2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

IoT Edge Device Based Key Frame Extraction for Face in Video Recognition

Xuan Qi, Chen Liu and Stephanie Schuckers
Department of Electrical and Computer Engineering
Clarkson University
8 Clarkson Avenue, Potsdam, NY 13699, US
{qix,cliu,sschucke}@clarkson.edu

***Abstract*—Following the development of computing and communication technologies, the idea of Internet of Things (IoT) has been realized not only at research level but also at application level. Among various IoT-related application fields, biometrics applications, especially face recognition, are widely applied in video-based surveillance, access control, law enforcement and many other scenarios. In this paper, we introduce a Face in Video Recognition (FivR) framework which performs real-time key-frame extraction on IoT edge devices, then conduct face recognition using the extracted key-frames on the Cloud back-end. With our key-frame extraction engine, we are able to reduce the data volume hence dramatically relief the processing pressure of the cloud back-end. Our experimental results show with IoT edge device acceleration, it is possible to implement face in video recognition application without introducing the middle-ware or cloud-let layer, while still achieving real-time processing speed.**

***Keywords*-Internet of Things, Edge Device, Face in Video Recognition, Key Frame Extraction**

I. Introduction

Among various Internet of Things (IoT) application scenarios, video surveillance and video analytics to recognize identities and reveal human related attributes such as gender and age, etc., are of commonly adopted applications. For biometric features, face is often regarded as one of the most prominent identifiers in law enforcement, access control and authorization. Hence, face in video recognition (FiVR) has become a research area under spot light and plays a key role in video surveillance and analytics related IoT applications.

In practical application scenario, embedded platforms such as surveillance cameras and unmanned aerial vehicles (UAVs) are used as IoT edge devices for taking video streams and forward them to high-performance backend such as the Cloud. But this approach poses a great deal of processing pressure onto the backend (the Cloud) and demands high network bandwidth, especially when there are lots of sensors in use and lots of people to identify. Moreover, as shown on the left side of Fig.1, the cloud backend is under pressure in performing both low-level processing such as face detection, face tracking and face recognition, as well as high-level processing such as pattern extraction and human behavior analysis. Nowadays, with the improvement on the computation power of mobile devices, embedded platforms with integrated computing engines such as mobile graphics processing unit (GPU) actually provide us with the capability of moving FiVR

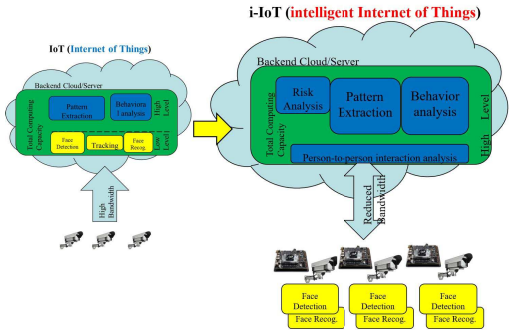

Fig. 1: Moving FR capability to IoT edge devices

to IoT edge devices. On the right side of Fig.1, by moving face detection and recognition services to IoT edge devices, we can leverage cloud backend with more computing capability for high-level analysis, such as route tracking, pattern extraction, and behavioral analysis. This is essential in moving the IoT to the next stage, i.e., intelligent IoT.

In this paper, we explore the possibility of running key-frame extraction (KFE) engine for face in video recognition on IoT edge platform. By applying GPU acceleration on our framework as well as optimization of the Convolutional Neural Network (CNN) model, we are able to reach real-time processing performance for Full HD video sequence without introducing any middle-ware components.The rest of the paper is organized as follows: in Section II, we review the main-stream approaches for distributing application across IoT center cloud and edge devices. In Section III, we describe the framework of our key-frame extraction engine. In Section IV, we evaluate the performance of our KFE engine on IoT edge device. Finally, we conclude our work in Section V.

II. Related Works

Hossian et al. proposed a cloud-assisted face and speech recognition framework [4] which used client device as image, video and voice collector and forward them to cloud server for further processing. In their experiment analysis, seconds-level processing time is acceptable for image based applications, but still not suitable for video surveillance based applications, which require real-time processing speed and low latency response. Beside, Tang et al. also suggested that offloading all

978-1-5386-5815-4/18/$31.00 ©2018 IEEE
DOI 10.1109/CCGRID.2018.00087

processing workloads to cloud back-end could lead to seconds-level latency, which does not meet real-time requirement, either [11].

