IoT and Facial Recognition at Scale: Using Amazon's DeepLens to search for matches from the US's Missing Persons Database

Andrew Carlson, Vicki Foss, Gerard Kelly, Michelle Liu

MIDS W251 - Scaling Up! Really Big Data
University of California, Berkeley
August 2018

Introduction

In the United States alone, at least 14,500 people are missing today [1]. By definition, these missing people could be anywhere, which makes "crowd sourcing" the search an important tactic in the effort to locate them. Historically, this has meant the television broadcasting of photos of the missing, disseminating posters or even printing missing children's faces on milk cartons—all in the hope that someone who encounters a missing person in a public space will recognize them for who they are. While there are some cases of these sorts of campaigns being successful, they are generally the minority. Furthermore, several psychology experiments have found that even when a poster of a "missing person" (really an actor) is placed close to where that real person is, a small percentage of passersby even recognize the person from the poster (generally less than 5%), and if they do, they often do not speak up out of fear of getting involved [2]. A separate experiment found that the more "missing person" announcements a person sees, the less likely they are to pay attention and identify those people when encountered in real life [3]. Other research has shown that there is a race and gender disparity in the missing persons cases that receive the most media attention, with white people and women and girls getting a disproportionate amount [4]. For all of these reasons, our fellow human beings tend to make poor partners in the search for missing persons.

Fortunately, computers have now surpassed humans in recognizing and identifying a human face [5]. Computers do not fail to identify a person because of their gender or race[1], for fear of getting involved or lack of paying attention. As such, well-placed cameras in public spaces equipped with facial recognition software and trained to recognize people from the images scraped from The National Missing and Unidentified Persons System (NamUs) database could be a better method for identifying missing people "in the wild" and alerting authorities in real-time. In this paper, we describe a cloud-based architecture that uses Amazon's DeepLens to

[1] It should be noted, however, that facial analysis algorithms are often prone to gender and racial bias themselves. A recent study which tested several commercially available gender classification systems demonstrated that darker-skinned women were most likely to be misclassified (with the highest error rate reaching 34.7%), while light-skinned men were rarely misclassified (maximum error rate was just 0.8% for this group) [6].

correctly identify test subjects and pictures of people in the NamUs database, and which sends a text message with the person's information upon finding a match. While there are monetary, logistic, and privacy concerns that might prevent such a system to be adopted nationwide, we are still able to demonstrate its remarkable effectiveness.

The Data Source

The National Missing and Unidentified Persons System (NamUs) has a database with records of approximately 14,400 missing people [1]. About 12,300 of those records contain at least one image, with an average of two images per person, resulting in a total of more than 24,600 images. The size of the images range from 100s of kB to 10s of MB. Due to the real-time requirements of the application, the entire corpus of images cannot be processed on each query. Achieving low latency and horizontal scalability with this type of data will be the design goals, and dictate the choice of technologies.

