Intelligent Solutions in the COVID-19 Era with AI Camera

Ming-Chih Lin
Cloud + AI
Microsoft
Taipei, Taiwan
tolin@microsoft.com

Ti-Hua Yang
Cloud + AI
Microsoft
Taipei, Taiwan
anneyang@microsoft.com

Yu-Kwen Hsu
Cloud + AI
Microsoft
Taipei, Taiwan
yukwenh@microsoft.com

Chris McMillan
Cloud + AI
Microsoft
Taipei, Taiwan
chrismc@microsoft.com

Dan Rosenstein
Cloud + AI
Microsoft
Taipei, Taiwan
danrose@microsoft.com

Jussi Niemela
Cloud + AI
Microsoft
Taipei, Taiwan
juniem@microsoft.com

ABSTRACT

To stand up our next generation AIoT ecosystem, Microsoft is working with industry partners to build AI camera that integrates lens, camera sensors, AI accelerator with security chips, a host machine that connects to Azure cloud services in a simple to develop, integrate and maintain software, services, and hardware solution. The AI camera supports popular classification and object detection deep neural networks and achieves high inference throughput in consideration of energy efficiency. In addition, there are new needs coming from verticals to suppress disease transmission after the outbreak of COVID-19. Thus, we use AI camera and Azure integrated toolchain to build intelligent solutions. Those solutions are easy to be built in a short time to quickly respond the emergent needs in COVID-19 era.

KEYWORDS

AIoT, AI camera, edge devices, cloud, container

ACM Reference format:

Ming-Chih Lin, Ti-Hua Yang, Yu-Kwen Hsu, Chris McMillan, Dan Rosenstein and Jussi Niemela. 2020. Intelligent Solutions in the COVID-19 Era with AI Camera. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 Introduction

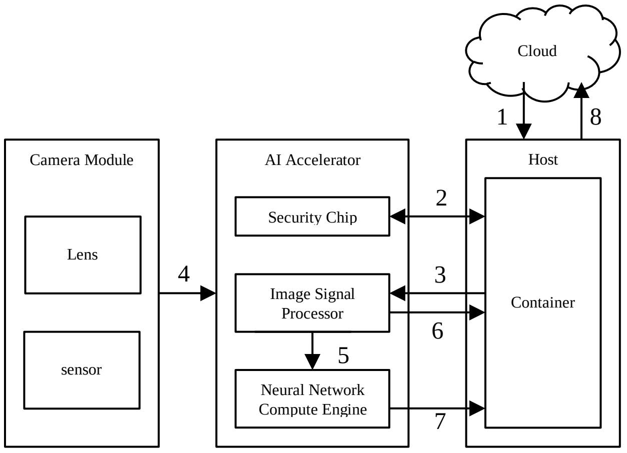
With the growth of connected IoT devices and AI, companies including start-ups and corporate giants recognize the paradigm shift that AIoT brings and the business opportunity with it. They start looking for intelligent edge solutions to process the sensor data with AI accelerators. In this AIoT ecosystem, silicon evaluation kits and reference devices are top of the stream. Device builders largely edit these to build application specific instances.

Cameras provided rich data across verticals. We have received the inquiries of building scalable and secure end to end AI solutions with camera sensors from construction, manufacturing, retail, smart building, transportation, etc. verticals. To stand up our next generation AIoT ecosystem, we work with ecosystem partners to build AI camera that integrates lens, camera sensors, AI accelerator with security chips, a host machine that connects to Azure cloud services in a simple to develop, integrate and maintain software, services, and hardware solution. In addition, Azure integrated toolchain provides various paths to build, customize, and manage their solutions easily.

After the outbreak of coronavirus COVID-19, verticals are looking for quick respond solutions to suppress disease transmission. For example, face masks detection for building entrance, car type, color, license plate detection for curb pickup, etc. We use the AI camera and Azure integrated toolchain to build several solutions for the scenarios in COVID-19 era.

2 Architecture of AI Camera

AI Camera integrates lens, camera sensors, AI accelerator with security chips, and a host machine. Figure 1 shows the flow of AI camera. AI Camera is an AIoT device that provides vision sensors to collect data. AI accelerator processes the image data and perform inference with deep learning models. Host machine is an IoT gateway connected to Internet to clean and forward data to cloud for further analysis. The camera module connects to the AI accelerator by a MIPI cable, and the AI accelerator connects to the host by a USB cable. First, the AI model is packed in a container [1] that deployed to the host. It also authenticates the security chips on edge compute module. Second, the host sends AI model to AI accelerator and a start signal to initiate the camera frame processing. The image frames are captured and sent to the image signal processor for pre-processing. The pre-processed image is directly sent to the AI accelerator for running inference. Then, the inference results and camera stream are sent back to the host. Finally, the host process the results in container, it can provide RTSP (real time streaming protocol) stream to RTSP clients and send data and telemetry to Azure cloud for further analysis.



