JST: Smart Systems and Devices
Volume 34, Issue 1, January 2024, 001-008

An Efficient Face Recognition System Based on Edge Processing Using GPUs

Ha Xuan Nguyen[1*], Dong Nhu Hoang[2], Hung Trung Nguyen[2], Hai Ngo Minh[2], Tuan Minh Dang[2,3,4]

[1]Hanoi University of Science and Technology, Ha Noi, Vietnam
[2]CMC Applied Technology Institute, CMC Corporation, Ha Noi, Vietnam
[3]CMC University, CMC Corporation, Ha Noi, Vietnam
[4]Posts and Telecommunication Institute of Technology, Ha Noi, Vietnam
[*]Corresponding author email: ha.nguyenxuan@hust.edu.vn

Abstract

In this work, an efficient and accurate face recognition system based on edge processing using GPUs was completely developed. A complete pipeline that contains a sequence of processing steps, including pre-processing, face feature extraction, and matching, is proposed. For processing steps, lightweight deep neural models were developed and optimized so that they could be computationally accelerated on an embedded hardware of Nvidia's Jetson Nano. Besides the core processing pipeline, a database, as well as a user application server were also developed to fully meet the requirements of readily commercialized applications. The experimental evaluation results show that our system has a very high accuracy based on the BLUFR benchmark, with a precision of 98.642%. Also, the system is very computationally efficient, as the computing time to recognize an ID in a dataset of 1171IDs with 10141 images on the Jetson Nano is only 165ms. For the critical case, the system can process 4 camera streams and simultaneously recognize a maximum of 40 IDs within a computing time of 458ms for each ID. With its high-speed and accuracy characteristics, the developed system has a high potential for practical applications.

Keywords: Face recognition, deep learning, GPUs, edge processing.

1. Introduction

The rapid development and advancement of deep learning and computing hardware have shown many advantages for the image processing problem. The face recognition issue has received much attention in recent years due to its very high potential for practical applications in access/check-in/check-out control systems, public security surveillance systems, and electronic commercial transactions [1]. As illustrated in Fig. 1, face recognition, like any other image processing problem, has three main modules: i) the pre-processing module, which includes processes such as face detection, anti-spoofing, alignment, and quality checking; ii) the face feature extraction module, which employs a deep neural network and typically yields face-feature vectors (FFVs); and iii) the face feature matching module, which calculates cosine distances between FFVs to obtain the final identification of a face. Each module requires much of research effort to improve the accuracy and performance of the system [2-4]. This processing pipeline requires a lot of computational effort, and as a result, high-performance computing hardware is needed. Thus, the need for developing and optimizing lightweight models that satisfy both high accuracy and computational efficiency is a current research trend [1].

For many applications of face recognition, systems have plenty of cameras, which generate a large amount of data to be processed. If all the generated data is sent to the central server to be processed, the system will confront a problem of very high throughput and bandwidth. This will make the system very difficult to deploy, even for real-time applications. Thus, the solution to this problem is the deployment of the hybrid edge-central computing architecture.

There have been many efforts to deploy deep face recognition techniques on embedded hardware using GPUs as edge computing [5-9]. The use of GPUs at the edge can help accelerate the computation of the processing pipeline. It has been demonstrated that the use of GPUs can significantly accelerate computation. However, due to the limited hardware resources of embedded hardware, the processing pipeline as well as deep neural network models for face recognition should continue to be optimized in such a way that they are lightweight and have high accuracy.

ISSN: 2734-9373
https://doi.org/10.51316/jst.171.ssad.2024.34.1.1
Received: February 15, 2023; accepted: September 27, 2023