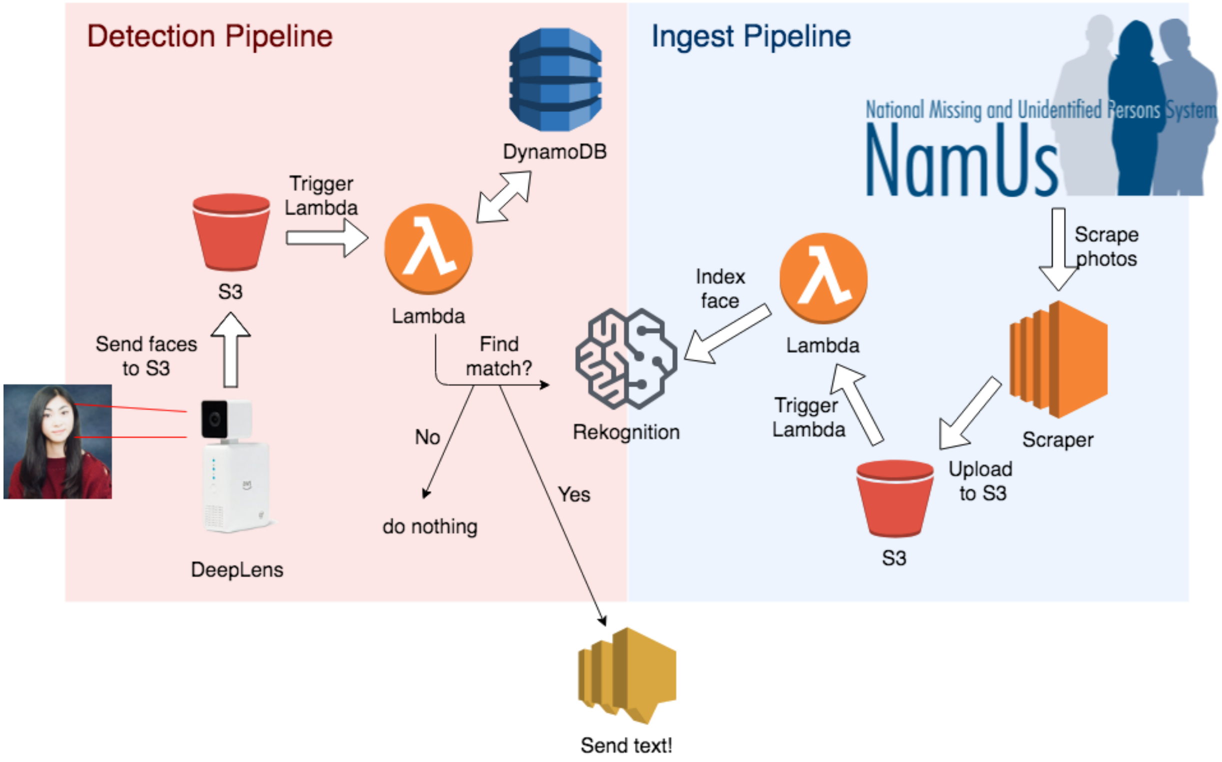
Technology Stack

The architecture is comprised of a cloud-based, serverless[2] backend that interfaces with edge devices via object storage uploads. In this case, our cloud provider is Amazon Web Services (AWS). AWS was chosen primarily for the abundance of documentation/tutorials, pleasant developer experience, and out-of-the-box integration with the Amazon DeepLens, an edge device which one of us happens to possess.

Arbitrary edge devices which recognize faces will upload to object storage that triggers downstream processing. The processing entails running a facial recognition model, and sending a notification when a match on a missing person is found. These operations will be collectively called the "detection pipeline". It assumes that a collection of missing persons' faces is built and ready to query.

This collection is built by a set of processes termed the "ingest pipeline". As mentioned above, this collection of faces must not be processed on each query, but rather preemptively indexed to meet our application's low-latency requirements. Each missing person's information and photos were enumerated from the NamUs servers and uploaded to object storage. This triggered the indexing operation, so that the model can recognize the photo in future queries.

[2] Serverless insofar as we did not need to write, or even manage our own servers. The only servers involved are owned by AWS and NamUs.



Tools

We relied heavily on Amazon Web Services in order to implement our system. This section details the specific AWS tools we used and how they were set up and incorporated into pipelines.

S3. S3 object storage is used to store each image. There are two buckets, one for ingestion and one for detection. In the ingestion pipeline, the bucket stores the database of missing persons. On the detection side, there is a bucket that stores every face detected which is a candidate for a missing person query. This serves as the interface for the edge devices to submit the faces they detect.

Lambda. AWS's implementation of serverless compute breaks application logic into function-based units called Lambdas. These are triggered either by user-defined, custom triggers such as changes in data or by other AWS services such as S3 and DynamoDB. Lambda makes it easy to build a real-time serverless data processing system that is scalable and fully managed on AWS's infrastructure. It is very cost-effective as we only pay for the compute time actually used, instead of having an always-on server.

