Face to Face with Efficiency: Real-Time Face Recognition Pipelines on Embedded Devices[⋆]

Philipp Hofer[1], Michael Roland[1], Philipp Schwarz[2], and René Mayrhofer[1]

[1] Johannes Kepler University Linz, Institute of Networks and Security, Austria
{philipp.hofer, michael.roland, rm}@ins.jku.at
[2] Johannes Kepler University Linz, LIT Secure and Correct Systems Lab, Austria
philipp.schwarz@jku.at

Abstract. While real-time face recognition has become increasingly popular, its use in decentralized systems and on embedded hardware presents numerous challenges. One challenge is the trade-off between accuracy and inference-time on constrained hardware resources. While achieving higher accuracy is desirable, it comes at the cost of longer inference-time. We first conduct a comparative study on the effect of using different face recognition distance functions and introduce a novel inference-time/accuracy plot to facilitate the comparison of different face recognition models. Every application must strike a balance between inference-time and accuracy, depending on its focus. To achieve optimal performance across the spectrum, we propose a combination of multiple models with distinct characteristics. This allows the system to address the weaknesses of individual models and to optimize performance based on the specific needs of the application.
We demonstrate the practicality of our proposed approach by utilizing two face detection models positioned at either end of the inference-time/accuracy spectrum to develop a multimodel face recognition pipeline. By integrating these models on an embedded device, we are able to achieve superior overall accuracy, reliability, and speed; improving the trade-off between inference-time and accuracy by striking an optimal balance between the performance of the two models, with the more accurate model being utilized when necessary and the faster model being employed for generating fast proposals. The proposed pipeline can be used as a guideline for developing real-time face recognition systems on embedded devices.

Keywords: Face recognition pipeline · efficiency · embedded devices · face detection · inference-time/accuracy.

[⋆] This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World and has partially been supported by the LIT Secure and Correct Systems Lab. We gratefully acknowledge financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, 3 Banken IT GmbH, ekey biometric systems GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH & Co KG, Österreichische Staatsdruckerei GmbH, and the State of Upper Austria.

1 Introduction

Biometric authentication in physical environments is becoming more and more widespread. For the past decade, each Chinese person has been given a score that changes depending on how the person's decisions are in line with the government [15]. In India, each person is assigned a 12-digit number, which is continuously supplemented with biometric characteristics and is required for many parts of life, such as banking, travelling, and even being admitted to schools [20]. In the Russian metro, there is a face recognition payment system [17]. The EU plans an entry/exit system, where fingerprints and facial images of travelers from third-countries will be collected [6]. These applications paint a scary picture for a privacy conscious user; biometrics from millions, in some cases billions, of users are stored in a single logical location. All requests for information must go through that point, creating a single point of failure that can be exploited by attackers or misused by operators:

1. A central place with potentially billions of personal data is an excellent target for technical, legal, and organizational attacks. Unfortunately, even with the highest security precautions, data breaches happen again and again, even (or especially) with the largest providers[3].
2. People must trust their providers. Users have to rely on the operator's integrity to ensure that their biometric data is only used for its intended purpose, without being shared or exploited for other purposes. Users have limited control over their data once it is collected.

To mitigate the risks associated with centralized biometric data storage, a decentralized approach is considered the gold standard [16]. By distributing biometric data across multiple instances, each instance becomes less attractive to potential attackers. However, decentralization also introduces additional complexity to the system, as Wolpert's no free lunch theorem suggests [22]: the gain in security comes at the cost of a more complex system.

One of the complexities associated with decentralized systems is the data economy perspective, whereby several smaller organizations operate sensors to reduce the amount of data available to any single entity [19]. As a result, decentralized systems aim to support as many sensors as possible by minimizing hardware requirements.

To address these challenges, this paper proposes an efficient face pipeline architecture capable of running on embedded systems. We tested its real-world behavior by mounting three cameras in front of our office doors and running the proposed pipeline. Furthermore, we evaluated if new systems and biometric architectures should provide additional metrics, to be able to make a more informed decision while deciding between components for face recognition pipelines.

[3] A list with recent large-scale data breaches is visualized at `https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/`

Overall, this research underscores the importance of prioritizing security in biometric data management while considering the trade-offs between security and complexity in designing decentralized systems.

2 Background

Authenticating a person using biometrics requires two main steps: face detection, followed by face recognition. First, the system must accurately detect and locate the face within the image or video frame. Once the face is detected, the system can then extract the relevant facial features necessary for recognition. Recognition involves comparing these features to a database of known faces to determine the identity of the individual. Therefore, accurate detection and recognition are essential for effective and reliable biometric authentication.

