep learning method that trains machines to recognize patterns and create classifications from training data, so that it can classify future data using the model produced during training [15]. A neural network parses input through a series of nodes, where the data is processed by a function that has weight and bias parameters. These weights and biases can change in each training iteration based on the result of parsing each training output into a *loss function*. The goal of the loss function is to reduce the difference between the expected output and the actual output of the CNN model. A lower loss function output means that the model has a higher probability of making accurate classifications from an input.

One type of neural network that is often used in computer vision tasks is a Convolutional Neural Network (CNN) [16]. A CNN consists of multiple convolutional layer and pooling layer sets. The convolutional layer first extracts features from an image using kernels and collects them as weighted sums, then this sum is parsed into a non-linear function such as a Rectified Linear Unit (ReLU). Next, the features outputted from the convolutional layer are fed into the pooling layer, where multiple similar features are merged into one. A score is computed in each pooling output. The convolution and pooling stages are repeated until a score for a given classification is outputted in the final iteration, also called a feature vector. These feature vectors are what we mainly care about when later discussing face recognition models.

C. MTCNN

The face detection method that we used in our experiments is Multi-task CNN (MTCNN) proposed by Zhang et al. [17]. The author's goal was to come up with a unified cascaded CNNs using multi-task learning solution in order to improve the accuracy of the face detection and enhance the face alignment performance. With the multi-tasking assistance, they succeeded in creating a multi-task learning framework which is based on three stages:

1) In the first stage, they use trivial CNN to obtain the candidate windows and their bounding box regression vectors. They then estimate the weight of these vectors and apply a non-maximum suppression (NMS) to combine the overlapped candidate windows.
2) The second stage involves passing the candidate windows to another improved CNN which they call Refine Network. The goal of the Refine network is to skip false candidates using calibration.
3) The last stage is analogous to the previous stage, but it uses an even more powerful CNN, that characterizes the face in more detail while also outputting five different facial points that triangulate the eyes, nose, and mouth corners of the subject.

In addition to this three-stage multi-task training framework, they intended to improve face detection performance by reducing the number of filters for the images. To increase the discrimination capabilities of MTCNN, filter sizes are reduced from a 5x5 kernel to a 3x3 kernel. The smaller filter size decreases the computing cost, allowing for increased depth in each CNN stage. Moreover, they used a sample mining method for face detection and alignment in order to further improve the framework process. The goal is to filter out the helpful samples, called hard samples, and discard the useless ones. This filtering helps to enhance the detector during the training process.

D. DeepFace

A team of researchers at Facebook created DeepFace by training a fine-tuned CNN using the Social Face Classification (SFC) dataset [6]. Prior to training, images from the dataset were aligned using 2D and 3D alignment procedures. Unlike traditional CNNs, DeepFace was trained using only two convolutional layers with a single pooling layer separating them. After the second convolution layer, subsequent locally connected layers extract features from different sets of kernels. Then, vector representation of images are extracted in the last two fully connected layers. Finally, these vector representations are fed into a loss function called the *softmax* function to determine the probability distribution between each training image class. The authors of DeepFace argue that training a model using multiple convolution-pooling layers, such as in traditional CNNs, can negatively affect the detail of facial structures and textures, hence their justification for fine-tuning the training phase. Metric verification for DeepFace was done using the Labeled Faces in the Wild (LFW) and the YouTube Faces (YTF) dataset, where benchmarks show that DeepFace achieved an accuracy of 97.25%.

E. DeepID

The Deep Hidden IDentity (DeepID) face recognition model was created as a competitor to DeepFace by a group of researchers from Hong Kong [7]. The model was created by training a CNN using 10,000 images taken from the CelebFaces dataset. Features in the CNN are extracted by four convolutional layers. After the four convolutions, results are fed into a fully connected hidden layer with a dimension of 160 called the DeepID layer. Each convolution layer and the last hidden layer implement a specific convolution function followed by a non-linear ReLU function. Results from their tests show that DeepID possesses an accuracy of 97.20% when the model is used to conduct face recognition on the LFW dataset. To increase the accuracy of the model further, the authors also trained the DeepID model by implementing a transfer learning (TL) process with a joint Bayesian model. Repeating the test on the LFW dataset using the TL-combined DeepID model yielded 97.45% accuracy.

F. FaceNet

A year after the inception of DeepFace, a team of Google researchers developed a face recognition model dubbed FaceNet [8]. The FaceNet CNN was trained using the LFW and YTF datasets. In their paper, the authors of FaceNet describe training and evaluating their model using two CNN architectures;

Zeiler & Fergus, as well as an inception model architecture based on GoogLeNet. [After inspecting the base model files, we found that the FaceNet models provided in the deepface library [18] use the latter CNN architecture with an input size of 160x160.] A batch of training images are fed into the CNN, then the scores are normalized with L_2 normalization to produce embeddings. The training images are categorized as images with proper true / false classification results called anchor values, images with only true classification results called positive values, and images with only false classification results called negative values. These embeddings are later inputted into a *triplet loss* function. The triplet loss function reduces the distance between the training results of an anchor value and a positive value. By default, FaceNet creates a model with an embedding dimension of 128, but it can also create a model with an embedding dimension of 512. A model with larger embedding dimensions requires additional training time in order to perform as accurate as a model with a smaller embedding dimensions. FaceNet is said to have 95% face recognition accuracy when trained on the YTF dataset, and 96% accuracy when trained on the LFW dataset.

G. VGG-Face and VGG-Face2

The Visual Geometry Group (VGG) from the University of Oxford curated two public datasets for face recognition that they named VGG-Face [9] and VGG-Face2 [10]. VGG-Face consists of multiple images each from 2,662 individuals, while VGG-Face2 contains images from 9,131 distinct individuals. Besides the larger sample space, the VGG-Face2 dataset also differs from its predecessor by having images with specified pose, age, and illumination variations.