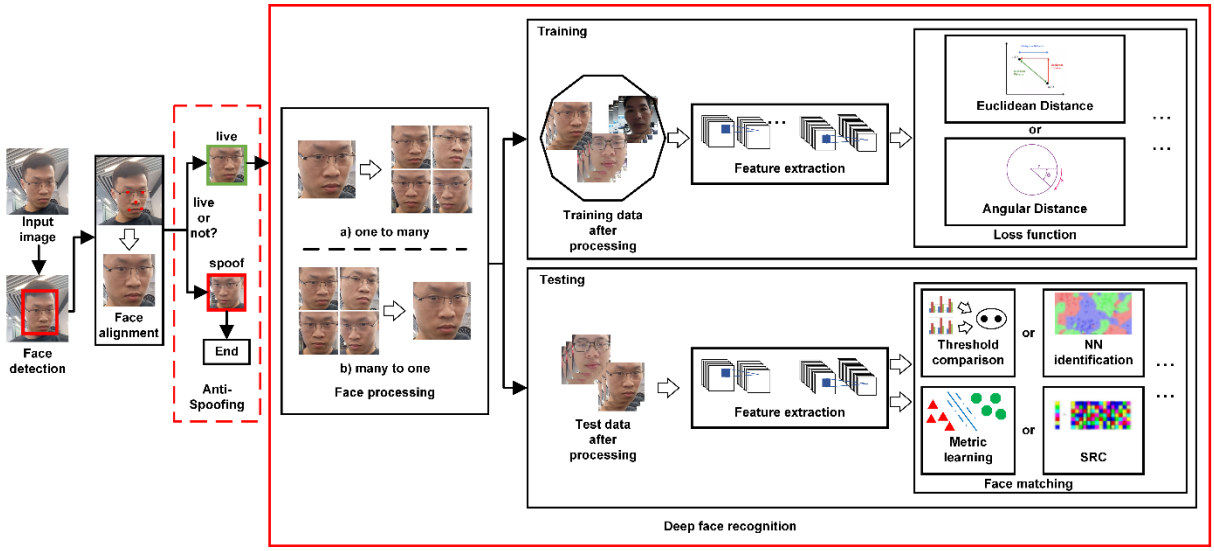

Fig. 1. General processing steps of the whole face recognition system Jetson Nano

