

Facial Verification System



Hilquias de Paiva Araújo

FaceV Company

PO-235



Contents

1. What is Face Verification?
 2. Use case
 3. Our Proposal
 4. FaceV Differential
 5. Implementation
 6. Next steps
-
- A. Appendix

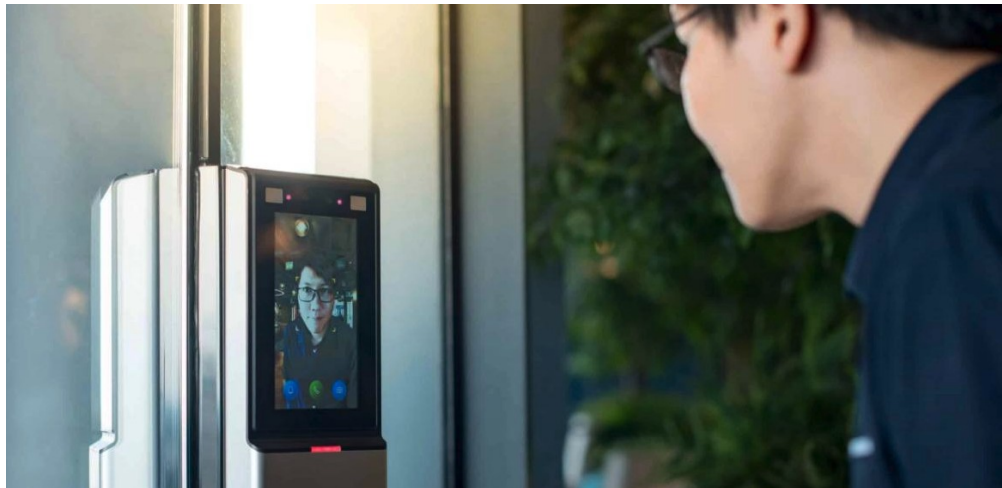
1. What is Face Verification?

- Biometric technology that verifies individuals by analyzing and comparing patterns based on facial contours and features.
- It has gained widespread popularity due to its accuracy, speed, and versatility.
 - Security: Access control for buildings, airports, and secure areas.
 - Law Enforcement: Criminal identification and surveillance.
 - Banking: Secure authentication for online transactions.



2. Use case

- Many commercial and residential buildings use facial recognition as the main tool of access control
- A motion sensor triggers the facial recognition camera and the image captured starts to be streamed in the display
- The image captured is then “compared” to the images database in order to identify who is trying to access the building
- A message is displayed in the streaming: “Welcome {person_name}” or “Not identified”



3. Our Proposal

- Provide a model that performs access control of residential buildings:
 - Verify registered people (residents, registered workers, etc.)
 - Deny access to unregistered people (visitors, unregistered service providers, etc.)
- Provide API that serves the face verification model
 - API connects to our database
 - Extract latest info about the model
 - Run inference (face verification predictions)
 - Provide feedback option for human operator



3. Our Proposal

- Provide database:
 - Model configuration
 - Prediction feedbacks
 - All model predictions (next version)
 - Model performance observability (next version)
- Our market differential:
 - We provide a custom model that follows client expectations *
 - The client is able to collect human feedback for each wrong model prediction, so the next version can improve



4. FaceV Differential

1. Human feedback
 - a. With human feedback collected through time, we enhance model capabilities in every model selection iteration
2. Custom model selection
 - a. We decide, with the client, the best weight for the model's error ("false positive", "false negative")
 - b. With that weight defined, we run our custom experiment that selects the best model (with the highest probability of minimizing errors in production)



5. Implementation

1. Client provides database with every registered resident and worker with:
 - One picture for each one
 - ID info: name, CPF, date of birth
2. We and the client decide the best weight of model errors, according to client necessity
3. Run model selection experiment with database and the error weight established
4. Grant access to our API with the best model:
 - Predict
 - Feedback