The Labeled Faces in the Wild (LFW) dataset [18] is a widely-used benchmark dataset in the field of computer vision to assess the performance of recognition models. This dataset has become an important standard in the field and is commonly used for performance comparisons between different models and comprises over 13,000 images of faces with variations in pose, lighting, and facial expressions. As there are multiple images of the same person, the dataset is suitable for evaluating face recognition models. We utilized the dataset for evaluating the performance of both face recognition and face detection models. To evaluate face recognition models, LFW provides a test dataset that contains 3,000 true positive and 3,000 true negative matches.

Specifically, we used a metric in which a predicted bounding box is considered successful if it has an overlap of more than 50 % with the ground truth bounding box. This metric was chosen as it is a commonly used standard for evaluating the performance of face detection models on the LFW dataset, e.g. by Yang et al. [23]. By employing this metric, we were able to quantitatively measure and compare the accuracy of different face detection models.

2.1 Face detection

Face detection is a computer vision task that identifies the presence and location of human faces in digital images and video frames. With the increasing demand for facial recognition technology, a wide variety of face detection models have been developed. Each model has certain advantages over their competitors. Some focus on finding tiny faces [12], occluded faces [11], or using multiple camera angles [7].

In order to quantify the quality of networks and being able to compare different models, they are evaluated on publicly available datasets. There is a focus on accuracy: Wider Face [23] shows precision-recall curve, LFW [18] shows the ROC-curve and the corresponding area under curve, VGGFace2 [2] shows false(-positive)-acceptance-rates and rank-accuracies, UMD Faces [1] shows the normalized mean error.

In this section, we will provide a brief overview of four popular choices of face detection networks.

Retinaface is based on a single-shot detector framework and uses a fully convolutional neural network (FCN) to detect faces in images. The architecture of Retinaface consists of three main components: a backbone network, a multiscale feature pyramid network, and three task-specific heads.

The backbone network is responsible for feature extraction and is typically a pre-trained ResNet or MobileNet. The feature pyramid network then takes the feature maps generated by the backbone network and produces a set of multiscale feature maps. Finally, the task-specific heads, which consist of a classification head, a regression head, and a landmark head, are applied to each of the feature maps to predict the presence of a face, its bounding box, and its facial landmarks.

ULFGFD is specifically designed to be lightweight and suitable for deployment on edge computing devices. The small size, just over 1 MB, stands out in particular. The network is based on a single-shot detector (SSD) architecture and consists of a backbone network and a prediction network. The backbone network is a lightweight MobileNetV2 architecture that is used to extract features from input images. The prediction network consists of a set of convolutional layers that are used to predict the bounding boxes and confidence scores of faces in the input images.

ULFGFD also uses a feature pyramid network (FPN) to detect faces at different scales. The FPN consists of a set of convolutional layers that are used to generate feature maps at different resolutions. These feature maps are then used to predict the bounding boxes and confidence scores of faces at different scales.

YuNet is a deep neural network architecture designed for efficient face detection and recognition in real-world scenarios [8].

YuNet is composed of three main components: a lightweight backbone network, a feature pyramid network (FPN), and a detection head. The backbone network is based on MobileNetV2, a popular architecture known for its efficiency and low computational cost.

The detection head of YuNet is responsible for predicting the locations of faces in the input image. It consists of a set of convolutional layers followed by two parallel branches. One branch performs classification to determine whether a given region of the image contains a face or not, while the other branch performs regression to predict the bounding box coordinates of the face.

Haarcascade is a widely used computer vision algorithm for face detection, having been introduced by Viola and Jones as early as 2001 [21]. Despite being around for over two decades, Haarcascade remains a popular choice for face detection in various applications due to its simplicity, efficiency, and effectiveness.

The Haarcascade algorithm works by using a series of classifiers to detect faces within an image. Each classifier is composed of a set of weak learners, which are typically decision trees that evaluate simple features such as edges and corners. These features are calculated on a sliding window that moves across the image, with the goal of detecting faces at different scales and orientations.

One limitation of Haarcascade is that it can be sensitive to changes in lighting conditions and occlusion, which can result in false positives or missed detections.

2.2 Face recognition

Face recognition is the process of identifying an individual based on their distinctive facial features. In recent years, the accuracy, reliability, and efficiency of this process have increased significantly due to advancements in deep learning algorithms and the availability of large datasets.

The majority of state-of-the-art (SOTA) algorithms requires a pre-processed RGB image as input, which is then used to create a high-dimensional vector that represents the individual's facial features. To ensure that the images are properly pre-processed, it is necessary to use landmarks from the individual's face. Typically, these landmarks consist of the eye, nose, and mouth points, which are used to ensure that the image is properly aligned and scaled.

For our pipeline, we tested a single instance of a state-of-the-art face recognition model. This decision was based on two factors: the model's negligible inference-time compared to face detection and its near-perfect accuracy. Therefore, our primary focus was not on selecting the best-performing face recognition model, but on optimizing the pipeline's overall efficiency.