1. Deploy the container from Azure cloud to host.
2. The container starts and authenticates with the security chip.
3. If the AI camera is authenticated, the container sends the AI model to AI accelerator and a start signal to initiate camera capture.
4. Camera frames are captured and sent to image signal processor of AI accelerator for pre-processing.
5. The pre-processed image is directly sent to neural network compute engine for running inference.
6. Camera stream is sent to host.
7. Inference result is sent to host.
8. Process the result in container, provide RTSP stream, send data and telemetry to Azure cloud for further analysis.

Figure 1: Flow of AI camera

To get an AI model, there are three primary options. The first one is to use the prebuilt models from the model zoo. Model zoo is a collection of pre-trained, state-of-the-art models contributed by community members. The second one is to leverage transfer learning to retrain your own computer vision models that fit with unique use cases with some images. Here we integrate with Azure Custom Vision [2], which is optimized to quickly recognize major differences between images, so you can start prototyping your model with a small amount of data and just a few clicks. Azure Custom Vision provides different output formats for various devices, including the model for the AI camera. Figure 2 shows the flow of retraining your own computer vision models for AI camera. The third one is for data scientists to build your own model and convert to the format for the AI camera.

To deploy the solution at scale, the software, including AI models, inference application, business logics, etc., are packed in containers [1], which is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Then, you can deploy those containers to any of your devices and monitor it all from the cloud. Azure IoT Edge [3] moves cloud analytics and custom business logic to devices so that your organization can focus on business insights instead of data management. Here we use Azure IoT Edge to deploy, monitor, and manage IoT edge devices.

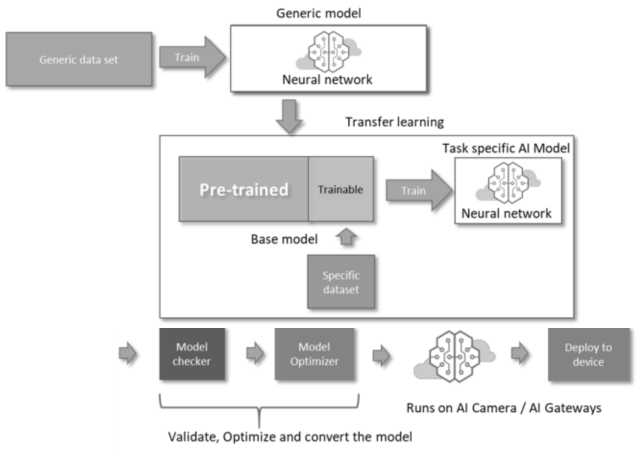


Figure 2: Flow of retraining your own computer vision models

Figure 3 shows two primary options to create and update the solution with container. The first one to create a new container that packs AI models, inference application, business logics, and all its dependencies to the container. Then, deploy the container by Azure IoT Edge. The Second one is to leverage an existing container and update the AI models using module twin update. Module twins are JSON documents that store module state information including metadata, configurations, and conditions. The AI models are updated by specifying module twin properties of the locations of AI models from cloud. The device receives the updated properties and downloads new AI models to host. Host sends the AI model to AI accelerator and sends a restart signal to initiate camera capture.

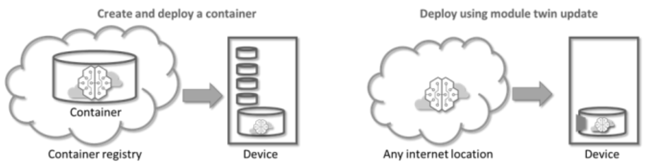

Create and deploy a container
Container
Container registry
Device
Deploy using module twin update
Any internet location
Device

Figure 3: Create and update the solution with container

3 Results

We evaluate the performance of AI camera by power consumption and inference throughput since they are important for AIoT devices. In addition, we use Azure integrated toolchain for AI camera to build solutions for the post COVID-19 world in a short time.

