Detect, analyze, and recognize faces

1. Introduction

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/1-introduction

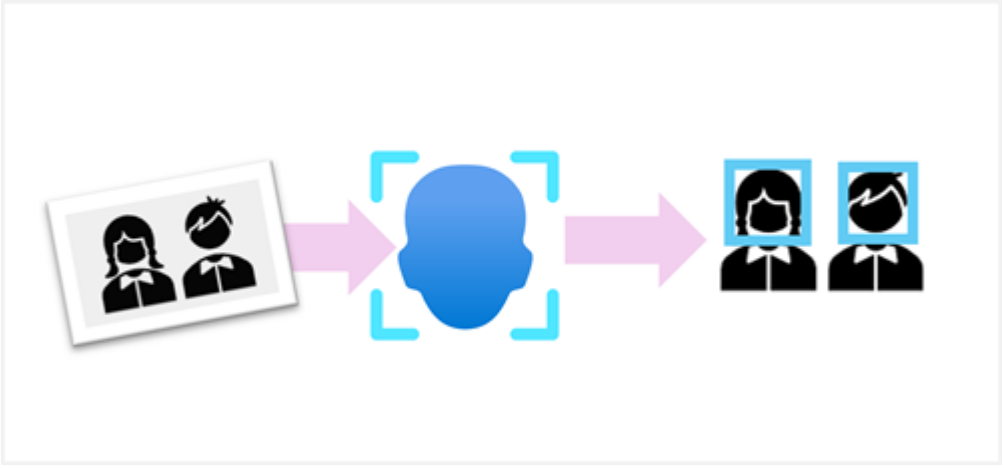
Introduction

Completed

- 1 minute

Face detection, analysis, and recognition are all common computer vision challenges for AI systems. The ability to detect when a person is present, analyze a person's facial features, or recognize an individual based on their face is a key way in which AI systems can exhibit human-like behavior and build empathy with users.

In this module, you'll explore how the Azure Vision *Face API* enables you to build solutions that analyze faces in images.



Note

Access to the full capabilities of the Face API is restricted in accordance with Microsoft's responsible AI policies. For details, see **Limited Access to Face API**. This module describes some

capabilities that require explicit access. The practical exercise in the module is based on unrestricted features of the service.

2. Plan a face detection, analysis, or recognition solution

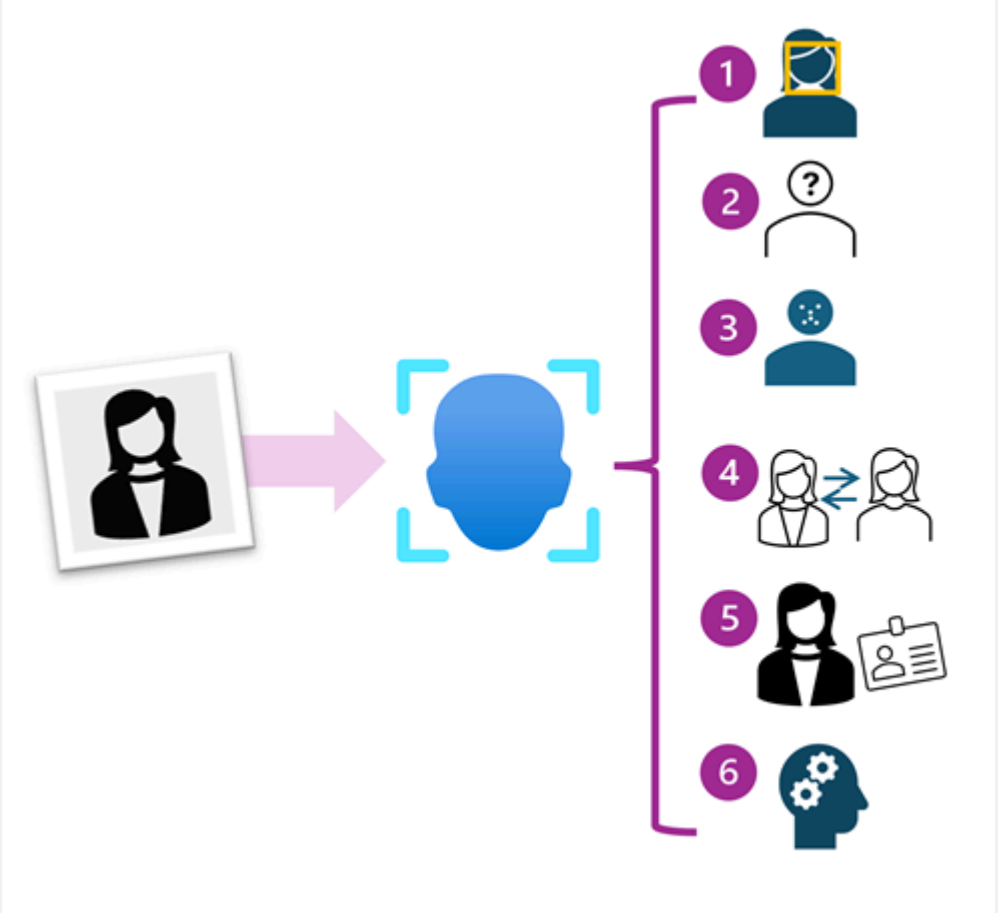
https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/2-understand-capabilities-of-face-service

Plan a face detection, analysis, or recognition solution

Completed

- 5 minutes

The **Face** service provides comprehensive facial detection, analysis, and recognition capabilities.