Arcface Arcface [4] is a SOTA face recognition method that uses a neural network-based approach to extract discriminative features from faces. The technical details of Arcface include a modified ResNet architecture with a large embedding size, a novel angular softmax loss function, and specific optimization techniques. The ResNet architecture consists of several convolutional layers, which extract features from the input face image. The embedding size of Arcface is a 512 dimensional floating point array.

Arcface is trained using a custom loss function (*Arcface loss*), based on cosine similarity between features because it enforces more inter-class discrepancy. Different distance functions are used for comparing two embeddings in practice. Typically, the L2 loss function is used as distance measurement. However, in certain applications different distance functions are preferable. For example, a zero knowledge proof might need an inner product for efficient calculation, therefore cosine distance might be the preferred function.

3 State-of-the-art face recognition pipeline

The typical SOTA setup for image-based face recognition consists of the following components:

$$\text{Camera} \rightarrow \text{Detection} \rightarrow \text{Recognition} \rightarrow \text{Comparison}$$

The inference time is heavily influenced by the size of the retrieved camera image. For this paper, we assume that the camera produces 4K images. For the default pipeline, we use Retinaface [3] as face detection model because it has SOTA accuracy on many datasets and returns face landmarks which are needed for face recognition. Similarly, Arcface [4] is used because it gives SOTA accuracy.

Two embeddings are compared using a distance function. There has been no study on the impact of using different distance functions during inference. Therefore, we evaluated the impact of three popular distance metrics used with Arcface, namely absolute, L2, and cosine distance. We calculated the embeddings of the 6,000 test image-pairs from the LFW dataset and followed their protocol to verify the accuracy of Arcface using different distance metrics. Our findings reveal that the choice of distance metric does not have a significant effect on the analysis outcome. The precision-recall plot presented in Fig. 1 indicates only minor differences, which are only visible if we zoom in on the plot. The inference time is not affected significantly as well, our benchmark indicates roughly 1 µs computation time for all three variants (L2: $1.0939\mu s \pm 3.9ns$, Cos-Dist: $1.1549\mu s \pm 20.9ns$, absolute: $1.0956\mu s \pm 8.6ns$).

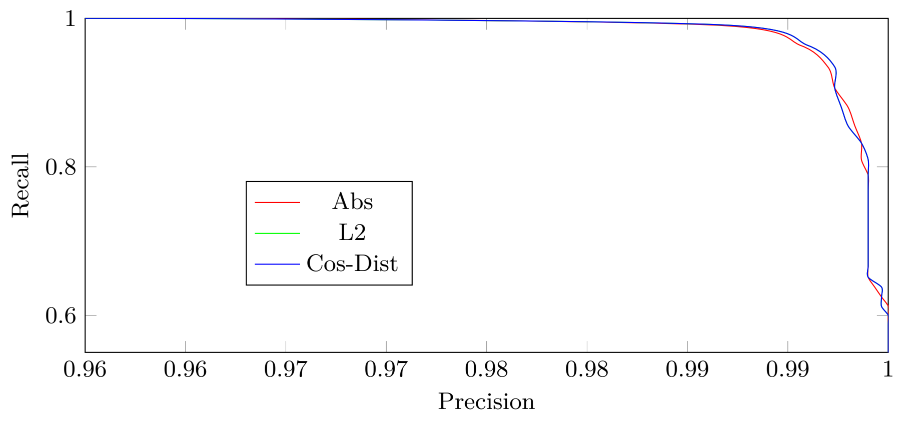


Fig. 1: Different distance functions for Arcface. Notice the magnified scale; plotting the whole spectrum (0-1) would yield no discernible distinction. The green line is not visible, as using L2 and COS distance functions yields an identical precision-recall curve. The Area Under Curve (AUC) is not significantly different either: $AUC_{L2} = 0.99884653$, $AUC_{ABS} = 0.9988512$, $AUC_{COS} = 0.99884653$.

Due to popular use, the L2 norm is used for the rest of this paper. This gives us the following architecture for our default pipeline:

$$\underbrace{\text{Camera}}_{\text{4k images}} \rightarrow \underbrace{\text{Detection}}_{\text{Retinaface}} \rightarrow \underbrace{\text{Recognition}}_{\text{Arcface}} \rightarrow \underbrace{\text{Comparison}}_{\text{L2 Norm}}$$

3.1 Performance

In order to establish a baseline for the performance, we implemented the pipeline in Rust using Tensorflow Lite (Retinaface and ULFGFD) and OpenCV (YuNet and Haarcascade). Due to popularity, all benchmarks are executed on a Jetson Nano[4], with an NVIDIA Maxwell GPU and a Quad-core ARM Cortex-A57 MPCore CPU.

