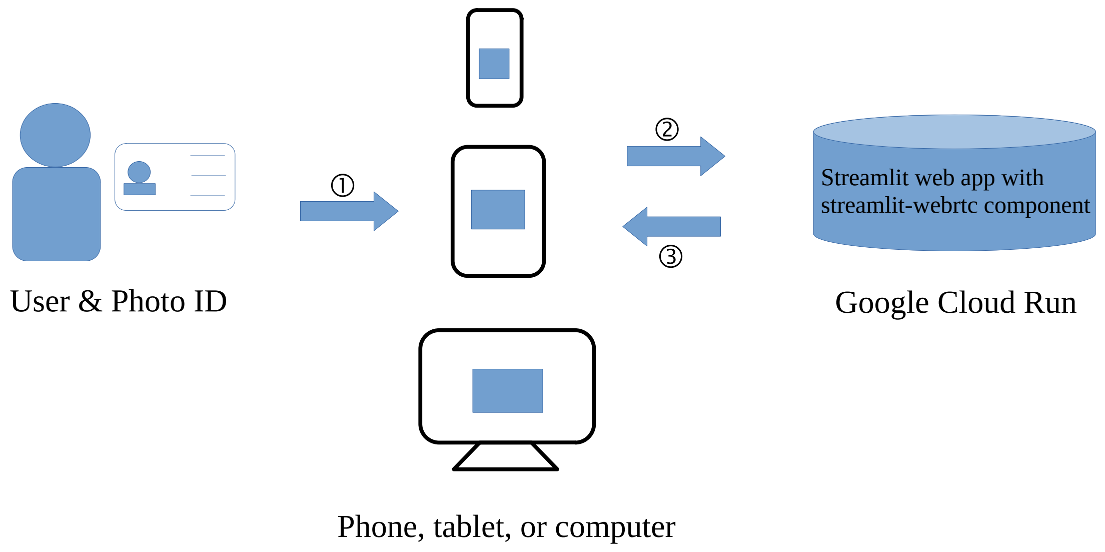
Deployment Architecture


User & Photo ID
①
②
③
Streamlit web app with streamlit-webrtc component
Google Cloud Run
Phone, tablet, or computer

① Input: A user holds his/her ID photo next to the face and stands in front of the camera. His/her face and the face in the ID photo are captured as two images.
② Request: The request is sent to the application stored in Google Cloud Run. The two images are processed in the siamese model.
③ Response: If the two images are verified as the same person, the screen displays "Verified" on top of the screen and green rectangles around the faces. If they are not verified, the screen displays "Not Verified" and red rectangles around the faces.

I originally deployed a Flask web application. However, it did not connect the web camera when the application was stored in Google Cloud. To make the web camera work, I changed it to a Streamlit web application. It connects the web camera, but the video streaming is not working. A further research is required to make the webrtc function work.

Checklist

1. Define the problem

The airline company I frequently use seems to have a human resource issue lately. The inefficiency of ID check before the passengers get on board is causing congestion in the waiting area and sometimes delaying the flight. Automating the ID check may solve a shortage of human resources and congestion in the airport.

2. Prepare the data

I downloaded face photos from Kaggle and other online resources.
375 positive and 375 negative pairs (750 in total) prepared for training and testing
115 positive and 115 negative pairs (230 in total) prepared for validation
The datasets includes females/males and black/white/Asian people relatively equally
The training and testing datasets randomly selected from 750 pairs (training: 600 and testing: 150)

3. Spot check algorithms

I used a convolutional neural network in a siamese network. For a convolutional neural network, I chose a pre-trained VGG16 model among others, such as ResNet50 and Xception, per recommendation.

4. Improve Results

I tried hyperparameter tuning--different epochs, activation functions, dropout rates, number of dense layers, number of layers to be re-trained, and input image sizes--to improve the recall and precision rates. Then I selected the hyperparameters that showed a balance between two rates.

5. Finalize project

I created a Streamlit web application and deployed it in Google Cloud Run (https://capstonertc-dzvisiwhya-wl.a.run.app).

This application compares the user’s face with his/her ID photo. If they are identified as the same person, the screen displays “Verified” and green rectangles around the faces. If they are not identified as the same person, the screen displays “Not Verified” and red rectangles around the faces.

Further improvements need to be made since the deployed application does not work as the one running in the local host, and the performance of the ML model is not high enough.