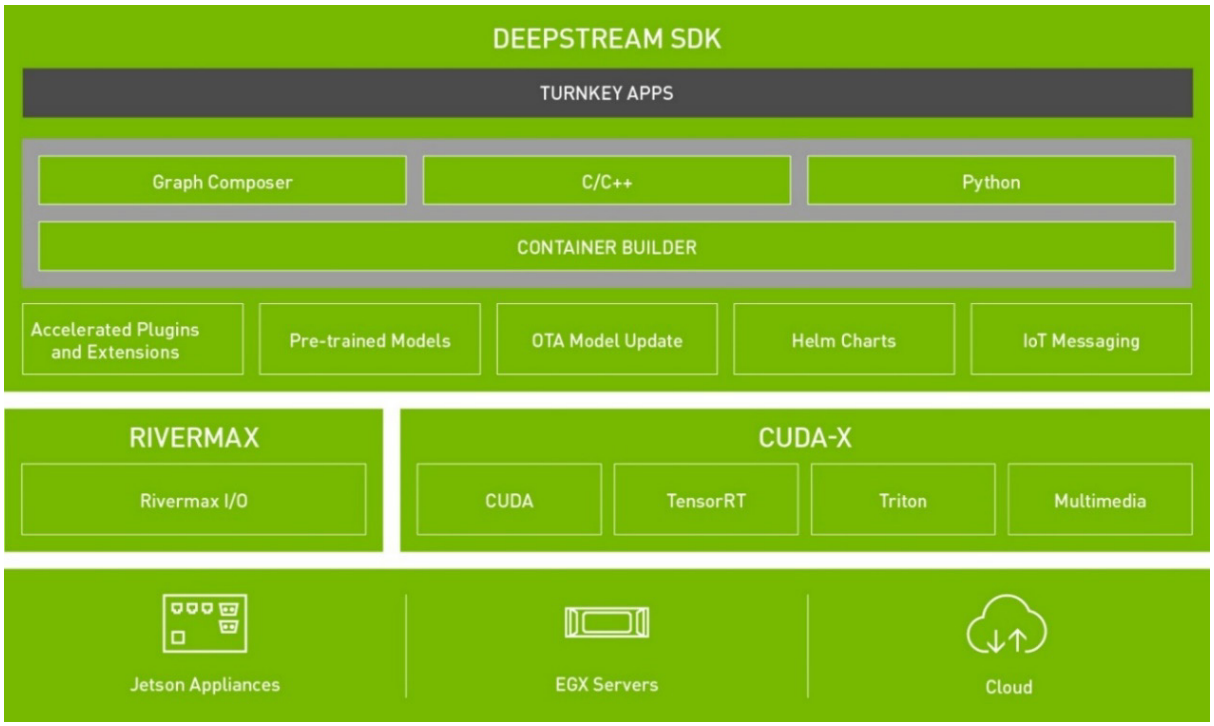

Fig. 2. A whole description of Nvidia's DeepStream platform

A key goal has been set that the system should be optimized so that the lightweight embedded hardware can process as many camera streams as possible. In conventional methods, a processing pipeline using the so-called Gstreamer [10] is often used. Each camera stream is attached to a Gstreamer-based pipeline, which contains a set of deep learning models. This approach has the disadvantage that the utilization of hardware for deep learning models is repeated through all camera streams. This limits the number of camera streams that can be processed on lightweight embedded hardware. Fortunately, Nvidia has released the DeepStream SDK [11], which allows you to mux multiple camera streams into a single processing pipeline. This new approach is quite impressive and thus should be further optimized and evaluated.

DeepStream SDK from Nvidia is a complete streaming analytics toolkit based on GStreamer for AI-based multi-sensor processing, video, audio, and image understanding. DeepStream's multi-platform support gives us a faster, easier way to develop vision AI applications and services. We can even deploy them on-premises, on the edge, and in the cloud. Fig. 2 depicts a description of the entire platform. It is ideal for vision AI developers, software partners, startups,

and OEMs who are developing IVA apps and services. Stream processing pipelines incorporating neural networks and other complex processing tasks such as tracking, video encoding/decoding, and video rendering can be created by developers. DeepStream pipelines enable real-time video, image, and sensor data analytics. This platform provides extremely powerful tools with numerous advantages, including:

a) DeepStream SDK is powerful and flexible, making it suitable for a wide range of use-cases across a wide range of industries.

b) Multiple Programming Options: The platform uses the simple and intuitive UI of C/C++, Python, or Graph Composer, you can create powerful vision AI applications.

c) Real-Time Insights: The platform can gain an understanding of rich, multimodal real-time sensor data at the edge.

d) Managed AI Services: The platform can deploy AI services in Kubernetes-managed cloud native containers.

e) Lower total cost of ownership: The platform increases stream density by training, adapting, and optimizing models with TAO toolkit and deploying models with DeepStream.

In this work, an efficient and accurate face recognition system based on GPU-based embedded hardware is developed. The system has a complete processing pipeline that contains pre- and post-processing algorithms as well as deep convolutional neural models. The DeepStream SDK is exploited for the video processing, which includes the decoder and the mux. The decoder of DeepStream is already optimized for hardware by Nvidia, so it is more efficient than building the Gstreamer pipeline from scratch. The mux allows for batching camera frames from several cameras into only one processing pipeline to reduce the utilization of the hardware resource. Our contribution is that, instead of using the already-available plugins of deep models for face recognition issues, we develop and optimize models ourselves. These models are developed so that they are lightweight while maintaining high accuracy characteristics. For this reason, pretrained models with the backbones of VIT-T [12], Retina Mnet [13], and MobileNetV2 [14] were fine-tuned using transfer learning techniques with a public dataset [15]. These models are then optimized and quantized so that they can be implemented and computationally accelerated on the Jetson Nano from Nvidia [16]. The accuracy of the system is evaluated via the benchmark BLUFR [17], and the computational efficiency is verified via several critical scenarios. For a complete system, a database that contains data for recognizing people and a user application server were developed.

2. System Description

The overall processing pipeline of the design system is shown in Fig. 3. There are four main modules, each of which is responsible for a specific task. The first module is the face recognition pipeline, which contains a sequence of several processing steps. Video frames captured from cameras are streamed to the processing unit via the RTSP protocol. The video streams are decoded by the NVDEC. The decoded frames are converted to the RGBA format. These frames are routed to the batching system for further processing on the Jetson Nano's hardware CPU. All decoded frames from different cameras are pushed to a meta data patch and then sent to our self-developed plug-in, which performs the following processes:

a) Attaching the tracking identification (ID) for each camera stream to be managed.

b) Resizing image frames to 640x360 and converting them to BGR format in order to satisfy the input requirement of the face detection model.

c) Performing the face detection inference to get the bounding box and landmarks of the detecting face, which are consequently used for the tracking and alignment process. Each detected face in video frames is assigned a tracking ID. Face tracking enables us to avoid duplication of a person's faces to be processed and reduce processing efforts.

d) Using face anti-spoofing to eliminate fake cases in which someone intentionally generates the face image of an interested person using images from smart phones, tablets, printed paper, or even wearing a mask [2].

e) Using the detected landmarks and the affine transformation to align the face. In sequence, the face feature vector extraction model is performed. Extracted vectors are then matched with vectors from the database to find a person's ID [3-4].