There are two distinct performance metrics:

1. With respect to **time**: We established benchmarks using Criterion [10]. To ensure statistical significance and reliability, each component underwent 100 iterations, and the reported time is based on the median of these runs. The variance is less than 4.8% of the value for all components. It is noteworthy that the times reported are calculated per image, with Retinaface requiring a total of 91 seconds for inference.

$$\underbrace{\text{Camera (4k)}}_{0.02s} \rightarrow \underbrace{\text{Retinaface}}_{\text{91s}} \rightarrow \underbrace{\text{Arcface}}_{\text{0.071s}} \rightarrow \underbrace{\text{Comparison}}_{\text{0.000028s}}$$

   Retrieving the 4k image from the camera is possible at that frequency, because hardware acceleration and MJPG compression are used.
2. With respect to **accuracy**: We use the 6,000 face comparisons proposed by LFW [18] and run the face recognition pipeline on it. If multiple faces are found, the one closest to the center is used, as the LFW images are pre-processed in that way. Retinaface manages to find all faces. As LFW primarily features single-person portraits, this accuracy was expected. Arcface uses the best threshold on that dataset to decide if the two faces are from the same person.

$$\text{Camera (4k)} \rightarrow \underbrace{\text{Retinaface}}_{100\%} \rightarrow \underbrace{\text{Arcface}}_{99.3\%} \rightarrow \text{Comparison}$$

3.2 Improvements

Time-performance (1.5 minutes per 4K image) is arguably too slow for real-time performance. Most time (99.9%) is spent on Retinaface. There are two options to reduce the inference time:

1. Reduce the input dimension, which yields the following time-performance:
   - 4k (3840x2160px): 91.24 s
   - Full HD (1920x1080px): 11.52 s
   - HD (1280x720px): 5.13 s
   - SD (640x480px): 1.72 s

[4] `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/`

How is accuracy-performance affected if input dimension is reduced? The theoretical lower-limit is detecting people of size 16 px x 16 px, as this is the smallest anchor used by Retinaface. We tested if such small faces are detected in practice. Starting with LFW's image size of 250 px x 250 px, we run our face recognition pipeline over all (test) images to determine the detected face size. Subsequently, the images were scaled down by 50 pixels, and the experiment was repeated until the image size was 50 x 50 px. The resulting face sizes were recorded. Fig. 2 illustrates the widths and heights in pixels for detected faces. It is apparent that the smallest anchors are not only used for sub-features (for use in higher levels of the FPN [13]), but also to directly detect faces. Interestingly, the smallest detected face has a dimension of 10 px x 14 px. This is smaller than the smallest anchor (16 px x 16 px) and possible because the network refines its predicted bounding box in later stages.

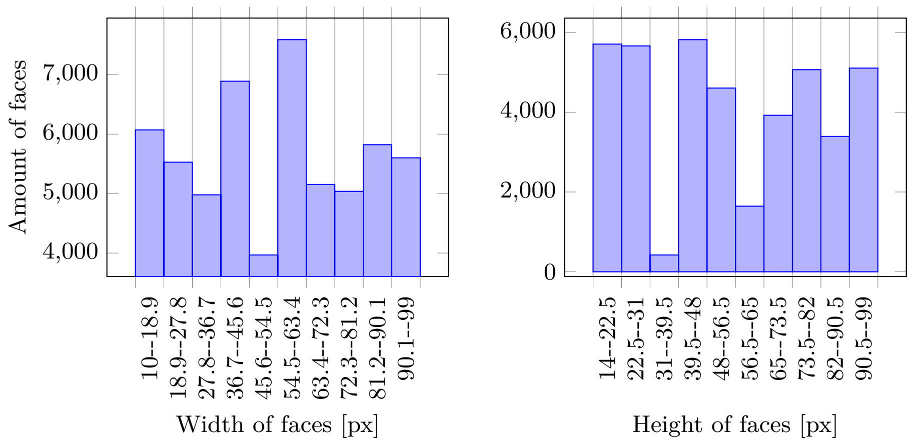
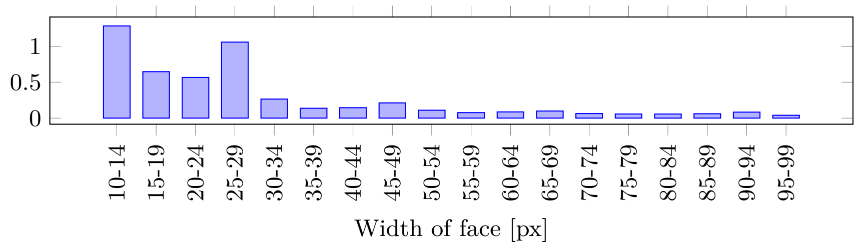


Fig. 2: The sizes of detected faces using Retinaface. Sizes larger than 99 pixels are not displayed as our focus was on identifying the smallest detectable faces.