All the application logic is implemented in two lambdas. The first is triggered during the scraping process. When a new photo is added to S3, a lambda adds the new face to an index in AWS Rekognition. The other lambda triggers on uploads to the other bucket. This takes any detected face and submits it to Rekognition in a search operation to see if it can find matches in our index. If it can, it calls AWS Simple Notification Service (SNS) to send a text message to the team.

Rekognition. Rekognition is Amazon's collection of pre-trained machine learning models exposed as a service. The model we used is a facial recognition model[3] which can perform object detection to first find faces in photos, and then encode faces into a vector known as "embeddings". These embeddings are a very compact numerical encoding for a face which is "learned" during the training process. During indexing, it is actually the embeddings which are stored in an index. On each query, the photo is converted to its embeddings and searched against the known embeddings of all the missing persons' faces. This is how over 20,000 images can be searched in real-time.

DynamoDB. DynamoDB is Amazon's configurable, scalable document store. We use it to store history of when notifications are sent. This is primarily to address the problem of notification "bursts". When a subject remains in the viewport of the DeepLens for a sustained period of time, many uploads are sent in rapid succession. If that person is a match and texts are sent every time, it leads to an irritating experience. This is prevented by persisting this information in DynamoDB, so the lambda can avoid sending too many repeat notifications.

SNS. AWS Simple Notification Service (SNS) is Amazon's publish/subscribe messaging service that can be called to send text messages.

IAM Roles. IAM roles are a set of credentials and permissions given to AWS services that specify access control to other services. For example, our Lambdas are given access to S3, Rekognition, and CloudWatch (for monitoring and logging). If these pieces of compute can only access other services that are actually needed, the surface area for security breaches is minimized. In the event that an attacker injects malicious code on one of our Lambdas, they will not be able to provision EC2 instances and start a botnet/bitcoin mining operation.

DeepLens. AWS DeepLens is a wireless deep learning enabled video camera that is integrated with the AWS Cloud. The DeepLens camera uses deep convolutional neural networks[4] to analyze visual imagery in real time. DeepLens runs Ubuntu OS-16.04 and is preloaded with

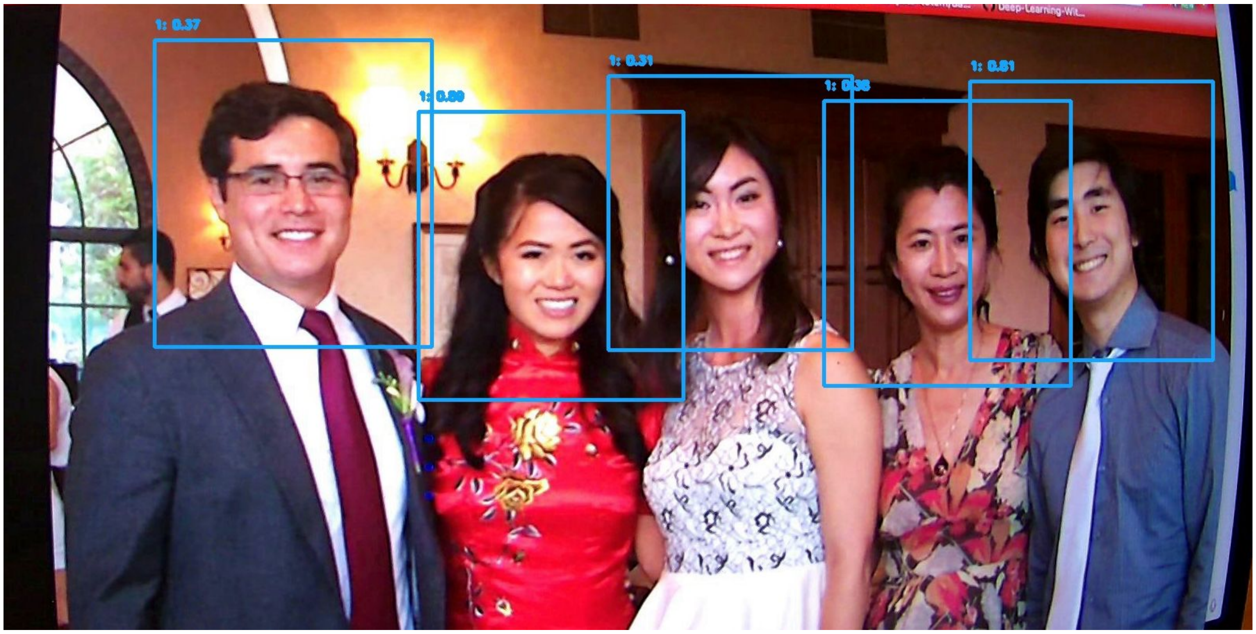
[3] For more information, see https://aws.amazon.com/blogs/machine-learning/build-your-own-face-recognition-service-using-amazon-rekognition.