In order to accelerate the processing speed and reducing the latency, other researchers introduced middle-ware to perform some pre-processing works. Tolga et al. [9] proposed MObile Cloud Hybrid Architecture (MOCHA) to build a mobile-cloudlet-cloud framework, which uses GPU-equipped cloudlet to transform raw face images into feature maps and then transfers them to cloud backend for final face recognition. With this cloudlet-based framework, they reduced the overall processing latency. Powers et al. [6] also applied cloudlet as middle level to perform the pre-processing for face recognition. Their experimental results showed that the overall processing frame rate can be accelerated by $128X$ at most. But, there are also some limitations associated with this type of cloudlet-based approaches: with the increase of face database, the acceleration ratio drops very quickly. When the database size increases to $20K$, the ratio even drops below $1X$. Hence, these results show that the cloudlet itself can also become the processing bottleneck for the whole face recognition framework.

There are some researches on performing face recognition (FR) on IoT edge devices. Cheng et al. [2] performed FR on mobile phone with integrated GPU. In their experiment, it took as much as 6s to perform feature extraction and recognition for one single detected face image, which is far from real-time speed for video processing. The speed will become even slower if we add another computing intensive procedure, face detection, to the processing pipeline. Mandal et al. proposed an FR solution on ARM-based wearable Google Glass [5], which is more similar to surveillance camera systems mostly equipped with only ARM CPUs. But in their experiment, the video resolution is only 640×360, which is too small to be representative. The computation overhead will increase dramatically when the video resolution increases to higher level such as 1080p. Hence, from all work mentioned above, we can see that performing FR is still "expensive" for IoT edge devices, and an efficient co-operating framework between cloud backend and IoT devices is needed.

The cloudlet-based works [9], [6] show the importance of reducing data volume to be transferred to cloud back-end for FR application. With the rapidly increasing processing power of IoT edge platforms, especially GPU integrated ones, the mobile device based works [2], [5] also hint the possibility of moving certain parts of the processing workloads to the edge of IoT, without introducing any middle-wares which may cause processing bottlenecks. In this work, we try to combine the advantages of both approaches.

III. KFE and FR on IoT edge devices

In our effort to meet the target of reducing data volume, we propose to pick video frames with best quality faces which are good for face recognition. Our solution is the key-frame extraction (KFE) engine.

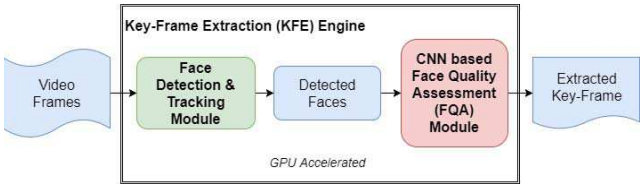

Fig. 2: Structure of our KFE engine [8]

A. KFE Framework

In Fig. 2 [8], we show the structure of our KFE engine. The KFE engine lies between the face detection/tracking module and the face recognition back-end. For every frame of the input video stream, we perform face detection first and track different people or identity. Next, for all detected faces, we forward them through the Face Quality Assessment (FQA) CNN network and get a face quality value. The meaning of the quality value here is the predicted recognition score if the face image were sent to the FR back-end. According to the quality value, the KFE engine extracts the frames with the best quality face for each identity, called key-frames, and forwards them to the FR back-end. Please note in the implementation of KFE engine, we use GPU to accelerate the face detection, face tracking and the CNN network of FQA module. We believe applying our KFE engine can bring following benefits:

- Reducing Data Volume: instead of transferring all face images or video frames to cloud back-end, our KFE engine only outputs frames with high quality faces which have higher probability to be recognized by FR back-end.
- Saving Computing Power: since the FR engine in cloud back-end only needs to process key-frames, the computing overhead can be dramatically reduced.
- Improving FR Accuracy: since the KFE engine picks frames with high quality faces, in other words, it rejects low quality faces which may cause wrong FR result. Hence, our KFE engine has the potential to improve FR accuracy, comparing with performing FR on every frame.

Our focus in this work is KFE engine. Hence, for face detection, we utilize the cascade based face detection module from OpenCV. For the face feature description file, we use the LBP descriptor with the consideration of execution speed, which is much faster than HAAR descriptor. For our CNN based KFE engine, we use CAFFE framework to implement the CNN model in our baseline version.