Deep neural networks used for this processing are listed in Table 1. Most of these processes are implemented on the deep learning processing unit (GPU) of the Jetson Nano. To meet the requirements of real-time applications, we can use the GPU to accelerate the computation of deep learning models by using parallel computing.

Table 1. Technical detail of the developed models.

Model	Back-bone/specification
Face anti-spoofing	Mobilenet V2 [14]
Face and landmark detection	Retina mnet [13]
Face feature extraction	VIT-T [12]
Face alignment	Affine transformation [18]

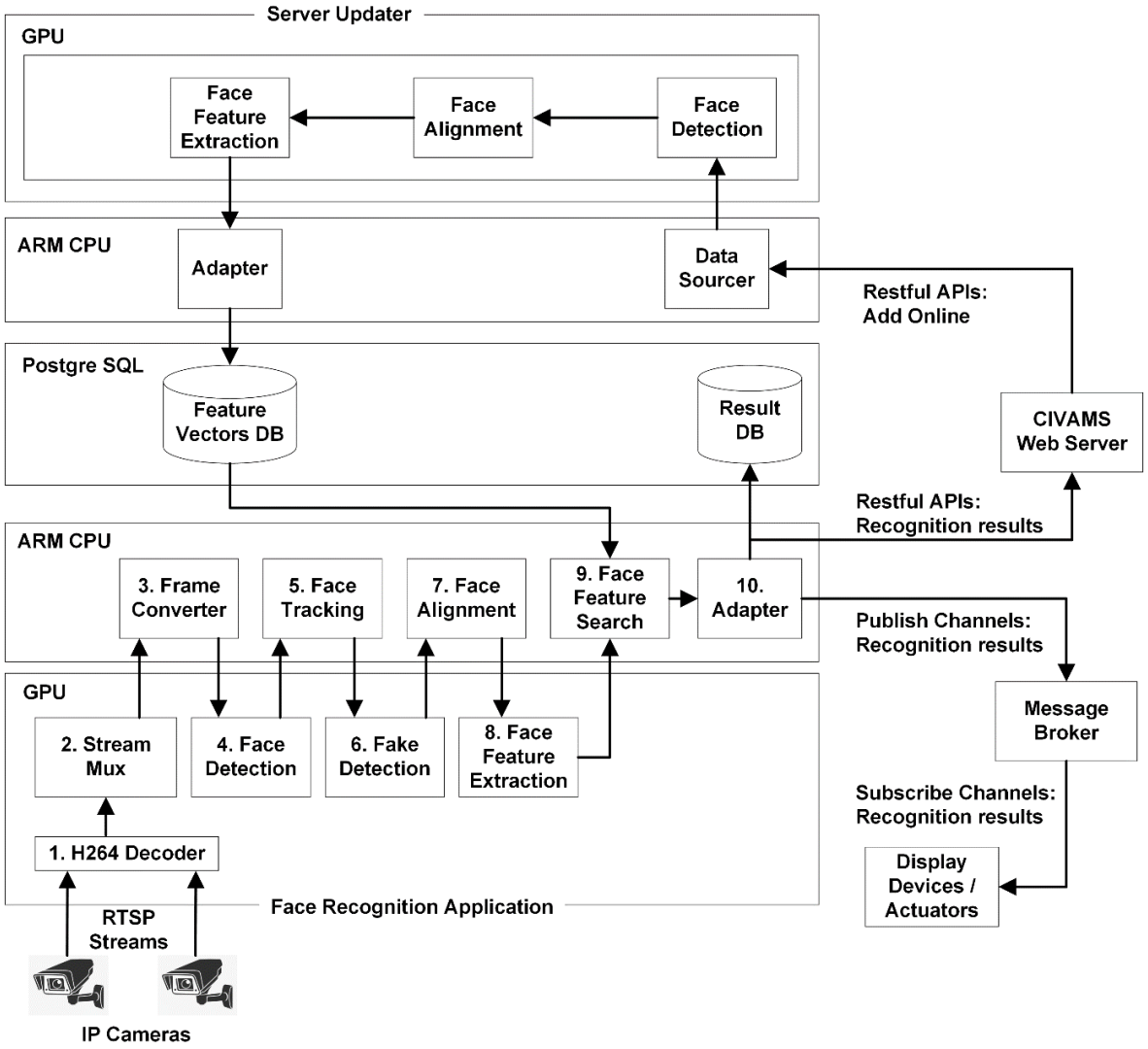

Fig. 3. Processing pipeline of the whole face recognition system based on Jetson Nano