Despite the successful detection of faces, there is no guarantee that the image has enough information for face recognition to recognize a person. Therefore, we created another experiment by performing the same shrinking of the images as before. An embedding of the scaled down version of the image is (L2) compared to the embedding of the full image. The results are plotted in Fig. 3.

As anticipated, our analysis reveals a distinct threshold at approximately 40 x 30 pixels, beyond which facial recognition accuracy is substantially diminished.

Even though an image of SD quality still has an inference time of 1.7 seconds, it skips 99.69% of potential input data (25,600 vs 8,294,400 pixel).



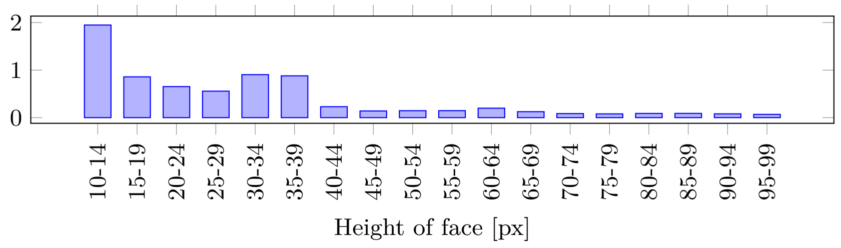


Fig. 3: L2 distance to reference embedding (full size face) using different face sizes (smaller is better).

We can calculate the real-world impact of this dimension reduction. For this calculation, we need a few hardware assumptions.

- **Face dimensions** Being able to detect an object depends on its size. Since we want to detect a face, we have to assume the dimensions. The US Department of Defense measured the width (bitragion breadth) and height (menton-crinion length) of the face to be between 12--15 cm and 15--21 cm, respectively [5]. We want to find the lower limit of face recognition pipelines possibilities. Therefore, we use the upper end of the face dimension scale:

$$\text{face}_{\text{width}} = 0.15\text{ m}, \text{face}_{\text{height}} = 0.21\text{ m}$$

- **Camera** For the camera, we assume typical 70 mm focal length with a full frame 35 mm sensor:

$$\text{camera}_{\text{focallength}} = 0.07\text{ m}$$

$$\text{camera}_{\text{imagewidth}} = 35\text{ mm}, \text{camera}_{\text{imageheight}} = 24\text{ mm}$$

Figure 3 demonstrates that facial recognition can reliably commence at sizes as small as 40 x 30 pixels.

$$\text{object}_{\text{width}} = 30\text{ px}, \text{object}_{\text{height}} = 40\text{ px}$$

We can now calculate the maximum distance in millimeters of a person with respect to the camera, such that the face is still recognizable:

$$\text{distance} = \frac{\text{camera}_{\text{focallength}} \times \text{pixel}_{\text{width/height}} \times \text{face}_{\text{width/height}}}{\text{object}_{\text{width/height}} \times \text{camera}_{\text{imagewidth/height}}}$$

If we use pixel$_{width/height}$ of 640 and 480 respectively, we can detect faces up to 6.4m. With a pixel$_{width/height}$ of 3840 and 2880 this distance increases to 38.4m.

2. Use a different, more lightweight model. Due to the use of a large backbone network (ResNet [9]) and its computationally heavy use of feature pyramid networks [13], the inference time of Retinaface is slow. There are lighter networks with fewer parameters, such as ULFGFD. One major deficiency of fast face detection algorithms is their tendency to produce false positives. Retinaface, on the other hand, has been shown to have a very low false positive rate, making it a more reliable option for these types of applications. As face recognition expects a pre-processed image and this pre-processing depends on the location of landmarks, it is not possible to calculate face recognition accuracy with ULFGFD.

4 Inference-time/accuracy tradeoff

The accuracy of face detection models has been extensively studied and reported in modern research (as demonstrated by the reported metrics described in Section 2.1). However, an often overlooked aspect in the evaluation of these models is their inference time. This information is important, as a slow inference time can lead to delays and long queues, compromising the effectiveness of the system (cf. Section 3.1). Inference time can also impact the scalability and cost-effectiveness of a face detection system, as a slow model may require more powerful hardware or computing resources to achieve the desired performance. Furthermore, inference time is especially important when considering the deployment of face detection models on embedded hardware. These devices often have limited computing resources and require models that can perform in real-time. Therefore, evaluating face detection models based on their inference time is essential for ensuring that they can be deployed effectively on embedded hardware and meet the performance requirements of real-world applications. Despite its critical importance in real-world deployment scenarios, none of the existing datasets currently available comprehensively address this aspect of performance evaluation. As a result, there is a significant gap in our understanding of the practical implications of face detection model performance in real-world settings.