Table 1 shows the maximum power consumption of components. The camera module only captures images from image sensor, which consumes little power. The AI accelerator runs a RTOS (real time operating system) that processes image and perform AI inference, which consumes most power in the AI camera. The host runs a Linux based OS that communicates between cloud and AI accelerator, it consumes less power than the AI accelerator.

Table 1: Power consumption of components

	Camera Module	AI Accelerator	Host
Max power consumption	<0.1 W	4.5 W	4 W

Table 2 shows the inference throughput of different models. We choose popular classification and object detection deep neural networks run on the AI camera with FP16 precision to measure the inference throughput of AI accelerator in FPS (frames per second). The experimental results show it can achieve high FPS for most of the popular classification and object detection deep neural networks with consideration of power consumption.

Table 2: Inference throughput of models

Model		**Input size (pixel)**	**Inference Throughput (FPS)**
Classificat ion	ShuffleNet v2 [4]	224 x 224	70.6
	ResNet-50 [5]	224 x 224	28.0
	Inception v4 [6]	299 x 299	6.3
Object Detection	MobileNet v2 SSDLite [7]	320 x 320	34.3
	Tiny YOLO v3 [8]	416 x 416	22.7
	YOLO v2 (without non maximum suppression) [9]	416 x 416	16.3
	MobileNet SSD v2 [10] [11]	300 x 300	14.0
	YOLO v3 [8]	416 x 416	5.6
	Faster RCNN (with ResNet50 backbone) [12]	600 x 600	0.6

3.1 Face mask detection

After the outbreak of COVID-19, lots of buildings need additional staff stand at the entrance to make sure anyone walking in wears a face mask and use infrared thermometers to check if anyone has a fever, which is to detect people who might have the coronavirus. However, most of buildings (including hospitals) would rather have their staff do other jobs. Those assigned to check people coming in for face masks and fever also face the risk of catching the coronavirus.

To build the mask detection solution with AI camera, we leverage the face detection model in the model zoo. Once the face is detected, we apply a classification model to check if there is a mask in the face region. The mask classification model is trained using Azure Custom Vision service and exported to run with the AI accelerator. The inference results can send to cloud and provide a dashboard for monitoring and analysis.

3.2 Curbside pickup

In this COVID-19 era, we still need to get essential supplies, groceries, etc., and we need to do this safely and securely. Imagine a customer is going online the local retailer and making a grocery purchase as part of the customer's account, and chosen to have the car type, color, and license plate registered with the retailer. When the customer pulls up to the retailer, the car type, color, and license plate are detected. The online purchase products are brought out and placed in the trunk.

To build car type, color, license plate detection solution for curbside pickup solution, we took pictures of different types of cars, different colors of cars from all angles, inside parking garages, in the shade, and outside in the sun. Some with license plates and others without. Those pictures are uploaded to the Azure Custom Vision to train a classifier. The model is exported, packed to containers, and deployed to the AI camera. The inference results can send to cloud and integrate with retailer's information systems.

4 Conclusion

AI camera integrates hardware accelerated AI with security chips and connect to Azure services in a simple to develop, integrate and maintain software, services, and hardware solution. It supports popular classification and object detection deep neural networks and achieves high throughput in consideration of power consumption. Azure integrated toolchain makes it easy to build the solution in a short time and deploy at scale. We will keep working with ecosystem partners to build various AI powered intelligent edge devices.

REFERENCES

[1] Bukhary Ikhwan Ismail, Ehsan Mostajeran Goortani, Mohd Bazli Ab Karim, Wong Ming Tat, Sharipah Setapa, Jing Yuan Luke, and Ong Hong Hoe. 2015. Evaluation of docker as edge computing platform. In *Proceedings of IEEE Conference on Open Systems*, 130-135.

[2] Mathew Salvaris, Danielle Dean, and Wee Hyong Tok. 2018. Cognitive Services and Custom Vision. *Deep Learning with Azure*. Apress, Berkeley, CA.

[3] David Jensen. 2019. Azure IoT Edge Core Concepts. *Beginning Azure IoT Edge Computing*, 17-47. Apress, Berkeley, CA.

[4] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of European Conference on Computer Vision*, 116-131.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

[6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of AAAI conference on artificial intelligence*, 4278-4284.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510-4520.

[8] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[9] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7263-7271.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE conference on computer vision and pattern recognition*, 4510-4520.

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, and Cheng-Yang Fu, Alexander C. Berg. 2016. Ssd: Single shot multibox detector. In *Proceedings of European conference on computer vision*, 21-37.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of Advances in neural information processing systems*, 91-99.