The second module is the database using Postgre SQL, which contains the face feature vectors of the corresponding images of the person's identification. This database is created by the third module, namely the server updater. This module also has full deep learning models for face detection and face feature vector extraction. The extracted face feature vectors are then updated in the Feature Vector DB. Inputs to this module are the face images of interested persons and the corresponding record of information such as their name, age, company, and email. These inputs can be imported from the so-called CIVAMS Web Server or from the image database. When a person is required to be queried, its face feature vectors are matched with feature vectors in the database to find the best-matched vector, which has a maximum cosine similarity larger than a threshold. This process is handled by the system's ARM CPU, which is housed on a Jetson Nano chip. The fourth module is the user application, which contains the CIVAMS Web Server and CIVAMS Application Servers and other display devices and actuators for user-specific applications.

Fig. 4 describes a block diagram of the development and deployment of a deep learning model on the Jetson Nano platform. First, a deep neural network is trained based on the so-called PyTorch framework and a given dataset. After the training and testing process, a weight file “model.weight” is obtained. This model is then converted to ONNX format to get "model.onnx". These files are then quantized and combined via the tool Nvidia TensorRT from the hardware provider to obtain "model.engine", which is stored on the SD card. For the hardware platform creation, several software frameworks are installed on the Jetson Nano, including Nvidia Cuda, Python, TensorRT, and OpenCV.

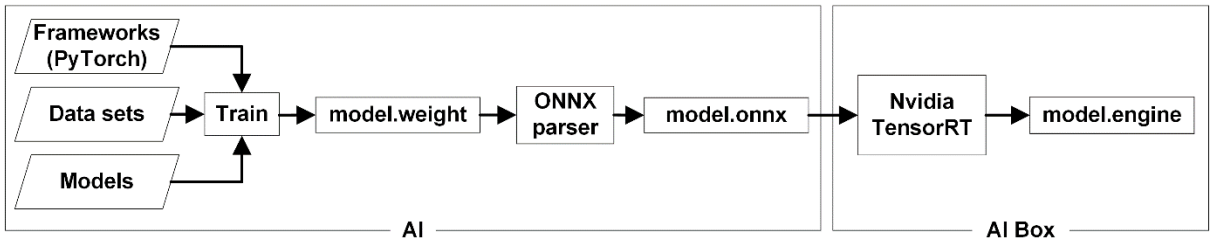

Fig. 4. Development and deployment pipeline of deep learning models on Jetson Nano

The detailed technical specifications of the Jetson Nano from Nvidia can be found in [16]. This hardware is intended for the edge processing of deep neural networks. The hardware is a completed multi-process system on chip (MP-SOC), which can be used for image processing. The hardware includes a 128-core Maxwell processor, a quad-core ARM A57 CPU running at 1.43 GHz, and 4 GB of 64-bit LPDDR4 memory (25.6 GB/s), 4K at 30 fps, 4x 1080p at 30 fps, 9x 720p at 30 fps (H.264/H.265), 4K at 60 fps, 2x 4K at 30 fps, 8x 1080p at 30 fps, 18x 720p at 30 fps (H.264/H.265); The computational benchmark for this hardware can be found on the homepage of Nvidia [16].

3. Results and Discussion

The face feature vector extraction model was developed based on the backbone of VIT-T [12] with a pretrained model. This model was fine-tuned based on a public dataset, namely the MS1M-ArcFace [15], which contains a set of 5.8M images of a total of 85k IDs. This dataset is additionally enriched by our self-collected dataset, which contains 200k images of 13k IDs. It is important that these 200k images have challenging characteristics including blur, side-view, light-mode (IR), and long-hair and glasses, as illustrated in Fig. 5. The dataset is divided into two parts: 80% for the training set and 20% for the testing set. The fine-tuned model is then evaluated using a standard benchmark, namely the BLUFR [17]. The evaluation results show that the developed model has an impressive accuracy of 98.642% and a true positive rate of 97.9%, with a false acceptance rate of only 0.0001. The obtained accuracy is high enough for practical applications. The VIT-T is a miniature version of the VIT. This will help to reduce the computational requirements of the inference process, allowing the system to be deployed on Jetson Nano's lightweight hardware.