These two datasets are used to created face recognition models by training different CNNs. In their first paper [9], the VGG-Face dataset is used to train a "very deep" CNN. Afterwards, the model is compared with other state-of-the-art face recognition models that includes DeepFace, FaceNet and DeepID. The VGG-Face-trained model performed on par with its competition. In their second paper [10] VGG-Face and VGG-Face2 are used to train a CNN model called ResNet-50. In terms of face identification capability, the VGG-Face2-trained model was able to predict the correct classification with higher probabilities than than the VGG-Face-trained model. When the two models are used on an external dataset for an image verification task, the VGG-Face2-trained model exhibits higher levels of TPR than the VGG-Face-trained model. The higher performance of the VGG-Face2-trained model can be attributed to the larger variety of the images used as a training set for the CNN [10].

H. OpenFace

OpenFace is an open source face recognition library intended for mobile applications. It uses a modified CNN architecture of the one presented in FaceNet [11]. Instead of 160x160 pixel images, the CNN model implemented in OpenFace accepts 96x96 images as training input. As mentioned in the FaceNet paper [8], a smaller input size reduces the amount of computing power required to train the CNN. Besides a smaller training image size, the OpenFace CNN was trained using less layers and parameters. The model was trained using 500,000 face images taken from the combined datasets of CASIA-WebFace and FaceScrub. It was then subjected to an image verification task on images taken from the LFW dataset, where it was reported to achieve an AUC of 0.973.

I. ArcFace

ArcFace is a proposed novel loss function that intends to improve the discriminative capabilities of CNNs [12]. ArcFace measures the angle between a calculated feature score and the target weight via an arc-cosine function. The authors of ArcFace suggest that ArcFace can be used with various types of CNNs, which they emphasize in their paper by describing their evaluation setup that integrates ArcFace in the training pipeline of a ResNet-50 and ResNet-100 CNN. The CNNs were trained using the CASIA, VGG-Face2, MS1MV2, and DeepGlint-Face datasets. To evaluate its performance, the produced CNN model was then subjected to an image verification task involving the LFW, CFP-FP, and AgeDB-30 datasets. The results show that the face recognition capability of ArcFace surpasses VGG-Face2 with an accuracy of 99.82% on the LFW dataset. The ArcFace model provided in the deepface library was created by training a ResNet-34 CNN. The difference between the separate ResNet models is in the number of convolutional layers used to create the model.

J. Dlib

Dlib is a machine-learning toolkit that has a powerful face detection and image alignment component [19]. It is written in C++, but other implementations such as Python are available. The Dlib-based face recognition model provided in the deepFace library was created by training a ResNet-34 CNN with training images taken from the FaceScrub and VGG-Face2 dataset [20]. Afterwards, it showed 99.38% accuracy on an image verification task using test images from the LFW dataset.

III. Methodology

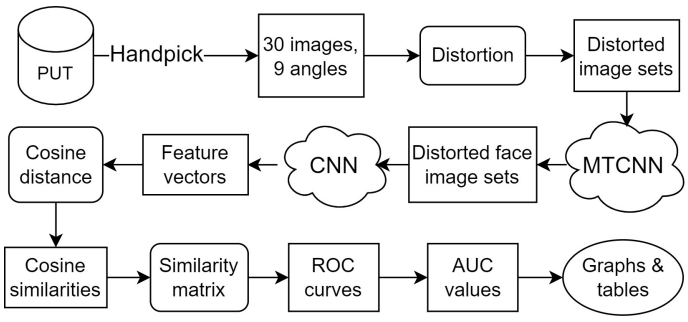

Fig. 1. Diagram of the face recognition model evaluation process.

See figure 1 for a summary of our evaluation setup.

A. Dataset and Image Preprocessing

1) Procedure and parameter values: We handpicked face images from 30 subjects in the PUT face database [21]. For each subject, we selected images with 9 different angles of the subject facing left to right, taking image 5 as the most facially centered one. To simulate real-world face recognition that involves low quality images, we artificially created distortions in our dataset using the scikit-image Python library to add 5 different levels of noise, or the Pillow image processing library to produce images with 5 different levels of compression, resolution, or brightness. After images were altered, we used the MTCNN face detector provided in the deepface framework to localize the face of each subject. The result is a collection of images with dimensions of 224x224. Figure 2 shows the selected face angles and variations on the quality of an individual subject's face image.

Compression is applied with the Pillow library, which uses default JPEG image compression given a quality parameter [22] [23]. JPEG image compression is a type of lossy compression that, amongst other algorithms, mainly uses Discrete Cosine Transform to reduce image file size. The quality parameter can range from 1 to 100, 95 representing original quality, and lower numbers representing higher compression. We used compression quality values of 95, 9, 7, 5, 3, and 1.

To decrease resolution, a new pixel width is specified when saving the images using the Pillow library. Pillow automatically scales pixel height to match the original image dimension ratios. We used pixel widths of 2048, 1024, 512, 256, 128 and 64, representing image size ratios of 1 to $\frac{1}{32}$ respectively.

To decrease brightness, the Pillow library linearly interpolates images with a purely black image [24], using a value α to set interpolation weights. An α of 0 produces the black image, and an α of 1 the original image. To increase brightness, α is set to > 1 with no maximum, and the original image is extrapolated with the black image instead of interpolated. We used α values 0.1, 0.5, 1, 1.5, 3, and 5.

To apply noise, scikit-learn uses a Gaussian distribution with a given degree of variance σ^2, determined by a set standard deviation σ. Standard deviation values used are 0, 0.1, 0.3, 0.5, 0.7 and 1.

