**Goal:**

Convert a **facial image (e.g., 224x224x3 RGB)** into a **128-dimension vector** that:

* Represents **who** the person is (identity),
* But not **what** the person looks like (privacy-preserving).

**Step-by-Step: Input → Conv Layers → Dense Layers → Signature**

**Let’s say the input is:**

* A photo of **Person A** at a teller window (e.g., 224x224 pixels RGB face)

**[Stage 1] — Convolutional Layers (Feature Extraction)**

Each layer **extracts features**, reduces dimensions, and increases abstraction.

**🔹 Conv Layer 1:**

* Filters: 64
* Input size: 224x224x3
* Output size: 112x112x64 (after pooling)
* **Captures:** edges, corners, basic textures

**🔹 Conv Layer 2:**

* Filters: 128
* Input: 112x112x64
* Output: 56x56x128
* **Captures:** eyes, nose, mouth shapes

**🔹 Conv Layer 3:**

* Filters: 256
* Output: 28x28x256
* **Captures:** spatial relationships between facial features

**🔹 Conv Layer 4:**

* Filters: 512
* Output: 14x14x512
* **Captures:** skin tone, facial symmetry, expression dynamics

**🔹Conv Layer 5:**

* Filters: 512
* Output: 7x7x512
* **Captures:** high-level face identity representations
* **Flatten this layer**: 7x7x512 = 25,088 features

**[Stage 2] — Dense Layers (Compression to Identity Vector)**

You now have a **very high-dimensional feature vector** (25k+ values).  
Now let’s compress it while preserving identity:

**Dense Layer 1:**

* Input: 25,088
* Output: 1,024 neurons
* **Purpose:** Reduce size, keep most salient traits
* **Activation:** ReLU

**Dense Layer 2:**

* Input: 1,024
* Output: 256 neurons
* **Purpose:** Identity trait condensation
* **Activation:** ReLU

**Dense Layer 3 (Final Signature Layer):**

* Input: 256
* Output: **128 neurons**
* **Activation:** **Sigmoid** (outputs values between 0 and 1)

🧠 This final layer is your **identity signature vector**, e.g.:

[0.21, 0.88, 0.02, ..., 0.73] ← length = 128

No image can be recreated from this. Yet, this vector is **unique to Person A** and **repeatable** from future visits — enabling matching.

**Privacy Advantage:**

* All the **reconstructive visual features** were discarded during compression.
* The network learns to **keep what’s needed for identity** but **throws away appearance data** — thanks to adversarial training with the discriminator.

**Example in Plain Language:**

Think of it like a factory that takes a passport photo, reads all its features (eyes, nose, skin tone), encodes them into a **secret 128-digit passcode**, and destroys the image. The next time Person A comes, we get a new image, generate a new code, and compare the two codes — no photo ever stored, no face ever recreated.

"Our 5-convolutional and 3-dense layer pipeline extracts just enough information to know who the customer is—but throws away everything else. It’s like boiling a face down into a private signature that only our system can understand."

What is sigmoid layer ?

A **sigmoid layer** is a neural network layer that applies the **sigmoid activation function** to its outputs.

**🔍 What is the Sigmoid Function?**

The **sigmoid function** takes a real number input and "squashes" it into a value **between 0 and 1**:

sigmoid(x)=11+e−x\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}sigmoid(x)=1+e−x1​

**Here's how it behaves:**

* Large negative numbers (e.g., -10) → **output ≈ 0**
* Zero → **output = 0.5**
* Large positive numbers (e.g., +10) → **output ≈ 1**

**Why Use a Sigmoid Layer?**

In your context (128-dimensional identity vector generation):

* It **normalizes** output between 0 and 1
* Each output neuron gives a **probability-like score**
* It ensures the signature is **bounded**, making:
  + Matching easier (cosine similarity, Euclidean distance)
  + Storage and encryption more consistent
  + Inputs less sensitive to outliers

**Example:**

Imagine the output of your last dense layer is:

[−3.2,0.8,2.5,−1.0][-3.2, 0.8, 2.5, -1.0][−3.2,0.8,2.5,−1.0]

Applying sigmoid:

[0.039,0.69,0.92,0.27][0.039, 0.69, 0.92, 0.27][0.039,0.69,0.92,0.27]

This becomes your **final signature vector** (privacy-safe, normalized, matchable).

**How It Helps in Your System**

* In the **generator's last layer**, sigmoid ensures each component of the 128D vector is:
  + Within [0, 1]
  + Comparable to stored signatures
  + Easier to encrypt and dynamically rotate

**Council Explanation:**

"We apply a sigmoid layer at the end to ensure the output facial signature is always bounded between 0 and 1. This makes it stable, comparable, secure, and suitable for real-time matching without ever needing to store or reconstruct images."