Facial Recognition for Fun and Profit: How Any Technologist can get started with AI

Artifical Intelligence are and Machine Learning are important and we are starting to see them everywhere. Almost every device we use, from our car, to our phone, to wearables have features that are enabled via AI or ML. I want to be a part of this technology revolution but I'm not a data scientist or a neural network expert. How can I get started with AI or ML? How can I learn?

You Start with an Idea

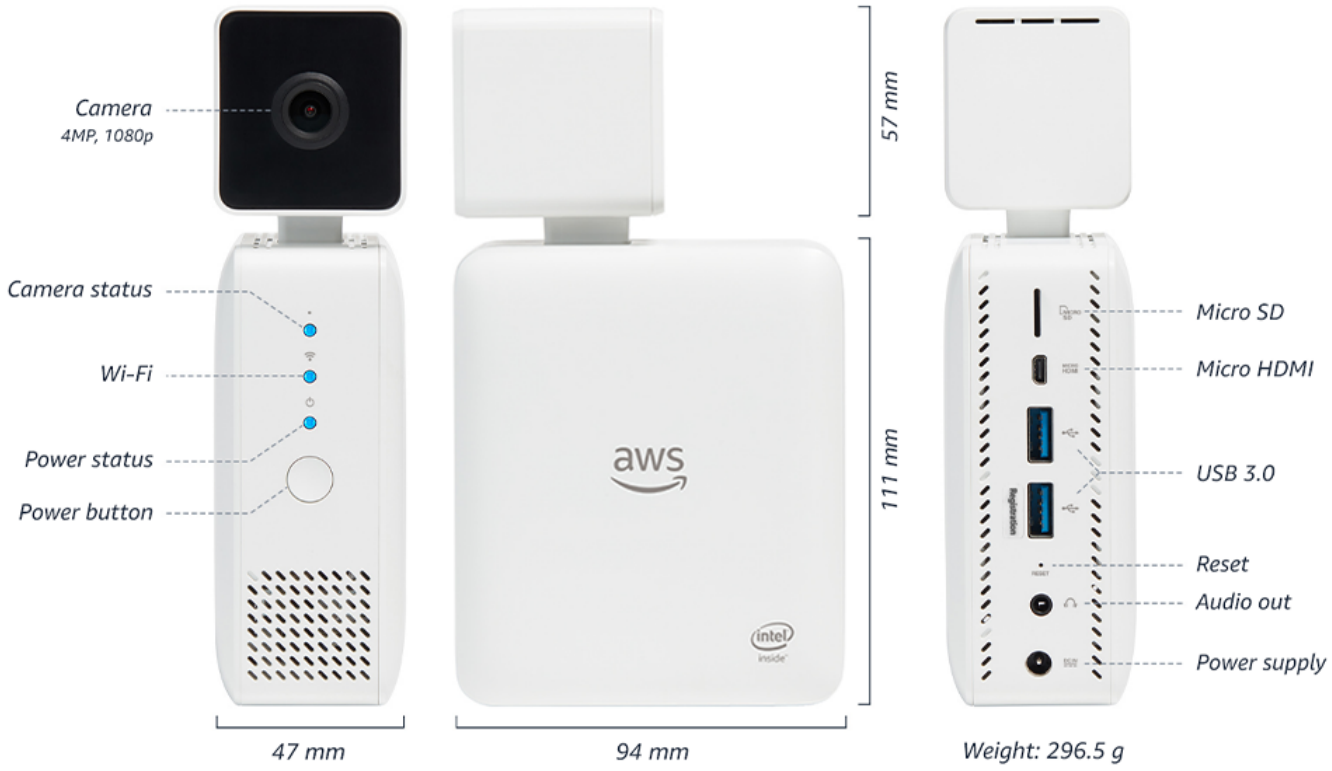
The good new is there is a path and it starts with an idea. I had an idea to build a system that uses facial recognition to identify people that enter a room and then play entrance music for them. So with this idea I started thinking about what that solution would look like.

I would need a few things:

1. A device that had a camera that I could access programmatically to detect faces when people entered the room
2. A component that could match those images taken with faces to known people with entrance music
3. An application or device to play the music if a match is found

The Device

I started by selecting an AWS DeepLens. The AWS DeepLens is a powerful learning tool that gives developers a computing platform that has a built in high-definition camera that is easy to integrate with other AWS services. For the beginner, there is a large learning community that have create machine learning models and starter application code that can be used to kick-start your project. For the experienced ML/AI technologist, you can create your own models and algorithms that take advantage of the AWS DeepLens' substatial capabilities.