2) MTCNN limitations: In our experiments we discovered that in some cases the MTCNN failed to detect faces. The MTCNN face detector calibrates a bounding box to isolate a face and build facial landmarks that triangulate the eyes, nose, and mouth corners of a subject [17]. As seen in the fourth line of images in Figure 2, the MTCNN failed to detect faces for images with very high or very low brightness because it could not properly isolate a face and determine the required facial landmark points. For the highest noise level this failure also occurred, but only with 6 images of subject number 15. In some high brightness cases the MTCNN even malfunctioned and zoomed in on a visual artefact. This has serious implications for the tests results:

- When comparing two faces, the CNNs have a harder time determining similarity when one image is full-size and the other one is a face image, compared to when both are in the same format. This means that scores involving high brightness could have been higher if we had removed the MTCNN component from our pipeline entirely in those speficic parts.
- Similarly, (random) visual artefacts outputted by the MTCNN instead of facial images decrease accuracy scores of models.

In short, performance values in extreme distortion cases could have been better if we had manually intervened in the image recognition process. The benefit of not having done so is that the results are more realistic in representing fully automated scenarios.

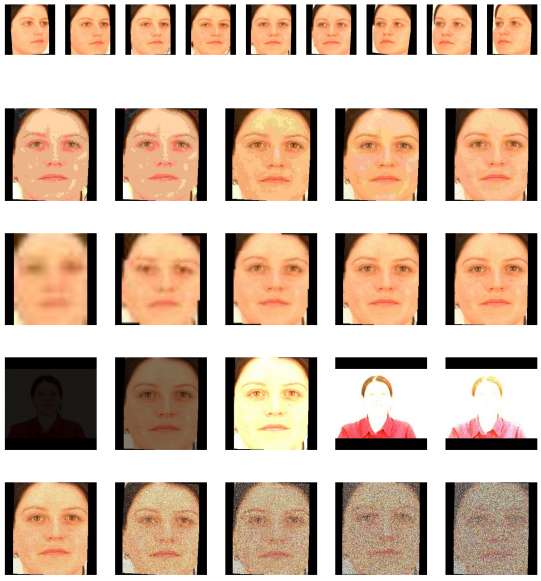

Fig. 2. Sample front-facing and left-facing angles (line 1), followed by various levels of image compression (line 2), resolution (line 3), brightness (line 4), and noise (line 5).

B. Comparing Face Recognition Model Performance

To compare the performance of DeepFace, DeepID, FaceNet, VGG-Face, OpenFace, ArcFace, and Dlib given our dataset, we used the corresponding models provided in the deepface library [14][1] to conduct three image verification experiments. In addition to these models, we implemented a VGG-Face2 trained ResNet-50 model and included it in our experiments as well. The first experiment aims to compare the performance of each model when front-facing reference images are paired with front-facing test images. The second experiment was conducted similarly, but with front-facing reference images paired with left-facing test images instead. The third experiment analyzes the performance of the highest performing model more in-depth, visualizing all possible different face angle combinations.

[1]To be clear, the Python library called 'deepface' and the Facebook model 'DeepFace' are two separate entities, but deepface does contain the DeepFace model

Each face image is parsed into a face recognition model, which results in an *embedding*, or a vector representation of each image according to the face recognition model. We calculated the *cosine distance* between the embeddings of a reference and test image to determine their *cosine similarity*, a value between 0 and 1. A lower cosine similarity means the faces are deemed to be more similar. The deepface library takes a single reference and test image at a time, produces embeddings using a specified model, and calculates a cosine similarity score. These scores were put in face comparison matrices. The True Positive Rate (TPR) and False Positive Rate (FPR) values are calculated from each of the similarity score matrices, and used to create AUC graphs. An ROC graph plots the TPR and FPR values to visualize the performance of a biometric system [3]. A high performing biometric system would show a TPR close to 1 on the ROC y-axis when the FPR on the ROC x-axis is close to 0; the area under the ROC graph would be larger for a high performance biometric system. Hence, we believe that it is more convenient to use AUC values when quantifying the performance of multiple cases, such as done in Dutta's paper [4].

The project and its source code are available in a public git repository [25].

IV. Experiments & Results

A. Experiment 1: comparing a central reference face to a central test face (5-5)

The results for the first experiment can be seen in Figure 3. When verifying faces using images with different resolution ratios or compression values, VGG-Face and VGG-Face2 exhibit the highest AUC value for all variants, with VGG-Face2 slightly outperforming its predecessor. The other face recognition models show moderately lower AUC scores than the two aforementioned models, but do generate AUC scores of around 0.9. We observed that all models besides VGG-Face2 show a drop in AUC when test images with resolutions lower than 512 pixel width are used. The AUC increases for a 64 pixel width test image when the reference image's resolution is on the lower end as well.

When comparing images with varying brightness levels, AUC values were relatively high at values > 0.8 when brightness values were inbetween 0.5 and 1.5, with 1.5 taking a slightly bigger hit than 0.5 in the worse performing models. Setting brightness values to 3 or 5 decreases performance significantly, with all but the best performing models quickly dropping to an AUC of 0.5 (which is the worst possible AUC value, equal to randomness). We can also clearly observe here that when brightness values are the same, the AUC is always 1. In this experiment, that value does come from a fair assessment: since we do not have two separate pictures for each subject for central (angle 5) faces, for the positive matches we are just comparing pictures with themselves. This pixel-by-pixel identicality means that the cosine similarity is always 0 in those cases. However, even when factoring that in, comparing pictures of extreme high or low brightness values generally gave slightly higher AUCs when both pictures were close in brightness value.

For noise, the only model that could handle noise levels up to 0.5 was VGG-Face2, other models started seeing a reduction in performance starting from σ=0.3 or 0.5. The trend of peak AUC values at matching brightness values can also be observed when comparing images with the same or similar levels of noise distortion. Likewise, comparing images with least noise and most noise produces the worst AUC scores for each model.

In summary, the VGG-Face2 model produces the best AUC variations for the compression, resolution and brightness tests. However, DeepID and especially DeepFace in particular show good overall resiliency against noisy face images.