6. Next Steps

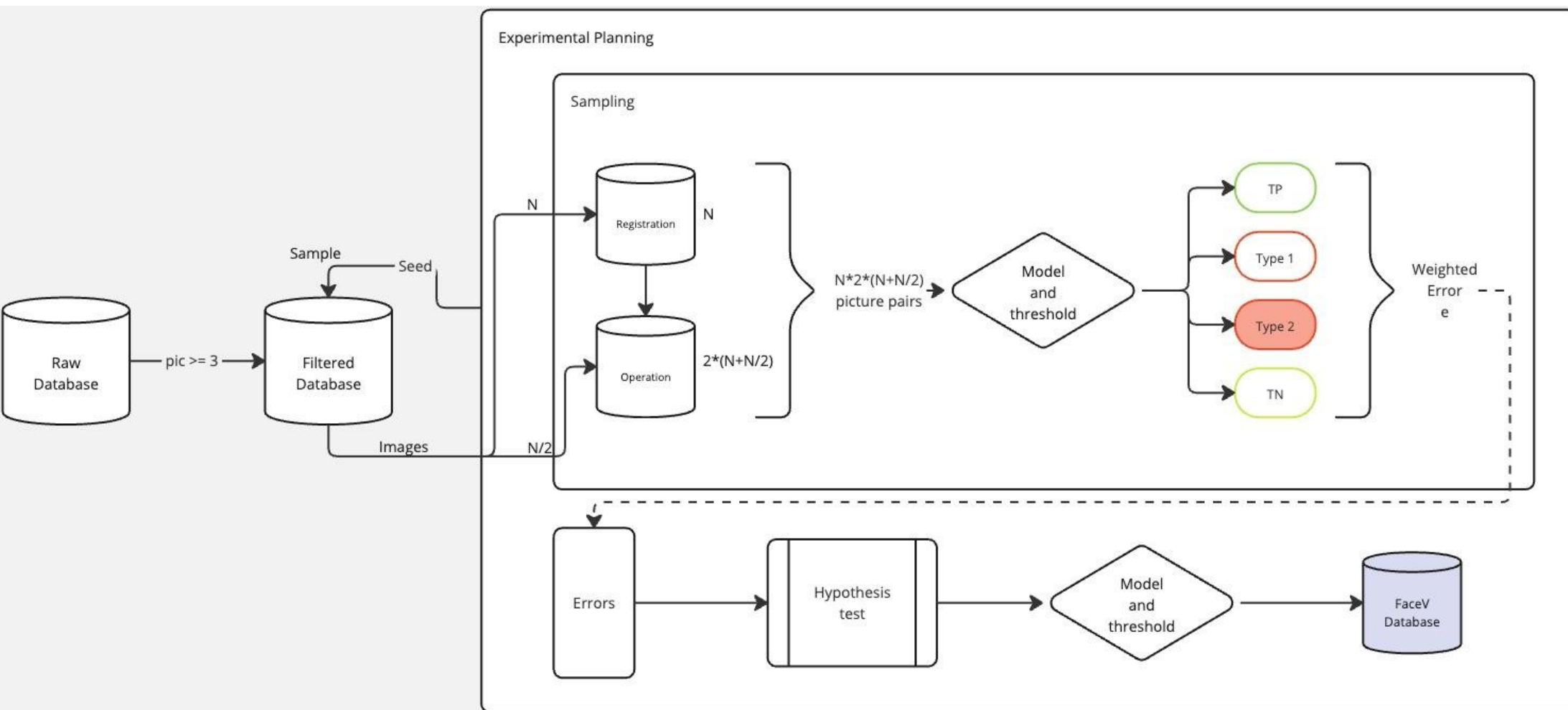
1. Direct to commercial and sales support
2. Define any new features, as client may request
 - a. Model observability
 - b. Model re-training
3. Client send the registered database
4. Define type 2 error weight
5. Run model selection experiment
6. Serve API



Thank you!