Aside from accuracy, the system's computational efficiency is the most important factor. Evaluation results of the computational performance of the Jetson Nano are shown in Table 2. The evaluation was performed on a test dataset containing 10141 images representing 1171 IDs that are collected on our real-life deployment systems. In this evaluation, the total processing time for the overall proposed pipeline is calculated under different scenarios. The number of camera streams and the number of IDs are varied to test the computing time for each processing step, including the detection time, the recognition time, the matching time, and, as a result, the total processing time for one ID. Critical cases are considered. It is seen that for the fastest case, one ID on only one camera stream, the total processing time is only 165 ms. For the critical case, where a total of 40 IDs from 10 camera streams can be recognized in a total processing time of 458 ms for each ID, it can be concluded that the designed system has very high computational efficiency on the Jetson Nano since the overall processing pipeline was optimized and only lightweight models were developed. This shows a high potential for practical applications.

Table 2. Computational performance evaluation results.

No. of IDs	No. of streams	fps	Detection time (ms)	Recognition time (ms)	Matching time (ms)	Total time for 1 ID (ms)	Total no. of processed ID
1	1	9	111	44	10	165	1
	2	5	150	67	12	229	2
	3	4	194	82	13	289	3
	4	3	294	117	14	425	4
5	1	7	146	63	11	220	5
	2	4	206	96	14	316	10
	3	3	269	108	14	391	15
	4	2	324	120	15	459	20
10	1	6	162	69	12	243	10
	2	4	207	93	13	313	20
	3	3	268	106	14	388	30
	4	2	324	119	15	458	40

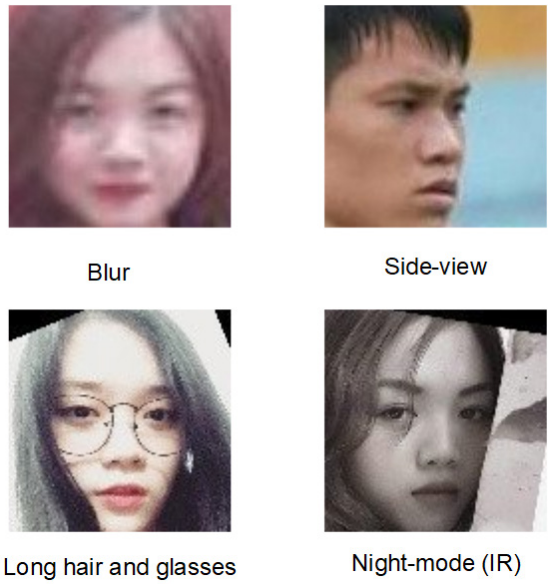

Fig. 5. Types of challenging conditions in the dataset

Several discussions are given as following:

1) Based on the test results, it is suggested that in practical applications, the hardware Jetson Nano can be efficiently used for the edge processing of four camera streams with full HD resolution. In this scenario, the processing consists only of face detection, anti-spoofing, and tracking. The detected face is then sent to the central computing server for further inference and matching. This can be called "hybrid edge-central computing."

2) If we want to perform a full processing pipeline, only two camera streams with a maximal frame rate of 5 fps should be used.

3) The computational workload can be reduced if we truncate several frames in the frame stream because the times for face feature extraction and matching are reduced. The number of frames to be truncated depends on the number of faces appearing in each camera frame. In fact, we use a queue for detected faces. At the same time, the recognition will be prioritized for the closest-appearing face images to optimize real-time recognition. The old face images are piling up in the queue. If the queue is full, it will be deleted to ensure that there is no RAM overflow that crashes the app.

The obtained results are compared to that of other published works [6-9]. Our results outperform others. For example, in [8], the authors also used the hardware Jetson Nano, the highest accuracy for face recognition is only 86.41% while our system reaches 98.642% of accuracy. In [9], evaluations on Movidus NCS2 are reported. It is shown that the performance of the system reaches only 7.031 fps within accuracy of 93%. It is seen that our system has higher accuracy. Importantly, this is the first system which has a complete processing pipeline also including the face anti-spoofing. The optimization of a series of deep learning models in order to be able to run on Jetson Nano is a significant contribution.