B. CNN based FQA in Key Frame Extraction

The CNN architecture of our FQA module is shown in Table I. Overall, our CNN has four stages of convolution operation. One unique feature of our design is that between the first and the second convolution stages, we implemented an inception module. The idea behind it is that we want to extract both fine grain and coarse grain features of a face image. Besides, all convolution layers are followed by pooling operations with 3×3 window and 2×2 stride. The last sigmoid function is used to remap numeric prediction value within the range

TABLE I: CNN architecture in our FQA module

Layer/Module	Kernel size	Num of Kernel
Image Input	64*64	1
Conv1	3*3	12
Inception	4 paths: 1*1, 3*3, 5*5 and max pooling	
Conv2	3*3	96
Conv3	3*3	128
Fc1	-	128
Fc2	-	64
Sigmoid	-	1

between 0.0 and 1.0. For more detailed design of our KFE engine, please refer to our related works [7], [8].

C. Optimization strategies on IoT edge devices

Since the CNN architecture is already defined, our focus here is computing-related optimizations. In order to reduce the computation overhead caused by our CNN model to better fit the IoT edge devices, we followed two optimization strategies: reducing the precision of the floating point and employing TensorRT framework. In case the IoT edge devices cannot reach real-time performance to meet the needs of identity extraction, object detection and behavior analysis under video based surveillance application scenario, further optimizations might be needed.

1) Half-precision float-point computing: One approach is to use reduced precision, for example, 16-bit half-length floating point, instead of using 32-bit single precision floating point to store the weights and parameters of CNN model. The hardware support for this feature is that the Maxwell and latest Pascal architectures of Nvidia's GPU support executing two 16-bit floating point based operation at the same time in one 32-bit floating point unit by following a SIMD fashion. People already showed very good outcome in using reduced precision float. For low-level computing performance, in Ho's work [3], they discovered that using half-precision can achieve at least $1.5X$, in some cases even $3X$ speedup over using 32-bit single precision floating point in their benchmarks. For deep learning related workloads, in Sze's survey [10], they showed using reduced precision floating point could lead to a huge speedup over using 32-bit precision, while at the same time the DNN model's overall accuracy loss brought by using reduced precision floating point is quite small. Hence, using half-precision float-point is highly promising in optimizing our KFE engine.

2) TensorRT framework: TensorRT is a framework provided by Nvidia for optimizing the inference of deep learning models like DNN. With TensorRT, researchers and developers can use reduced precision data type: 8-bit integer (INT8) or half-precision floating point (FP16) to replace the single-precision floating point in representing the weights and parameters of deep learning models. As a result, the overall overhead of running the models will be dramatically reduced. On the other aspect, with the support of reduced precision data on hardware level, we can meet real-time demand easier in applications with time constraints such as video based surveillance. Hence, for the optimized version of KFE engine, we replaced the Caffe framework for running the CNN based FQA model in our original design, which was oriented towards high performance computing (HPC) platform, with TensorRT framework, which is a perfect fit for IoT edge devices.

IV. Experiments

A. Experimental Settings

In this work, we use the Jetson TX2 mobile GPU board from Nvidia as our experimental platform. The Jetson TX2 platform has 6 ARM cores in total and one integrated GPU which has 256 CUDA cores. Besides, the core computing element on the TX2 board is of only credit card size, which ideally fits IoT platforms like surveillance cameras, UAVs and ground robots.

B. Video Benchmarks

To verify the performance of our key-frame extraction engine, we evaluate the baseline implementation first and introduce optimizations after that. For the evaluation, previous analysis [1] already shown that using reduced precision on DNN inferencing does not affect identification accuracy and only causes very limited precision trade-offs. Hence, we mainly pay close attention to processing speed and data volume reduction under real application scenarios, especially when there is real-time processing demand, even video surveillance with HD (1920×1080 or 1280×720) resolutions. Hence, we also took two Full HD videos (1920×1080) for the testing of our KFE engine's processing speed and data volume reduction. The first video was taken in a corridor with 4 different identities, and the second video was taken in the hallway with 22 different identities of a crowd scenario.

C. Baseline implementation

Here we ported our KFE engine, which was designed towards HPC platform, directly on the Jetson TX2 platform with no optimization. Fig.3 shows the sample output results of our KFE engine. In Fig.3a, the detected person with ID 67 appeared from Frame 1528, and Frame 1538 is extracted as the key-frame which contains the best face image for that person. In Fig. 3b, the detected person with ID 69, the Frame 1557 is extracted as key-frame. All the information is displayed as the "water-mark" at the bottom of the extracted key-frame.

To evaluate the processing speed of our KFE engine on mobile platform, we test our videos in 3 different resolutions: 1920×1080, 1280×720, and 640×480. The results are shown in Table II. We can see that for baseline implementation, it fails to reach the real-time speed (30fps) except one 640×480 case. Hence, further optimization is needed to reach real-time level processing and response.