[4] https://gluon.mxnet.io/chapter04_convolutional-neural-networks/cnn-scratch.html

the Greengrass Core[5], which allows for the deployment of either pre-trained or custom deep learning models that can detect faces, objects, activities, texts, and much more.

Our project leverages the pretrained model template for face detection functionality.[6] In the AWS DeepLens console, we created a project with the face detection template. The AWS console provides the default model with a Lambda function that come with the project template. We updated the lambda function so that when the device detects human faces, the project stream containing the frame is extracted, and the face is cropped out and sent as a JSON message IoT.[7] Next, we created a rule in IoT to upload the image to an S3 bucket through another Lambda call if the confidence level of a face detection is over 80%.

Once the project is deployed to the DeepLens device, the project output can be viewed on screen. Below is an example of the project stream:



Data Ingestion from NamUs

For the typical user, NamUs is typically interacted with via the web page. After making an account, the user is directed to an AngularJS application which allows them to search missing persons based on several criteria and see other information about them, including photos. Searching by image cannot be done. This dashboard is not a useable format for our application.

[5] https://aws.amazon.com/greengrass/

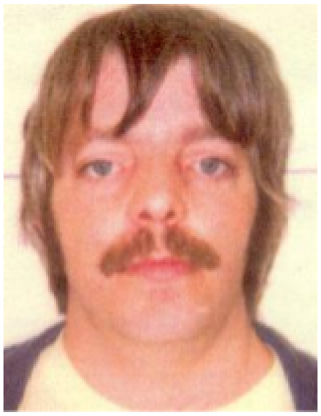
[6] See https://github.com/darwaishx/Deep-Learning-With-Deep-Lens/tree/master/4-FaceDetectionAndVerification/1-FaceDetection.

[7] See https://github.com/acarl005/iot-missing-persons-detector/blob/master/src/deeplens_face_detection.py.

A virtual server was provisioned (AWS EC2) for the scraping. An Ubuntu 16.04 image was given an IAM role with write access to the S3 bucket, and used to run a Python 3 program. The program enumerated data about each missing person via an undocumented API. The URL for this API was found by reverse-engineering the Angular app with the Chrome DevTools. Inspecting the “Network” tab in the DevTools, and filtering on the XHR requests, we looked through all the requests made from the results page for a missing person. We identified this URL, which returns information about the case: **`https://www.namus.gov/api/CaseSets/NamUs/MissingPersons/Cases/<primary_key>`**.

Once the appropriate URL was found, it was clear that it was parameterized by a primary key, which is an incrementing integer. The Python program loops up through the space of primary keys starting from 1, gathering information about each person. Among this information, there are URLs to a static file server which hosts the person’s photos.[8] These image files are subsequently downloaded and submitted to the S3 bucket, triggering the indexing process.[9]

The indexer was able to find a face in about 94% of the images. Photos work best when the face is in color, and the subject is facing the camera.