For user applications, a web server was developed to manage the results of the system. The graphical user interface is shown in Fig. 5. The web server contains both a back-end and a front-end. The back-end has a user database and modules to allow administration as well as let the normal user manage the results. This back-end can communicate with other functional systems via APIs. The front-end contains several tabs for statistics, dashboard, timeline, access control, device configuration, and import and export data tools

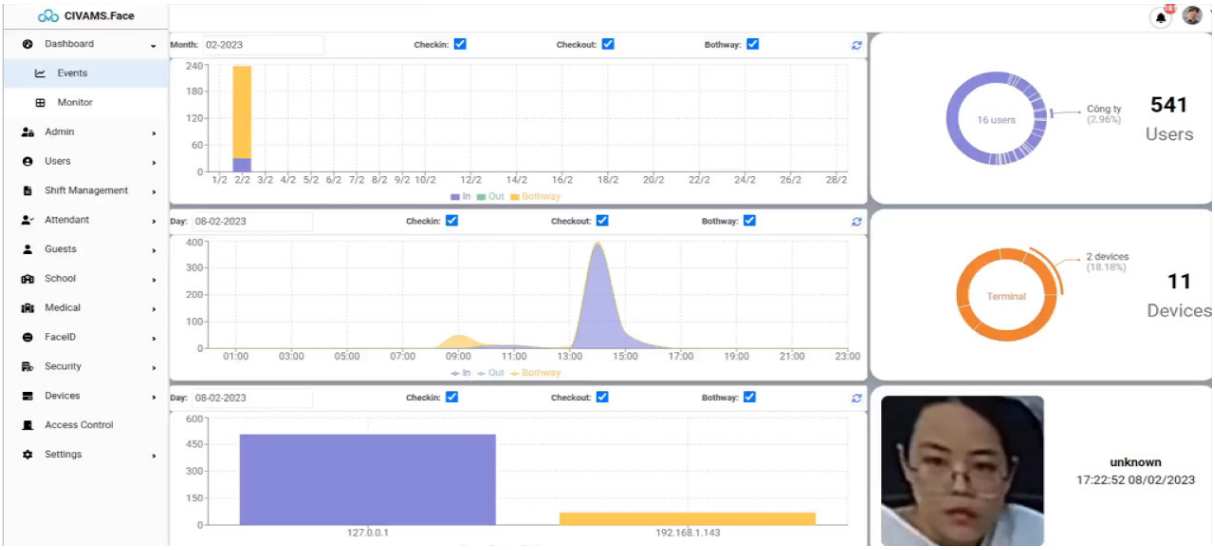

Fig. 6. User application web server

4. Conclusions and Outlook

In this work, a complete, efficient, and accurate face recognition system has been successfully developed. For an accurate system, the processing pipeline must contain a certain number of processing units, where the preprocessing is of the greatest importance. Our system is computationally efficient since only lightweight models were developed and optimized. Furthermore, the computation of these models can be accelerated using the Jetson Nano. The application of Jetson Nano, on the one hand, allows for higher computational performance, with a maximum processing ability of 4 camera streams for 40 IDs in only 458 ms. On the other hand, the SDK DeepStream has the advantage of mixing several camera streams into a batch, which reduces the utilization of hardware resources and thus improves processing capacity. It can be concluded that the system is efficient and can be used for edge-processing applications.

Several application guidelines can be developed based on the obtained results. First, the Jetson Nano can be efficiently used for the processing of two camera streams with full steps. In this case, the hardware can work independently without connecting to a central server for further processing, which allows for the high availability of the hybrid edge-central system. When the connection between the edge and the center is interrupted, the Jetson Nano can work in offline mode. Second, in case more camera streams need to be processed on site, the Jetson Nano should be used just for the detection, tracking, and anti-spoofing steps. The metadata is then sent to the center for further processing. With this concept, the jetson can be used for a maximum of six camera streams. Also, the use of the hybrid edge-center computing architecture allows us to deploy a more accurate and powerful face feature extraction model at the center. This will lead to the ability to perform face recognition on very large-scale datasets with maximal accuracy and computational performance.

In the future, the system will be extended in some directions. The use of the Jetson Nano has demonstrated that it has a high potential for use as an edge processing device. Thus, a thorough evaluation of the device in the concept of hybrid edge-central computing should be investigated.

Acknowledgement

This research is funded by the CMC Applied Technology Institute, CMC Corporation, Hanoi, Vietnam.