The Face service provides functionality that you can use for:

1. *Face detection* - for each detected face, the results include an ID that identifies the face and the bounding box coordinates indicating its location in the image.
2. *Face attribute analysis* - you can return a wide range of facial attributes, including:
    - Head pose (*pitch*, *roll*, and *yaw* orientation in 3D space)
    - Glasses (*No glasses*, *Reading glasses*, *Sunglasses*, or *Swimming Goggles*)
    - Mask (the presence of a face mask)
    - Blur (*low*, *medium*, or *high*)
    - Exposure (*under exposure*, *good exposure*, or *over exposure*)
    - Noise (visual noise in the image)
    - Occlusion (objects obscuring the face)
    - Accessories (glasses, headwear, mask)
    - QualityForRecognition (*low*, *medium*, or *high*)
3. *Facial landmark location* - coordinates for key landmarks in relation to facial features (for example, eye corners, pupils, tip of nose, and so on)
4. *Face comparison* - you can compare faces across multiple images for similarity (to find individuals with similar facial features) and verification (to determine that a face in one image is the same person as a face in another image)
5. *Facial recognition* - you can train a model with a collection of faces belonging to specific individuals, and use the model to identify those people in new images.
6. *Facial liveness* - liveness can be used to determine if the input video is a real stream or a fake to prevent bad-intentioned individuals from spoofing a facial recognition system.

Face detection and recognition models

The Azure Vision Face API is built on face detection and recognition models that have been pre-trained. Multiple versions of these models are available, each with specific strengths and capabilities. For example, newer models exhibit greater accuracy when working with small images; but may not provide the same breadth of facial analysis capabilities. When you use the service in an application, you must select the model you want to use based on your requirements.

> **Tip**
>
> For guidance about selecting a *detection* model, see **Specify a face detection model**. For guidance about how to select a *recognition* model, see **Specify a face recognition model**.

3. Detect and analyze faces

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/3-detect-analyze-faces

Detect and analyze faces

Completed

- 5 minutes

To use the Azure Vision Face API, you must provision a resource for the service in an Azure subscription. You can provision **Face** as a single-service resource, or you can use the Face API in a multi-service **Foundry Tools** resource; which can be provisioned as a standalone resource or as part of A Microsoft Foundry project.

To use your resource from a client application you must connect to its *endpoint* using either *key-based authentication* or *Microsoft Entra AI authentication*. When using the REST interface you can provide the authentication key or token in the request header. When using a language-specific SDK (for example, the Python **azure-ai-vision-face** package or the Microsoft .NET **Azure.AI.Vision.Face** package), you use a **FaceClient** object to connect to the service.

```
 from azure.ai.vision.face import FaceClient
from azure.ai.vision.face.models import *
from azure.core.credentials import AzureKeyCredential

face_client = FaceClient(
    endpoint="<YOUR_RESOURCE_ENDPOINT>",
    credential=AzureKeyCredential("<YOUR_RESOURCE_KEY>"))
```

```
 using Azure;
using Azure.AI.Vision.Face;

FaceClient faceClient = new FaceClient(
    new Uri("<YOUR_RESOURCE_ENDPOINT>"),
    new AzureKeyCredential("<YOUR_RESOURCE_KEY>"));
```

To detect and analyze faces in an image, you must specify the model-specific features you want the service to return, and then use the client to call the **Detect** method.

```
 # Specify facial features to be retrieved
features = [FaceAttributeTypeDetection01.HEAD_POSE,
            FaceAttributeTypeDetection01.OCCLUSION,
            FaceAttributeTypeDetection01.ACCESSORIES]

# Use client to detect faces in an image
with open("<IMAGE_FILE_PATH>", mode="rb") as image_data:
    detected_faces = face_client.detect(
        image_content=image_data.read(),
        detection_model=FaceDetectionModel.DETECTION01,
        recognition_model=FaceRecognitionModel.RECOGNITION01,
        return_face_id=True,
        return_face_attributes=features,
    )
```

```
 // Specify facial features to be retrieved
FaceAttributeType[] features = new FaceAttributeType[]
{
    FaceAttributeType.Detection01.HeadPose,
    FaceAttributeType.Detection01.Occlusion,
    FaceAttributeType.Detection01.Accessories
};

// Use client to detect faces in an image
using (var imageData = File.OpenRead(imageFile))
{
    var response = await faceClient.DetectAsync(
        BinaryData.FromStream(imageData),
        FaceDetectionModel.Detection01,
        FaceRecognitionModel.Recognition01,
        returnFaceId: false,
        returnFaceAttributes: features);
    IReadOnlyList<FaceDetectionResult> detected_faces = response.Value;
}
```

The response from the service depends on:

- The model-specific features requested.
- The number of faces detected in the image.