This paper evaluates SOTA face detection models with respect to these metrics. We assessed the performance and accuracy of four face detection models, Retinaface [3], ULFGFD [14], YuNet [8], and Haarcascade [21]. In this paper, Figure 4 illustrates the space of performance-accuracy for current models. It is important to note that only the networks situated at the border of the performance-accuracy spectrum are relevant, and their selection depends on the specific application requirements. Different applications may require different points on the performance-accuracy spectrum, and our study provides insights into the trade-offs involved in selecting an appropriate face detection model for a given application. Our results, depicted in Fig. 4, clearly show that Haarcascade has a detection failure rate of over 50%, even for the relatively simple portrait-like datasets such as the LFW dataset. Retinaface achieves high accuracy but requires high inference time, while ULFGFD has lower accuracy but a faster inference time.

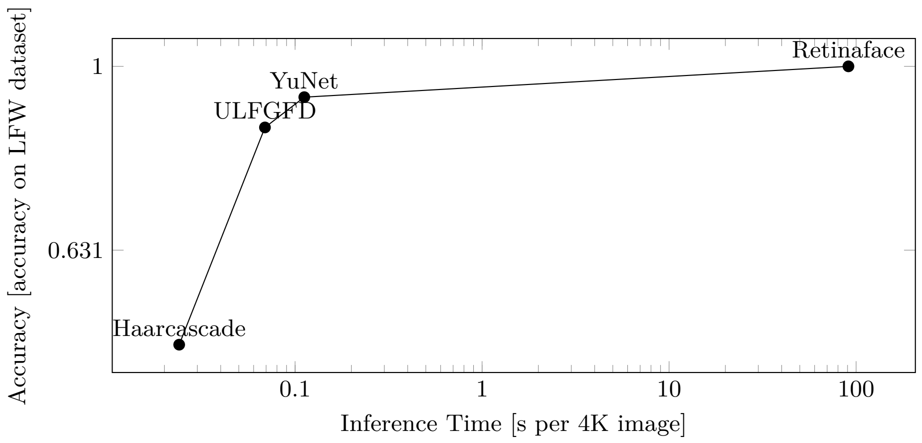


Fig. 4: The figure illustrates the trade-off between inference-time and accuracy for various face detection networks. The x-axis represents the inference time, while the y-axis represents the accuracy of the networks. The solid line in the figure represents the Pareto frontier, which is the optimal trade-off between accuracy and inference time.

5 Fast and accurate face recognition pipeline

In order to optimize face detection for both speed and accuracy, we propose an approach that combines two algorithms with distinct characteristics in the inference/time spectrum to harvest the strengths of each. A fast algorithm is used as a proposal generator to quickly create possible face detections. We prioritize minimizing false negatives in the proposal generator, as false positives can be verified by the subsequent algorithm. While our analysis shows Haarcascade to be the fastest method, it misses more than half of the faces even in the easy LFW dataset. Therefore, we use ULFGFD as our algorithm for generating proposals. These proposals are then confirmed and augmented with face landmarks by a more accurate algorithm, Retinaface. This yields the following pipeline:

$$\text{Camera (4k)} \rightarrow \text{ULFGFD} \rightarrow \text{Retinaface} \rightarrow \text{Arcface} \rightarrow \text{Comparison}$$

As discussed in Section 3, face recognition requires face dimensions of at least 30 px x 40 px. To achieve higher accuracy, we recommend using face images with dimensions of 50 px x 65 px or larger. Our experimental results indicate that performance degrades when face images are smaller than this threshold.

As the pipeline should take bounding box errors of ULFGFD into account, we performed a systematic search on a grid of possible dimensions and performed additional benchmarks on Retinaface with respect to inference time:

- 150 px x 150 px: 0.169 s
- 125 px x 125 px: 0.093 s
- 100 px x 100 px: 0.052 s
- 75 px x 75 px: 0.038 s
- 50 px x 50 px: 0.018 s

Subsequently, we constructed the complete pipeline and evaluated the accuracy of each individual component as follows:

$$\text{Camera (4k)} \rightarrow \underbrace{\text{ULFGFD}_{\text{th}=0.05}}_{96.45\%} \rightarrow \begin{array}{c} \underbrace{\text{Retinaface}_{50\text{x}50}}_{73.2\%} \\ \underbrace{\text{Retinaface}_{75\text{x}75}}_{86.1\%} \\ \underbrace{\text{Retinaface}_{100\text{x}100}}_{98.3\%} \\ \underbrace{\text{Retinaface}_{125\text{x}125}}_{98.9\%} \\ \underbrace{\text{Retinaface}_{150\text{x}150}}_{99.7\%} \end{array} \rightarrow \begin{array}{c} \underbrace{\text{Arcface}}_{92.3\%} \\ \underbrace{\text{Arcface}}_{95.3\%} \\ \underbrace{\text{Arcface}}_{97.3\%} \\ \underbrace{\text{Arcface}}_{97.7\%} \\ \underbrace{\text{Arcface}}_{98.3\%} \end{array} \rightarrow \text{Comparison}$$

A size of 100 px x 100 px for the Retinaface input seems to be a good tradeoff between time and accuracy performance. With this, the full pipeline runs on ~ 4.7 FPS on a Jetson Nano and achieves an overall accuracy of 92.3% $(0.9645 * 0.983 * 0.973)$ on the LFW dataset.