Face detected

Face detected

No face detected

Detection Pipeline

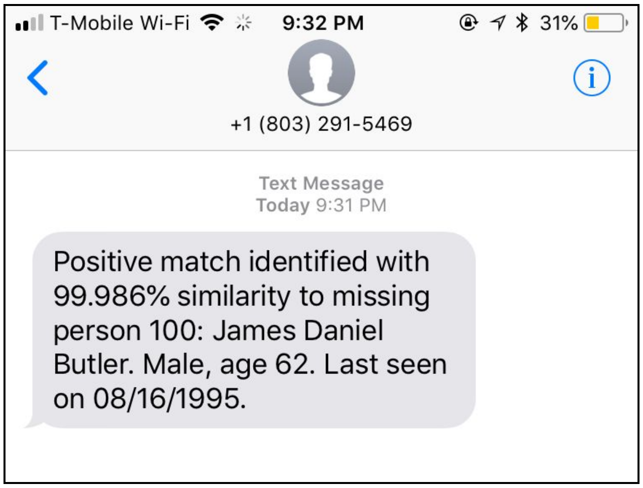
Face comparison via Rekognition. The lambda function, 'lambda_rekognition.py', is invoked every time the DeepLens detects a face in the frame and sends the cropped face image to the S3 bucket. The script calls the Amazon Rekognition Search By Face API to compare faces in the S3 bucket, created in the Data ingestion from NamUs process. If a match is found, it checks to see if the ID already exists in the database we created with DynamoDB. If the ID is not already in the DynamoDB database, this means that the person has not been detected by DeepLens before, and a new entry is created in the table with the face ID as the primary key. The

[8] See https://github.com/acarl005/iot-missing-persons-detector/blob/master/src/ingest/scrape.py.

[9] See https://github.com/acarl005/iot-missing-persons-detector/blob/master/src/ingest/index_face.py.

current time is also stored in the table as time of detection in order to address the problem of notification bursts as discussed earlier. If the ID already exists in the table, the entry is only updated if at least one minute has passed since the time of last detection. In the case of a new match being found (that was not already in DynamoDB) or a match that was already in DynamoDB but with a timestamp of more than a minute before, the alert process begins.

Retrieving person information and sending alert if new match found. When a match is found, the external ID of the matched image from the Rekognition index is assigned to a variable, likewise for the percent similarity of the match. The external ID is then used to call the same NamUs (undocumented) API previously described in the Data Ingestion section of this paper. This allows us to retrieve the most up-to-date information about the person identified, including their name, age, sex, and date last seen. This information is then formatted into a short text message and sent to a previously determined number using AWS Simple Notification Service (SNS), allowing the end-user of this system to receive an alert about a missing-person sighting in real-time.[10]



Positive match identified with 99.986% similarity to missing person 100: James Daniel Butler. Male, age 62. Last seen on 08/16/1995.

Discussion of Results

We are unable to test the "true positive" rate of our detection system due to the nature of this problem. In order to approximate the real-world performance, we added some of our own faces into Rekognition manually. The system is able to detect our faces with high fidelity, with receipt of the subsequent text message occurring one to three seconds later.

By leveraging the benefits of serverless computing, we have an inherently horizontally scalable infrastructure with no management overhead. As more faces get detected, the lambdas

[10] For the Lambda script, see https://github.com/acarl005/iot-missing-persons-detector/blob/master/src/lambda_rekognition.py.

will execute on elastic compute resources managed by Amazon. The compute utilization is high since we are billed only when lambdas are actually called, removing the risk of over-provisioning. Furthermore, the design of the detection pipeline is extensible with other edge devices, enabling geographically diverse deployment. The end result is that we have built a scalable, highly cost-effective, and robust detection system around Amazon Rekognition.

Future Work

The success of our person recognition and alert system--as well as its rather elegant, efficient, and effective technology stack--naturally raises the question of how we could build upon it. Extensions could be made to both the detection and ingestion sides.

On the ingestion side, the collection of missing person images could be augmented by equivalent official databases (e.g. those of the UK[11], Canada[12], Australia[13], or Interpol[14]) for countries other than the US. The scope of the project could also be broadened beyond missing persons by including databases for other persons of interest (e.g. wanted criminals[15], fugitives[16], terrorists[17]), from which facial images could be similarly scraped and indexed.