B. Experiment 2: comparing a central reference face to a left-facing test face (5-1)

The results for the second experiment can be seen in Figure 4. When comparing these results with those of experiment 1 in Figure 3, it can be observed that the AUC scores for all the models apart from VGG-Face and VGG-Face2 are noticeably lower. This decrease can be attributed due to the now differing reference angle and test angle.

The apparent lead in terms of AUC scores that the two top models exhibited in the previous experiment is still present here. Likewise, DeepID and DeepFace are still the most stable models when dealing with noisy images. Other noticeable observations are consistent with the results from the previous experiment. For example, AUC scores tend to decrease when comparing test images with resolutions of pixel width lower than 256. Once again, the lowest AUC scores are produced when the darkest images are compared with the brightest, as well as when comparing images with the most noise to images having no distortions.

Another observation can be made is that when the best case AUC values are already (significantly) lower than 1, the drops in AUC due to image distortion are relatively smaller. For example, the AUC drop of the Deepface model when resolutions approach 64 in width is relatively bigger in experiment 1 than in experiment 2, but in experiment 2 the AUC values are already much closer to 0.5.

Lastly, from the graphs it seems that when comparing a central-looking face to a left-looking face, it does not matter which image is the more distorted one. In general, the best AUC values occur when one of the two images is of high quality, or if the distortion parameters are close to each other. The latter especially holds true in the most extreme distortion cases.

C. Experiment 3: a more in-depth look at VGG-Face2 through face angle tables

The results of the final experiment are presented in a collection of AUC score heatmaps that can be seen in Figures 7, 5, and 6. We selected VGG-Face2 for this experiment because it showed the highest performance overall for all image quality categories.

The AUC scores for the compression and resolution variation experiments show consistent results that support data from the previous 2 experiments. For all levels of compression and resolution distortion, VGG-Face2 was able to produce very high AUC scores for all face angles.

When comparing images with different brightness levels, a concentration of high AUC scores, visualized by the red-colored heatmap squares, can be observed in Figure 5 at points where the reference and test images have a brightness value between 0.5 times less and 0.5 times more than the original brightness level. When comparing the furthest face angles with each other, i.e. comparing face angle 1 against 9, this concentrated heatmap is still present, albeit not as concentrated. When test and reference images have the same angle and brightness levels (meaning the images are exactly the same), VGG-Face2 produces high AUC scores, just as we could see in the previous experiments.

When comparing noisy images, the concentration of high AUC scores in Figure 6 are present at points where the reference and test images have a noise ratio between 0 and 0.5. The strongest concentration of AUC scores can be seen when the test and reference angles are relatively centered, such as for angles 4-6.

In general, we can say that the more centered both face images are, the better the verification accuracy will be when the images are distorted.

V. Limitations of The Study

Firstly, as mentioned before in the methodology section, during our preprocessing phase we discovered that MTCNN did not effectively detect faces for images with very high or very low brightness. As shown in Figure 2, the MTCNN algorithm outputs the full original image if it fails to detect a face. This limitation might affect the validity of our experiments involving images that have very low or very high brightness levels.

Secondly, to evaluate the models we used a specific `deepface` library function `verify()` that takes a single reference and test image at a time, produces embeddings using a specified model, and calculates the cosine similarity. Using this function was a technical mistake: because we compare each test image to all reference images, embeddings for every single picture are calculated multiple times over instead of just once, creating significant overhead. Processing large numbers of images this way was inefficient and required large amounts of time. Instead of `verify()`, we should have used the `represent()` function to calculate feature vectors just once instead, similarly to how we approached evaluating the VGG-Face2 model.

Thirdly, we selected images from only 30 individuals from the PUT dataset. The decision to do so was made because of two factors: picking face angles by hand was a time-consuming process, and calculating AUC scores for just 30 faces at 9 angles only took about 2 days already because of the previously mentioned technical error. This limited set of faces might reduce validity and reproducibility of our experiments.

Lastly, handpicking face angles from the PUT dataset does not only require a lot of time and effort, it also causes the experiments to be less precise. The face angles and their transitions from left to right were estimated by the research team members, so the angles have no exact degree values. The dataset itself also did not seem to have any strict angles defined when faces were photographed, the creators simply required the subjects to turn their heads roughly from left to right. This causes significant variance in the degrees of the sharpest photographed face angles between subjects. Future efforts at this type of research would benefit from requiring strictly defined face angles when selecting the dataset.

VI. Future Research

As we mentioned in section V, a more optimal `deepface` method should be used to compare larger number of individuals from a dataset, since the implementation in this research increases the processing time exponentially for every individual that is added for comparison. With more efficient processing, recreating the experiments with a larger number of faces is possible, which could provide sounder results.

Additionally, either selecting a dataset with defined face angles or creating an algorithm-based method that automatically selects images with defined face angles would increase validity and reduce required manual labor. Moreover, in this research we limited ourselves to horizontal head turns, but vertical variation in face angles, with subjects looking up or down, could also be researched.

Another opportunity for future research is to implement smaller transition steps for distortion variables that show a sharp decline in AUC values between points. For example, there could be more steps inbetween brightness values 1 and 3 or 3 and 5.

In this report we used different variables for the modification of the images such as compression, resolution, brightness and noise. These are just a small set of possible image distortions; more types of distortions could be researched, such as adding shadows in the picture, changing the image color, or applying fractal compression or any other type of compression technique.

Furthermore, while MTCNN seems a good state-of-the-art face detector, it would be interesting to compare it to other face detectors such as RetinaFace**??** or OpenCV**??**. Researching the performance impact of using such face detectors would relatively easy using the framework we provided [25].

Also, it could be interesting to combine distortion variables to simulate a worst-case scenario. What constitutes as a worst-case scenario would be heavily dependent on the use case context.