D. Processing speed of optimized implementation

Next, we show the results after TensorRT and reduced precision optimization in Table III. We can see that after optimization, our KFE engine successfully reaches real-time processing speed (30fps) in all resolutions on Jetson platform. Especially for 1080p videos, we can see that we get more than

(a) Result for Person ID 67 (b) Result for Person ID 69

Fig. 3: Sample Result

TABLE II: Processing Speed of Baseline implementation Under Different Resolution

Video	Resolution Category	Processing Speed (Fps)
Corridor	1920×1080	14.29
	1280×720	13.88
	640×480	22.73
Hallway	1920×1080	18.05
	1280×720	18.96
	640×480	31.87

TABLE III: Processing Speed of optimized implementation Under Different Resolution

Video	Resolution Category	Processing Speed (Fps)
Corridor	1920×1080	34.07
	1280×720	33.4
	640×480	34.02
Hallway	1920×1080	34.42
	1280×720	35.87
	640×480	40.43

100% performance gain after using optimized FQA CNN with half-precision float point weights.

The reasons for getting this satisfying speedup are: first, the TensorRT framework is highly optimized towards our platform's hardware architecture. Hence, the overhead of running deep learning support framework itself is reduced. Second, the accumulative computing overhead brought by FQA engine is reduced since we use 16-bit precision instead of 32-bit. Hence, the overhead of running CNN based FQA model in our KFE engine is saved.

E. Data volume reduction

To quantify the benefit of reducing data volume brought by our KFE engine, we show the data reduction ratio in Table IV. From the table we can also see the benefit brought by our framework: more than 95% of data volume is reduced, which means now only less than 5% of video streams is transferred to cloud back-end for face recognition, meeting our goal.

V. Conclusions

With the increasing computation capability of IoT edge devices, it is possible to offload certain operations close to data, while focus the Cloud backend more on high-level processing and analysis. In this paper, we introduce a Face in Video Recognition (FivR) framework which performs real-time key-frame extraction on IoT edge devices. We employed optimization approaches which can utilize hardware architecture features of IoT edge devices such as vectorization for CPU and half-precision floating point for GPU. With our proposed approach, we are able to reduce the data volume by 95%, hence dramatically relief the processing pressure of the cloud back-end. On the other hand, with our optimization strategies, we also achieve real-time performance even for HD videos without introducing any middle layers between the cloud and edge devices.

TABLE IV: Data Reduction Ratio

Video	Total Frames	Extracted Key-Frames	Data Reduction Ratio
Corridor	300	15	95.00%
Hallway	1560	53	96.60%

Acknowledgement

This material is based upon work supported by the Center for Identification Technology Research and the National Science Foundation (NSF) under Grants No.1068055 and 1650503. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] 8-bit inference with tensorrt. http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf. Accessed: 2017-12-12. 3

[2] K.-T. Cheng and Y.-C. Wang. Using mobile gpu for general-purpose computing–a case study of face recognition on smartphones. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1–4. IEEE, 2011. 2

[3] N.-M. Ho and W.-F. Wong. Exploiting half precision arithmetic in nvidia gpus. In *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, pages 1–7. IEEE, 2017. 3

[4] M. S. Hossain and G. Muhammad. Cloud-assisted speech and face recognition framework for health monitoring. *Mobile Networks and Applications*, 20(3):391–399, 2015. 1

[5] B. Mandal, S.-C. Chia, L. Li, V. Chandrasekhar, C. Tan, and J.-H. Lim. A wearable face recognition system on google glass for assisting social interactions. In *Asian Conference on Computer Vision*, pages 419–433. Springer, 2014. 2

[6] N. Powers, A. Alling, K. Osolinsky, T. Soyata, M. Zhu, H. Wang, H. Ba, W. Heinzelman, J. Shi, and M. Kwon. The cloudlet accelerator: Bringing mobile-cloud face recognition into real-time. In *Globecom Workshops (GC Wkshps), 2015 IEEE*, pages 1–7. IEEE, 2015. 2

[7] X. Qi and C. Liu. Gpu-accelerated key frame analysis for face detection in video. In *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, pages 600–605. IEEE, 2015. 3

[8] X. Qi, C. Liu, and S. Schuckers. Boosting face in video recognition via cnn based key frame extraction. In *The 11th IAPR International Conference on Biometrics (ICB), 2018 IEEE*. IEEE, 2018. 2, 3

[9] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000059–000066. IEEE, 2012. 2

[10] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer. Efficient processing of deep neural networks: A tutorial and survey. *arXiv preprint arXiv:1703.09039*, 2017. 3

[11] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot. Enabling deep learning on iot devices. *Computer*, 50(10):92–96, 2017. 2