References

[1] Wang, Mei, and Weihong Deng, Deep face recognition: A survey. Neurocomputing, 429, 2021, pp. 215-244. https://doi.org/10.1016/j.neucom.2020.10.081

[2] Nguyen-Xuan H, Hoang-Nhu D, Nguyen-Duy A, and Dang-Minh T, A New Method for IP camera based face anti-spoofing systems, in Proc. Intelligent Systems and Networks: Selected Articles from ICISN 2022, Vietnam (pp. 445-454). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-3394-3_48

[3] Nguyen-Xuan H, Hoang-Nhu D, and Dang-Minh T, An evaluation of the multi-probe locality sensitive hashing for large-scale face feature matching. In Proc. Intelligent Systems and Networks: Selected Articles from ICISN 2022, Vietnam (pp. 445-454). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-3394-3_51

[4] Dong HN, Ha NX, Tuan DM, A new approach for large-scale face-feature matching based on LSH and FPGA for edge processing. in Proc. Intelligent Systems and Networks ICISN2021: Lecture Notes in Networks and Systems, vol 243. Springer, Singapore. https://doi.org/10.1007/978-981-16-2094-2_42

[5] Cheng KT, Wang YC, Using mobile GPU for general-purpose computing-a case study of face recognition on smartphones, in Proc. VLSI-DAT, Apr. 25, 2011, pp. 1-4.

[6] Aleksandrova O, Bashkov Y, Face recognition systems based on Neural Compute Stick 2, CPU, GPU comparison, presented at the 2nd Int. Conf. on Advanced Trends in Information Theory, Nov. 25, 2020, pp. 104-107. https://doi.org/10.1109/ATIT50783.2020.9349313

[7] Saypadith S, Aramvith S, Real-time multiple face recognition using deep learning on embedded GPU system, in Proc. APSIPA ASC, Nov. 12, 2018, pp. 1318-1324. https://doi.org/10.23919/APSIPA.2018.8659751

[8] Sati V., Sánchez S. M., Shoeibi N., Arora A., Corchado J. M., Face detection and recognition, face emotion recognition through Nivida Jetson Nano, ISAml, 2021, pp. 177-185. https://doi.org/10.1007/978-3-030-58356-9_18

[9] Xie, Y., Ding, L., Zhou, A. and Chen, G., An optimized face recognition for edge computing, presented at the 13th Int. Conf. on ASIC, Oct, 2019, pp. 1-4. https://doi.org/10.1109/ASICON47005.2019.8983596

[10] GStreamer Plugin Overview - DeepStream 6.2 Release Documentation. [Online] Available: docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_plugin_Intro.html.

[11] Nvidia DeepStream SDK. Nvidia Developer, 3 Feb. 2023, [Online] Available: developer.nvidia.com/deepstream-sdk.

[12] Wu, K., Zhang, J., Peng, H., Liu, M., Xiao, B., Fu, J. and Yuan, L., Tinyvit: Fast pretraining distillation for small vision transformers, in Proc. ECCV, Tel Aviv, Israel, October 23-27, 2022, Part XXI, pp. 68-85. https://doi.org/10.1007/978-3-031-19803-8_5

[13] Deng, J., Guo, J., Ververas, E., Kotsia, I. and Zafeiriou, S., Retinaface: Single-shot multi-level face localisation in the wild, in Proc. CVPR, Seattle, WA, USA, 2020, pp. 5202-5211. https://doi.org/10.1109/CVPR42600.2020.00525

[14] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L. C., Mobilenetv2: Inverted residuals and linear bottlenecks, in Proc. CVPR, Salt Lake City, UT, USA, 2018, pp. 4510-4520. https://doi.org/10.1109/CVPR.2018.00474

[15] Guo, Y., Zhang, L., Hu, Y., He, X. and Gao, J., Ms-celeb-1m: A dataset and benchmark for large-scale face recognition, in Proc. ECCV, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14, pp. 87-102. https://doi.org/10.1007/978-3-319-46487-9_6

[16] Nvidia Jetson Nano Developer Kit. Nvidia Developer, 28 Sept. 2022, [Online] Available: developer.nvidia.com/embedded/jetson-nano-developer-kit.

[17] Liao, S., Lei, Z., Yi, D. and Li, S. Z., A benchmark study of large-scale unconstrained face recognition, presented at Int. Conf. on biometrics, September, 2014, pp. 1-8. https://doi.org/10.1109/BTAS.2014.6996301

[18] Weisstein, Eric W. Affine Transformation. From MathWorld - A Wolfram Web Resource. [Online] Available: https://mathworld.wolfram.com/AffineTransformation.html