FaceV Company

A.1. Experimental Planning

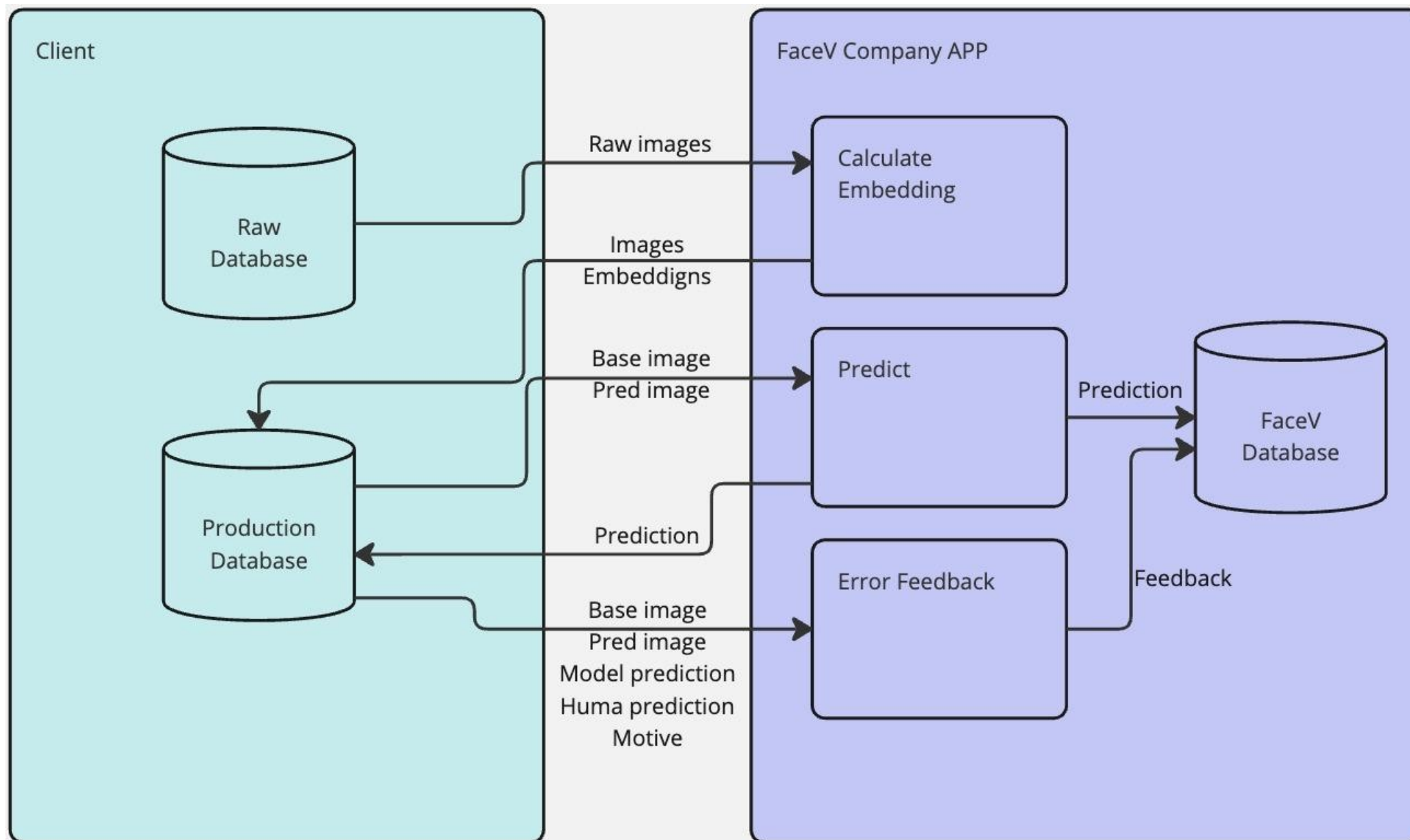


A.2. Error metric

- Type 1: “false positive”; Type 2: “false negative”
- A discussion with the client is needed to understand how bad a type 2 error would be, and then we decide our type 2 error weight (ω)
- A model that does not correctly predicts TP will have greater loss
- A model that does commit errors will have greater loss
- A good model will have small loss, being able to correctly predict TP as well as being able to minimize both errors