Lastly, more in-depth research and benchmarking could be done with the best-performing algorithms like VGG-Face2 (or rather, ResNet-50 trained on the VGG-Face2 dataset). Comparisons could be made between this CNN-based face recognition approach and other methods, such as Gabor wavelet-, Face descriptor-, or 3D-based solutions [26].

VII. Conclusion

In this report, we have provided performance benchmarks for multiple CNN-based face recognition models subjected to images with varying resolution, compression, brightness, and noise distortion levels. Our results show that the ResNet50 model trained with the VGG-Face2 dataset has the best overall performance compared to other models described in this report. In addition, our results show that images with different levels of resolution and compression distortion do not negatively affect modern CNN-based face recognition models as much as images with different levels of brightness and noise distortion.

References

[1] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2d and 3d face recognition: A survey," *Pattern Recognition Letters*, vol. 28, no. 14, pp. 1885–1906, 2007, image: Information and Control. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865507000189

[2] A. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, 2004.

[3] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[4] A. Dutta, R. Veldhuis, and L. Spreeuwers, "The impact of image quality on the performance of face recognition," in *33rd Symposium on Information Theory in the Benelux and the 2nd Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux*. Netherlands: Werkgemeenschap voor Informatie- en Communicatietheorie (WIC), May 2012, pp. 141–148, 33rd WIC Symposium on Information Theory in the Benelux and the 2nd Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux 2012 ; Conference date: 24-05-2012 Through 25-05-2012.

[5] M. Golla and P. Sharma, *Performance Evaluation of Facenet on Low Resolution Face Images: First International Conference, CNC 2018, Gwalior, India, March 22-24, 2018, Revised Selected Papers*, 01 2019, pp. 317–325.

[6] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[7] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1891–1898.

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[9] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.

[10] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2018, pp. 67–74. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FG.2018.00020

[11] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[12] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] Y. B. Chandra and G. K. Reddy, "A comparative analysis of face recognition models on masked faces," *International Journal of Scientific & Technology Research*, vol. 9, pp. 175–178, 2020.

[14] S. I. Serengil and A. Ozpinar, "Lightface: A hybrid deep face recognition framework," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2020, pp. 23–27. [Online]. Available: https://doi.org/10.1109/ASYU50717.2020.9259802

[15] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang, "Biometrics recognition using deep learning: A survey," 2021.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, p. 436–444, 2015.

[17] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[18] S. I. Serengil, "deepface," https://github.com/serengil/deepface, 2020.

[19] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, p. 1755–1758, dec 2009.

[20] S. I. Serengil, "Face recognition with dlib in python," https://sefiks.com/2020/07/11/face-recognition-with-dlib-in-python/, 2020.

[21] A. Kasiński, A. Florek, and A. Schmidt, "The put face database," *Image Processing and Communications*, vol. 13, pp. 59–64, 01 2008.

[22] J. W. O'Brien, "The jpeg image compression algorithm," *APPM-3310 FINAL PROJECT*, no. 4, pp. 4–7, 2005.

[23] M. S. Al-Ani and F. H. Awad, "The jpeg image compression algorithm," *International Journal of Advances in Engineering & Technology*, vol. 6, no. 3, p. 1055, 2013.

[24] P. Haeberli and D. Voorhies, "Image processing by linear interpolation and extrapolation," *IRIS Universe Magazine*, vol. 28, pp. 8–9, 1994. [Online]. Available: http://www.graficaobscura.com/interp/index.html

[25] "Biometrics," https://github.com/BCvS/biometrics, 2022.

[26] W. Wójcik, K. Gromaszek, and M. Junisbekov, "Face recognition: Issues, methods and alternative applications," *Face Recognition-Semisupervised Classification, Subspace Projection and Evaluation Methods*, pp. 7–28, 2016.