A response for an image containing a single face might look similar to the following example:

```
 [
    {
        'faceRectangle': {'top': 174, 'left': 247, 'width': 246, 'height': 246}
        'faceAttributes':
        {
            'headPose':{'pitch': 3.7, 'roll': -7.7, 'yaw': -20.9},
```

```
            'accessories':
                [
                    {'type': 'glasses', 'confidence': 1.0}
                ],
            'occlusion':{'foreheadOccluded': False, 'eyeOccluded': False, 'mouthOccluded': False}
        }
    }
]
```

4. Verify and identify faces

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/4-compare-match-detected-faces

Verify and identify faces

Completed

- 6 minutes

In addition to detecting and analyzing faces, you can use the Azure Vision Face service to compare and recognize faces.

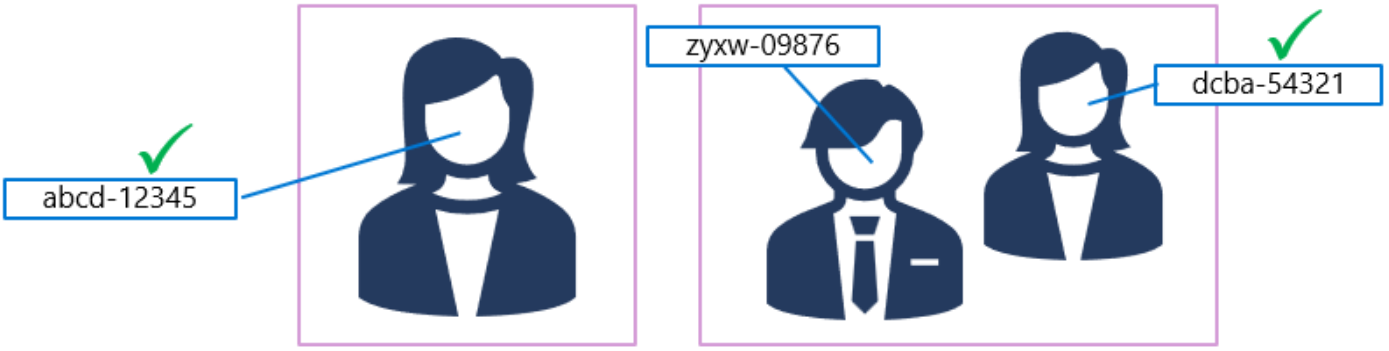
Important

Usage of facial recognition, comparison, and verification requires approval through a **Limited Access policy**.

Verifying faces

When a face is detected by the Face service, a unique ID is assigned to it and retained in the service resource for 24 hours. The ID is a GUID, with no indication of the individual's identity other than their facial features.

While the detected face ID is cached, subsequent images can be used to compare the new faces to the cached identity and determine if they're *similar* (in other words, they share similar facial features) or to *verify* that the same person appears in two images.