$$\mathcal{L} = \frac{type_1 + \omega \cdot type_2}{TP + \epsilon}$$

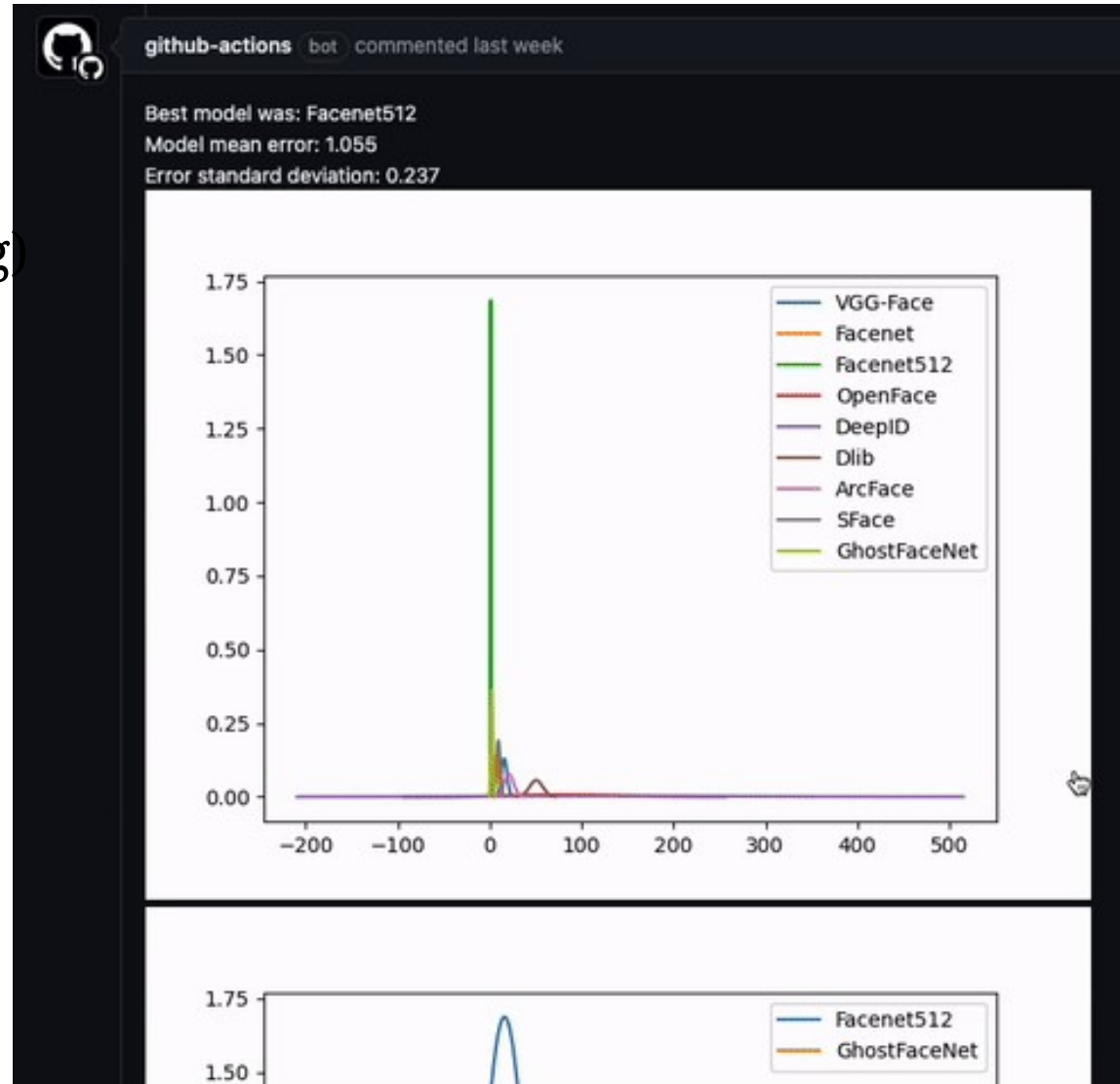
A.3. Production Architecture



A.4. CI/CD

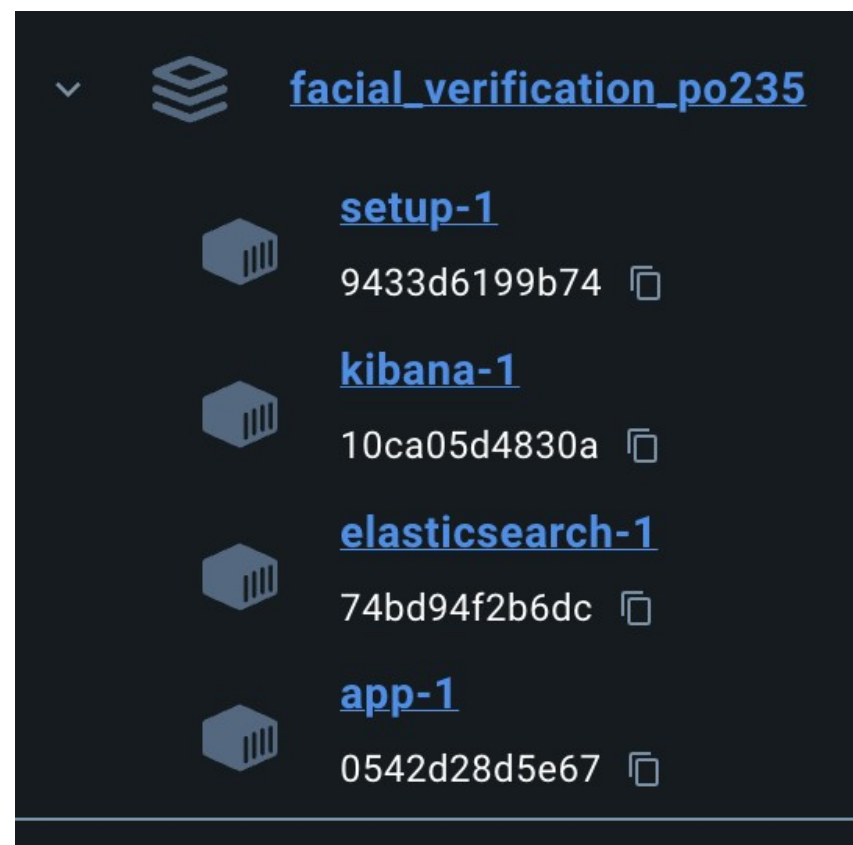


- Github Actions
- CML (Continuous Machine Learning)
- Custom model report
- Error plot (all models)
- Error plot (champion vs challenger)



A.5. Docker

- Setup (elasticsearch and kibana)
- Elasticsearch: FaceV Database
- Kibana: observability tool
- App: API application



A.6. API



Face Verification 0.1.0 OAS 3.1

/openapi.json

default

GET

/ Home

POST

/predict Predict

POST

/feedback Error Feedback

POST

/calculate_embedding Calculate Embedding