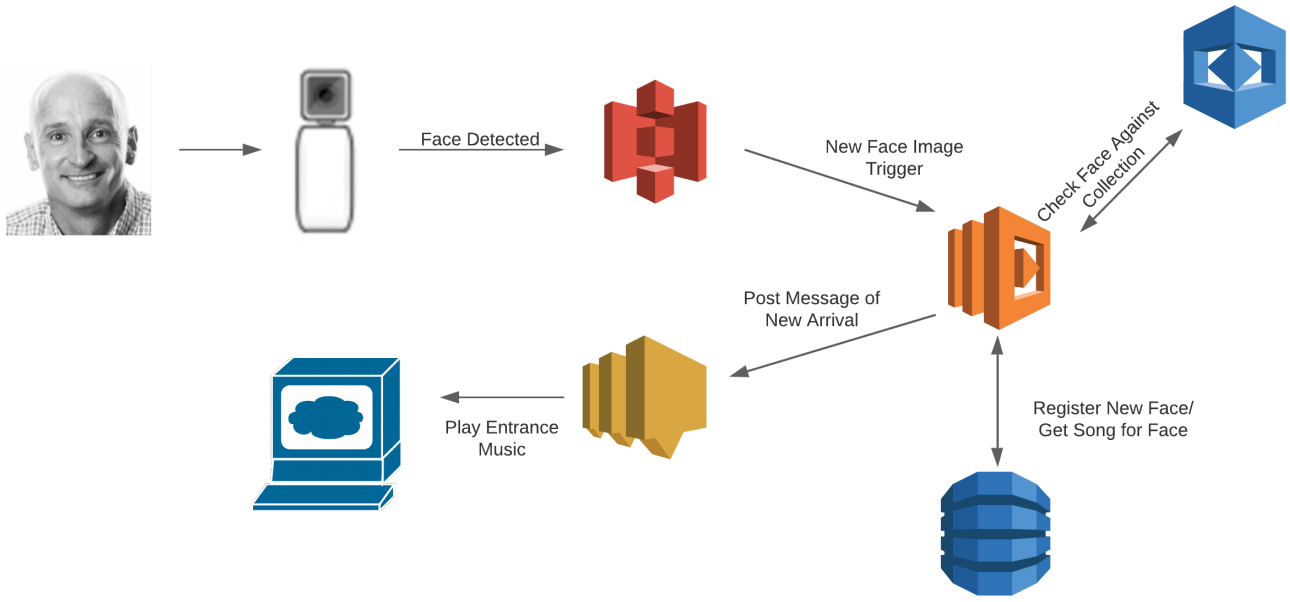
The Starting Point

The approach I took was to use an existing AWS DeepLens starter project that did something similar or related to what I wanted to do and modify it to my purpose. The goal was to implement my idea and learn about how ML/AI projects were designed without having to start from scratch in an area to which I was new.

My starting point was a project that used facial and object detection to determine who in an office was drinking the most coffee and then update a Coffee Leaderboard. The sysetm detected faces and if the person in the image was holding a coffee cup, the system would update the leaderboard. Using this as a starter for my idea gave me a headstart. I would have to manage some code, but the starter project had great examples on doing the things in which I had the least experience.

The Design

Cloud native designs to some might look like Rube Goldberg machines due to the number of components and interactions that are in play. As an solution architect I have to remind myself of the beauty of these systems in that they trade application simplicity for infrastructure complexity. These solutions do not disappoint. The composite, cloud-native design uses several AWS Services together to implement much of the solution and it requires relatively low amounts of code writing to bring it together.



AWS Services Used

- Simple Storage Service (S3) - used to store images captured by the AWS DeepLens
- Lambda - serverless code container that executes the application code
- Rekognition - performs the facial recognition function
- DynamoDB - stores information about identified faces
- Simple Notification Service (SNS) - notifies the music playing application that it should play a song.

How It Works

The entire process is kick-started when the AWS DeepLens detects a face in its camera. The DeepLens takes a picture, draws a border around the face and saves the edited image to an AWS S3 bucket.

When the image is saved in the S3 bucket, an AWS Lambda function is triggered. The purpose of the Lambda function is to grab the image from S3 and make a request to AWS Rekognition to identify the face in the image and to determine if that face is known to this system. It does this by executing a small Python application. The Python code passes the image to Rekognition and identifier is returned.

Rekognition is a power image and video analysis tool. It is a proven library built on reliable models and algorithms that "require no machine learning expertise to use." Using the tool is simple and requires one line of code.

```
faces = rekognition.search_faces_by_image(CollectionId={Rekogniton
Collection Name}, Image=[image object]],
                    FaceMatchThreshold={facial recognition error
threshold}, MaxFaces=1)
```

The Lambda function then queries the DynamoDB to see if that identifier already exists. If it already exists, then the person in the image has visited before. Their entrance song name is returned.

If the face does not exist already in the DynamoDB, it is registered and the default entrance song name is returned.

The Lambda creates and posts an SNS message that contains pertinant information that can be consumed by any application that subscribes to the message. In this instance a web application is polling for new messages in the SNS topic. When a new visitor is identified, then the web application will play the song associated with the user.

The Future

There are many application to technology like this. Of course there are many applications for facial recognition. The The OneEye Project has been adapted to do customer identification and for an Amber Alert use case.

There are also applications in retail, healthcare, and other industries. I am particularly interested in applications that make use of Rekognition's abilty to find and transcribe text in images and video and to idenfy objects.

Finally

If you are interested in applications in Machine Learning and Artifical Intelligence, there is a path for you to get started. There is a body of work that you can use as a starting point and reasonably priced hardware that you can use to interact with the world. Once you get started you will start to see how these systems are composed. You will feel increasingly comfortable with doing more difficult and challenging things. Then you can branch out and maybe create something that will change the world!