On the detection side, the Amazon DeepLens's on-board pre-trained face-detection model, combined with its easy integration into Amazon's suite of services for data storage and retrieval, make it a particularly suitable device for real-time recognition of missing persons or other persons of interest. However, such deep-learning models for the detection of faces can be used not only with edge devices but also with images sourced from social media (e.g. Twitter streams) or acquired from the Web (e.g. sites like backpage.com that have been implicated in human trafficking), with the same recognition pipeline able to compare these detected faces against a database of missing persons and persons of interest.

The type of real-time human recognition and alert system demonstrated by this project will be made increasingly effective by the rapid proliferation of edge devices and the continuing development of cloud technologies and artificial intelligence. In the short-term, some of the main barriers to developing such systems will be non-technological, including concerns over the implications of mass-surveillance on individual privacy. The relative absence of these barriers in China have contributed to a greater prevalence of facial recognition technology in that country,

[11] https://www.missingpeople.org.uk/help-us-find.html

[12] https://www.services.rcmp-grc.gc.ca/missing-disparus/search-recherche.jsf

[13] https://missingpersons.gov.au/view-all-profiles

[14] https://www.interpol.int/notice/search/missing

[15] https://www.interpol.int/notice/search/wanted

[16] https://www.fbi.gov/wanted/fugitives

[17] https://www.fbi.gov/wanted/wanted_terrorists

where its use by law enforcement is developing rapidly, and a national database of individuals flagged for automated detection is reported to include 20 to 30 million people [7].

Despite the rapid advancement in the deep learning underpinning these recognition systems, the core technology for matching the mapped embeddings for different images of the same face remains imperfect. This can lead not only to a failure to detect matches, but also to false positives. This vulnerability was recently demonstrated when Amazon Rekognition wrongly matched images of 28 U.S. legislators (disproportionately African-American and Latino) with images in a database of 25,000 people who had been arrested. However, this error rate resulted from using a "confidence threshold" of 80 percent, lower than the 95 percent recommended by Amazon to law enforcement. Amazon's response also noted that Rekognition is typically used as a complement to human judgement [8].

References

[1] National Institute of Justice, *National Missing and Unidentified Persons System*, 2018. https://www.namus.gov/.

[2] J. M. Lampinen and K. N. Moore, “Prospective Person Memory in the Search for Missing Persons,” in *Handbook of Missing Persons*, S. J. Morewitz and C. S. Colls, Eds. Cham, Switzerland: Springer, 2016, pp. 145-162.

[3] J. M. Lampinen, and K. N. Moore, “Missing person alerts: does repeated exposure decrease their effectiveness?”, *Journal of Experimental Criminology*, vol. 12, no. 4, pp 587-598, Dec. 2016. https://doi.org/10.1007/s11292-016-9263-1.

[4] Z. Sommers, “Missing White Woman Syndrome: An Empirical Analysis of Race and Gender Disparities in Online News Coverage of Missing Persons”, *Journal of Criminal Law and Criminology,* vol. 106, no. 2, pp. 275-314, 2016. https://scholarlycommons.law.northwestern.edu/jclc/vol106/iss2/4.

[5] C. Lu and X. Tang, “Surpassing Human-Level Face Verification Performance on LFW with GaussianFace,” arXiv:1404.3840v3 [cs.CV], Dec. 20, 2014.

[6] J. Buolamwini and T. Gebru, “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification”, *Proceedings of Machine Learning Research,* vol. 81, pp. 1-15, 2018. http://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf.

[7] P. Mozur, “Inside China’s Dystopian Dreams: A.I., Shame and Lots of Cameras.” *New York Times*, July 8, 2018. https://www.nytimes.com/2018/07/08/business/china-surveillance-technology.html.

[8] N. Singer, “Amazon’s Facial Recognition Wrongly Identifies 28 Lawmakers, A.C.L.U. Says”, *New York Times*, July 26, 2018. https://www.nytimes.com/2018/07/26/technology/amazon-aclu-facial-recognition-congress.html.