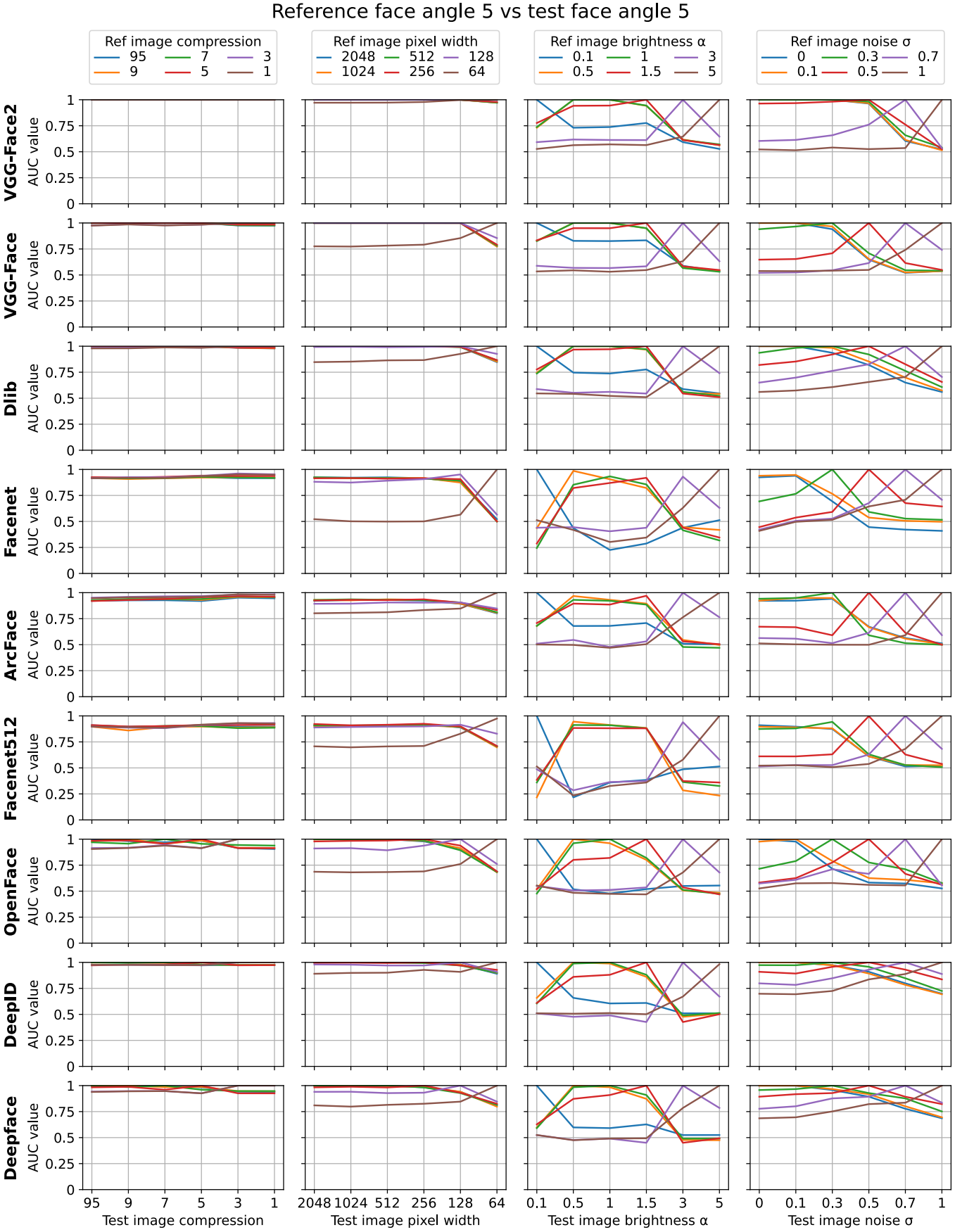

Fig. 3. AUC graphs of front-facing reference and test images.

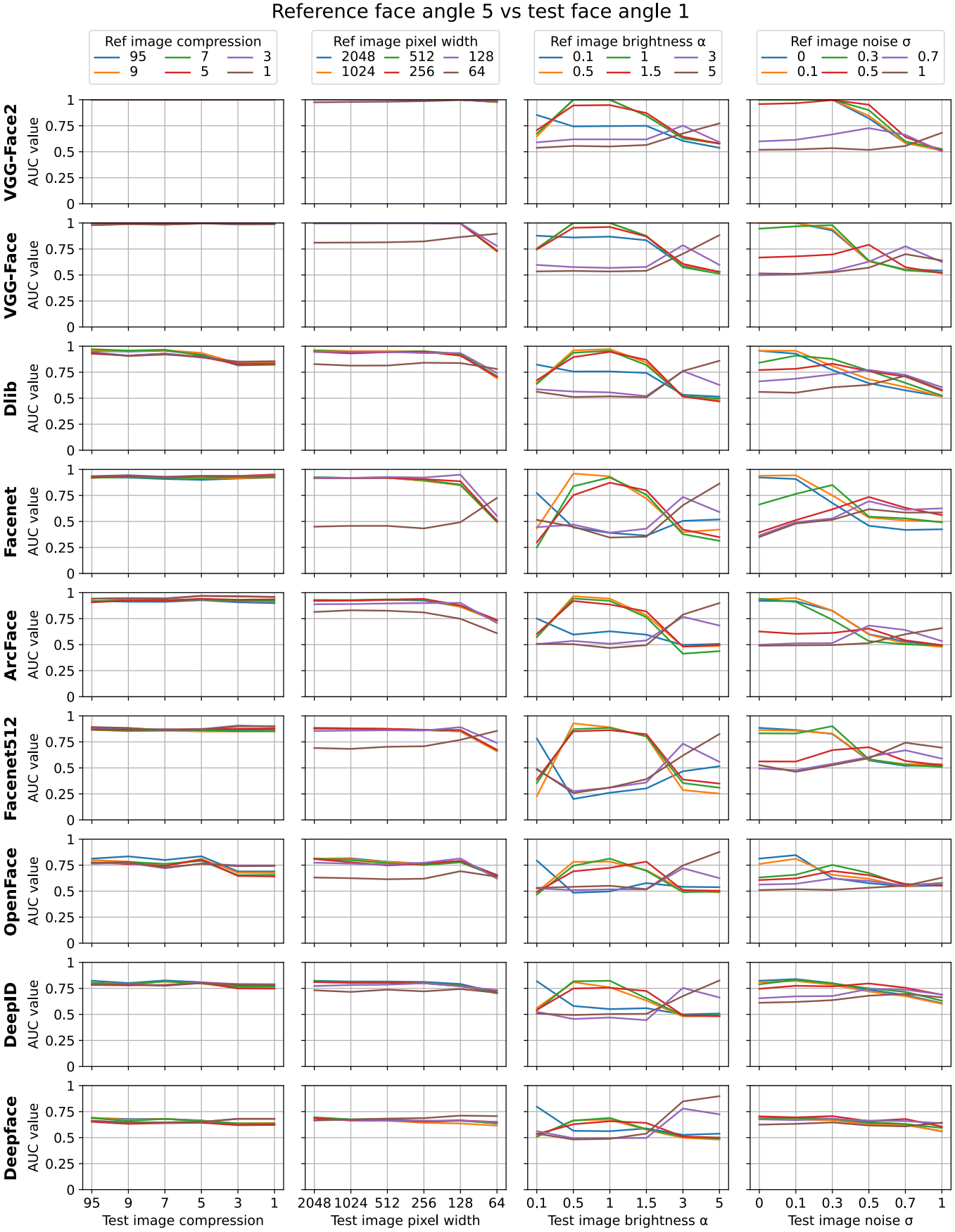

Fig. 4. AUC graphs of front-facing reference images and left-facing test images.