Next, we can calculate both the inference time and accuracy of the full pipeline with these three combinations of networks and compare it to existing algorithms. Fig. 5 clearly demonstrates that by integrating multiple different networks, the trade-off border in the inference-time/accuracy spectrum is increased and a better balance between these two metrics is achieved. This indicates the effectiveness of our approach in improving face detection performance.

Notably, the selection of suboptimal parameters, as observed in Fast50 and Fast75, can lead to an unexpected outcome where the desired effect is inverted. Specifically, this may result in a decrease in accuracy despite a slower inference time. Therefore, it is crucial to carefully select appropriate parameters by utilizing techniques such as analyzing the inference-time/accuracy plot to ensure the desired performance outcome is achieved.

6 Future work

To improve the efficiency of face detection systems, future work can analyze the impact on inference time and accuracy if positional information of previously

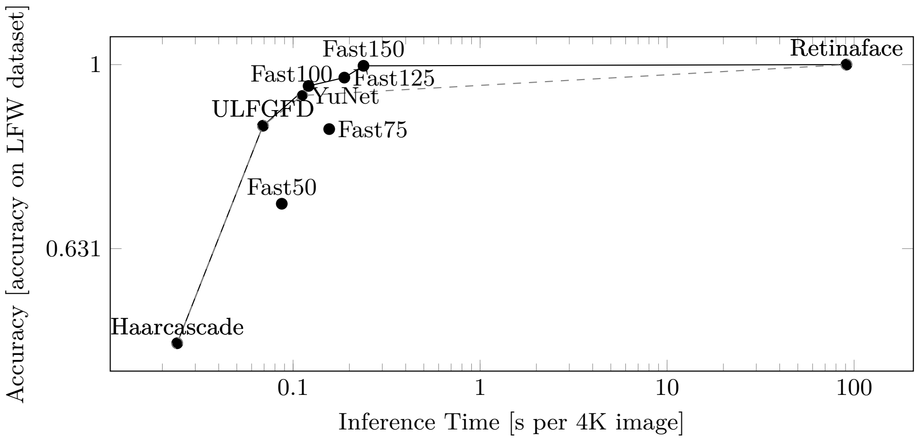


Fig. 5: This plot adds our proposed models to the initial plot of Fig. 4, which is represented by the dashed line. Our proposed Fast100, Fast 125, and Fast150 networks increase the Pareto front in the *inference-time/accuracy* spectrum, as visualized in the solid line. Fast50 and Fast75 do not increase the border, as they are slower and have less accuracy than ULFGFD.

detected faces are utilized to narrow down the search area in subsequent readings, instead of scanning the entire image each time. This strategy can significantly reduce the computational cost and time required for the detection process.

Furthermore, to enhance the system's resilience and prevent it from being deceived by fake representations, such as videos or photos, a liveness detection mechanism can be incorporated. This mechanism can distinguish between the presence of an actual person in front of the camera and a non-living representation, such as a still image or a pre-recorded video. By incorporating such a mechanism, our system can be more reliable and effective in real-world scenarios where security and authenticity are crucial factors.

7 Conclusion

In recent years, real-time face recognition has become increasingly popular, particularly for applications in decentralized systems and on embedded hardware. However, this popularity has come with several challenges, including the trade-off between accuracy and inference-time on constrained hardware resources. Achieving higher accuracy is desirable, but it often comes at the cost of longer inference-time, which is particularly problematic for embedded devices with limited processing power.

To address this challenge, we conducted a comparative study to investigate the effect of using different face recognition distance functions. Future datasets and models should include inference time as a metric for performance evaluation.

This will allow researchers to better understand the trade-offs between accuracy and efficiency in real-world deployment scenarios, and enable the development of more effective and efficient face detection models that can be deployed in real-world applications. By including inference time as a metric, the practical relevance of face detection research is improved and models can be optimized for real-world deployment. We also introduced a novel inference-time/accuracy plot that enables the comparison of different face recognition models. Our analysis showed that different models have different strengths and weaknesses, and every application must strike a balance between inference-time and accuracy, depending on its focus.

To achieve optimal performance across the spectrum, we proposed a combination of multiple models with distinct characteristics. This approach allows the system to address the weaknesses of individual models and optimize performance based on the specific needs of the application. We demonstrated the practicality of our proposed approach by developing a multimodel face recognition pipeline. We utilized two face detection models positioned at either end of the inference-time/accuracy spectrum to achieve superior overall accuracy, reliability, and speed. Specifically, we employed the more accurate model when necessary and the faster model for generating fast proposals, thereby improving the trade-off between inference-time and accuracy.