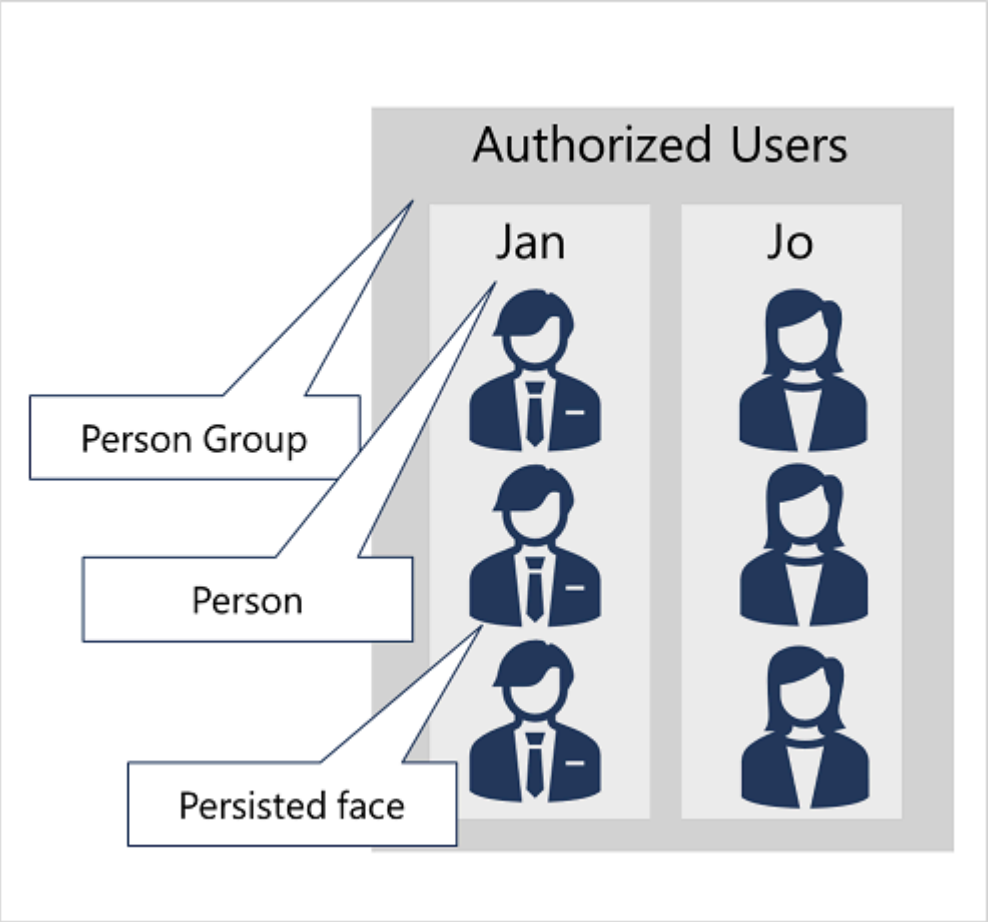
This ability to compare faces anonymously can be useful in systems where it's important to confirm that the same person is present on two occasions, without the need to know the actual identity of the person. For example, by taking images of people as they enter and leave a secured space to verify that everyone who entered leaves.

Identifying faces

For scenarios where you need to positively identify individuals, you can train a facial recognition model using face images.

To train a facial recognition model with the Face service:

1. Create a **Person Group** that defines the set of individuals you want to identify (for example, *employees*).
2. Add a **Person** to the **Person Group** for each individual you want to identify.
3. Add detected faces from multiple images to each **person**, preferably in various poses. The IDs of these faces will no longer expire after 24 hours (so they're now referred to as *persisted* faces).
4. Train the model.



The trained model is stored in your Face (or Foundry Tools) resource, and can be used by client applications to:

- *Identify* individuals in images.
- *Verify* the identity of a detected face.
- Analyze new images to find faces that are *similar* to a known, persisted face.

> **Tip**
>
> To learn more about using face verification and identification to implement a facial recognition solution, see **Face recognition** in the Azure Vision Face documentation.

5. Responsible AI considerations for face-based solutions

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/5-understand-considerations-for-face-analysis

Responsible AI considerations for face-based solutions

Completed

- 3 minutes

While all applications of artificial intelligence require considerations for responsible, system that rely on facial or other biometric data can be particularly problematic.

When building a solution that uses facial data, considerations include (but aren't limited to):

- **Data privacy and security**. Facial data is personally identifiable, and should be considered sensitive and private. You should ensure that you have implemented adequate protection for facial data used for model training and inferencing.
- **Transparency**. Ensure that users are informed about how their facial data is used, and who will have access to it.
- **Fairness and inclusiveness**. Ensure that your face-based system can't be used in a manner that is prejudicial to individuals based on their appearance, or to unfairly target individuals.

> **Note**
>
> You can find details of the responsible AI measures taken in the implementation of the Face API in **Data and privacy for Face**.

6. Exercise - Detect and analyze faces

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/8-exercise-identify-faces

Exercise - Detect and analyze faces

Completed

- 30 minutes

Now it's your turn to try using the Azure Vision Face service.

In this exercise, you use the Azure Vision Face API to develop a client application that detects and analyzes faces in an image.

> **Note**
>
> To complete this lab, you need an **Azure subscription** in which you have administrative access.

Launch the exercise and follow the instructions.

Launch Exercise

> **Tip**
>
> After completing the exercise, if you've finished exploring Foundry Tools, delete the Azure resources that you created during the exercise.

7. Module assessment

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/9-knowledge-check

Module assessment

Completed

- 3 minutes

8. Summary

https://learn.microsoft.com/en-us/training/modules/detect-analyze-recognize-faces/10-summary

Summary

Completed

- 1 minute

In this module, you have learned how to detect, analyze, and recognize faces. Facial analysis and recognition solutions can help you address business requirements for security and authorization, digital asset management, and other scenarios. However, you should be mindful of the need to use biometric data such as facial imaging responsibly.

> **Tip**
>
> To find out more about the Azure Vision Face service, see the **Face documentation**.