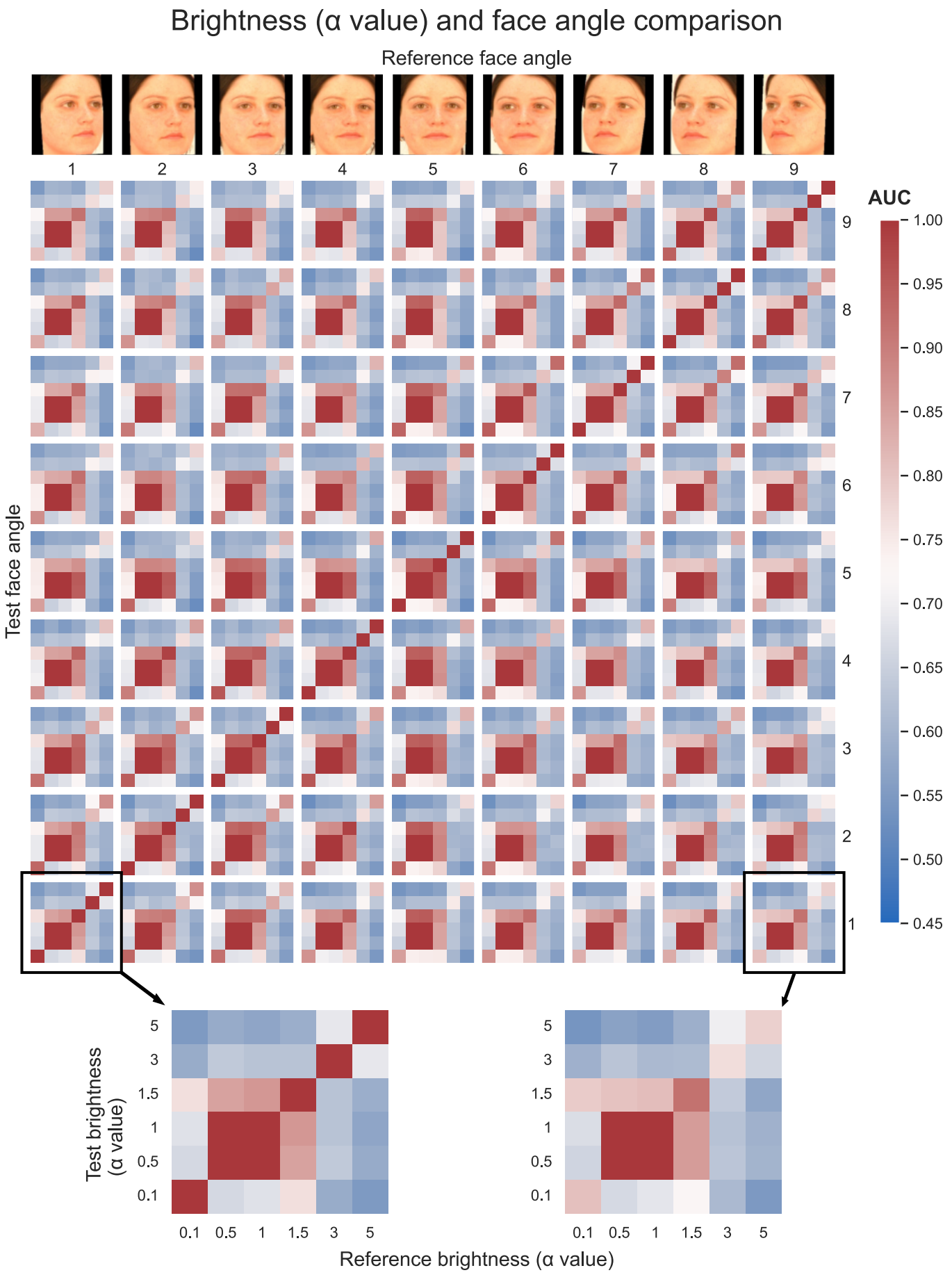

Fig. 5. AUC heatmap showing VGG-Face2 performance with varying image brightness values.