Overall, our proposed pipeline can serve as a guideline for developing real-time face recognition systems on embedded devices. By striking an optimal balance between the performance of different models, we can improve the overall accuracy, reliability, and speed of such systems.

References

1. Bansal, A., Nanduri, A., Castillo, C.D., Ranjan, R., Chellappa, R.: Umdfaces: An annotated face dataset for training deep networks. In: IEEE International Joint Conference on Biometrics (IJCB). pp. 464--473. IEEE (2017). https://doi.org/10.1109/BTAS.2017.8272731
2. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: VGGFace2: A Dataset for Recognising Faces across Pose and Age. In: 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). pp. 67--74. IEEE (2018). https://doi.org/10.1109/FG.2018.00020
3. Deng, J., Guo, J., Ververas, E., Kotsia, I., Zafeiriou, S.: Retinaface: Single-shot Multi-Level Face Localisation in the Wild. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5202--5211 (2020). https://doi.org/10.1109/CVPR42600.2020.00525
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4685--4694 (2019). https://doi.org/10.1109/CVPR.2019.00482
5. Department of Defense: Human engineering design data digest (2000), accessed April 3, 2023. https://apps.dtic.mil/sti/pdfs/ADA467401.pdf
6. europa.eu: Entry/exit system (ees) (2023), accessed April 3, 2023. https://home-affairs.ec.europa.eu/policies/schengen-borders-and-visa/smart-borders/entry-exit-system_en

7. Farfade, S.S., Saberian, M.J., Li, L.J.: Multi-view face detection using deep convolutional neural networks. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval. pp. 643--650 (2015). https://doi.org/10.48550/arXiv.1502.02766
8. Feng, Y., Yu, S., Peng, H., Li, Y.R., Zhang, J.: Detect Faces Efficiently: A Survey and Evaluations. IEEE Transactions on Biometrics, Behavior, and Identity Science **4**(1), 1--18 (2022). https://doi.org/10.1109/TBIOM.2021.3120412
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770--778 (2016). https://doi.org/10.1109/CVPR.2016.90
10. Heisler, B.: criterion.rs: Statistics-driven benchmarking library for Rust. `https://github.com/bheisler/criterion.rs` (2014)
11. Kumar, A., Kumar, M., Kaur, A.: Face detection in still images under occlusion and non-uniform illumination. Multimedia Tools and Applications **80**, 14565--14590 (2021). https://doi.org/10.1007/s11042-020-10457-9
12. Li, Z., Tang, X., Han, J., Liu, J., He, R.: PyramidBox++: High Performance Detector for Finding Tiny Face (2019). https://doi.org/10.1904/arXiv.1904.00386
13. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature Pyramid Networks for Object Detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 936--944 (2017). https://doi.org/10.1109/CVPR.2017.106
14. Linzaer: 1MB lightweight face detection model. `https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB` (2019)
15. Liu, C.: Multiple social credit systems in China. Economic Sociology: The European Electronic Newsletter **21**(1), 22--32 (2019)
16. Mayrhofer, R., Roland, M., Höller, T.: Poster: Towards an Architecture for Private Digital Authentication in the Physical World. In: Network and Distributed System Security Symposium (NDSS Symposium 2020), Posters (Feb 2020)
17. mos.ru: The Face Pay system for fare payment was launched at all metro stations (2023), accessed February 2, 2023. `https://www.mos.ru/news/item/97579073/`
18. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: Proceedings of the British Machine Vision Conference (BMVC). pp. 41.1--41.12. BMVA Press (September 2015). https://doi.org/10.5244/C.29.41
19. Roland, M., Höller, T., Mayrhofer, R.: Digitale Identitäten in der physischen Welt: Eine Abwägung von Privatsphäreschutz und Praktikabilität. HMD Praxis der Wirtschaftsinformatik **60**(2), 283--307 (Mar 2023). https://doi.org/10.1365/s40702-023-00949-1
20. uidai.gov.in: Unique Identification Authority of India (2023), accessed February 2, 2023. `https://uidai.gov.in/en/`
21. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. vol. 1, pp. I--I (2001). https://doi.org/10.1109/CVPR.2001.990517
22. Wolpert, D.H.: The Lack of A Priori Distinctions Between Learning Algorithms. Neural Computation **8**(7), 1341--1390 (1996). https://doi.org/10.1162/neco.1996.8.7.1341
23. Yang, S., Luo, P., Loy, C.C., Tang, X.: WIDER FACE: A Face Detection Benchmark. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5525--5533 (2016). https://doi.org/10.1109/CVPR.2016.596