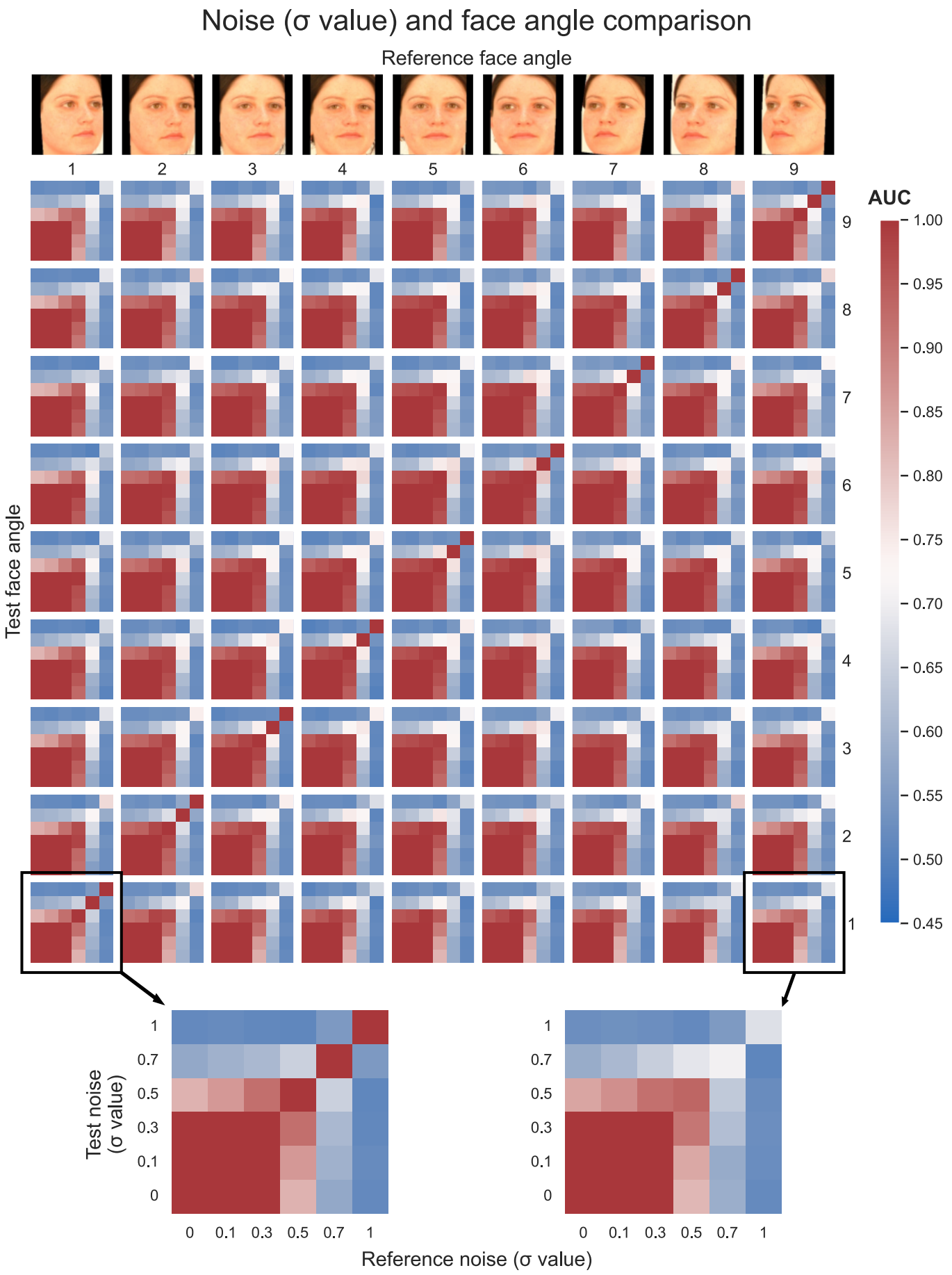

Fig. 6. AUC heatmap showing VGG-Face2 performance with varying image noise levels.

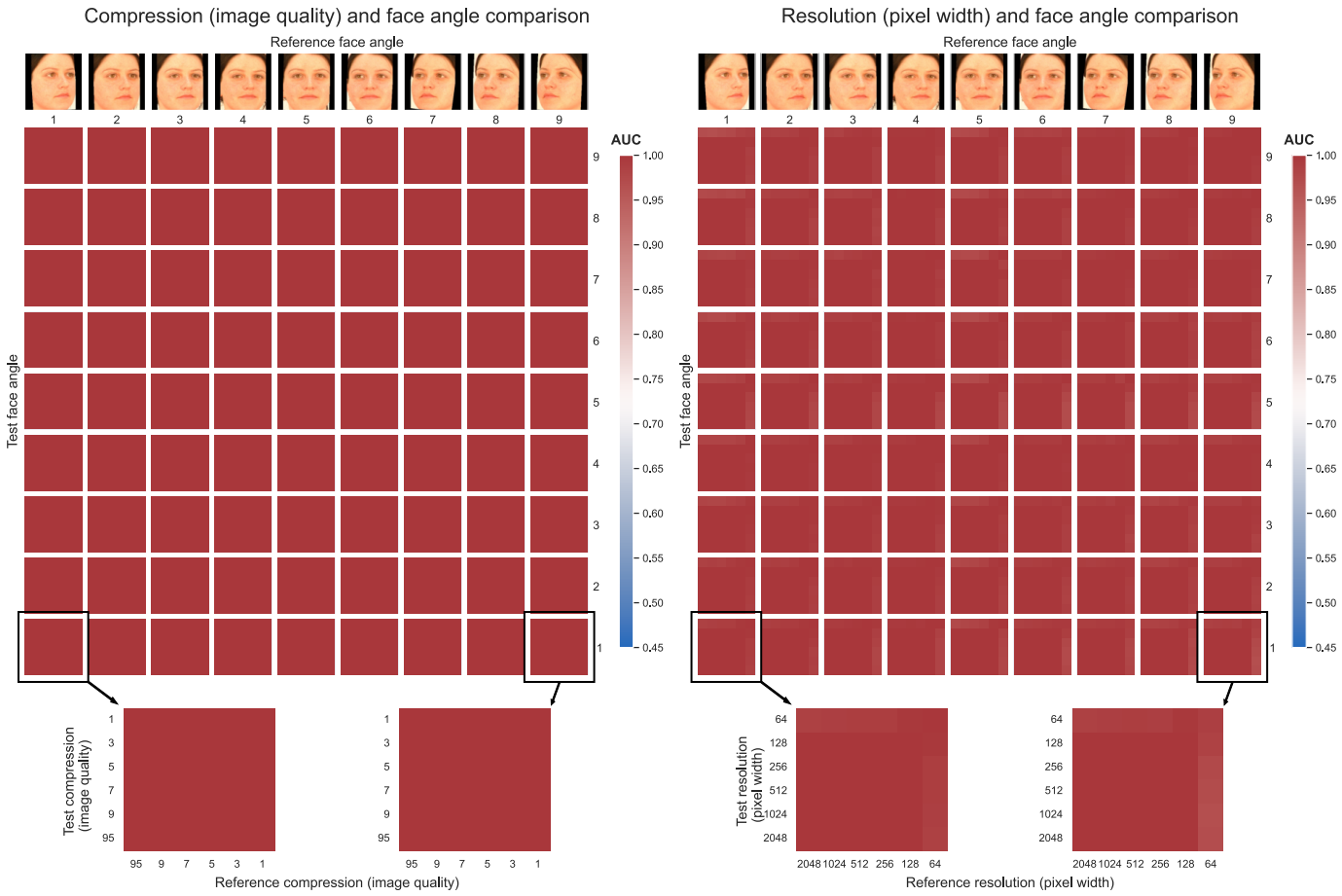

Fig. 7. AUC heatmaps showing VGG-Face2 performance with varying image compression and resolution levels.