

Analisi Comparativa di Riduzione Dimensionale

Metodi Lineari vs Non-Lineari per
Face Recognition e Verification

Roberto Carriero • Massimiliano Leone

Laurea Magistrale in Ingegneria Informatica – Artificial Intelligence and Data Science

A.A. 2025/2026

Struttura della Presentazione

7 Capitoli · Analisi comparativa

1 Introduzione e Motivazioni

Contesto applicativo, curse of dimensionality, obiettivi del progetto

3 Dataset e Preprocessing

LFW, analisi sbilanciamento, pipeline preprocessing, mean face

5 Risultati: Face Recognition

Protocollo sperimentale, risultati quantitativi, ROC curves, confusion matrix

OBIETTIVO

Confronto rigoroso tra **PCA (lineare)** e **Autoencoder (non-lineare)** per riduzione dimensionale nel riconoscimento facciale

2 Framework Matematico

PCA, SVD, Autoencoder, SVM, Neural Network, metriche di similarità

4 Esperimenti: Riduzione Dimensionale

Ablation study PCA, eigenfaces, training AE, confronto ricostruzione

6 Risultati: Face Verification

Setup verifica, cosine vs euclidean, distribuzione similarità, heatmap

Applicazioni

Sicurezza

Controllo accessi, sorveglianza, autenticazione biometrica

Mobile

Sblocco smartphone (Face ID), app di autenticazione

Embedded Systems

Dispositivi a bassa potenza, IoT, edge computing

Requisiti Chiave

1 Modelli leggeri e veloci

Inferenza in tempo reale su dispositivi con risorse limitate

2 Riduzione dimensionale

Compressione feature per deployment efficiente

3 Alta accuratezza

Precisione nel riconoscimento anche con variazioni

Motivazione: Perché la Riduzione Dimensionale?



Deployment Edge

Dispositivi con memoria limitata



Minore Latenza

Inferenza rapida in tempo reale



Consumo Energetico

Meno calcoli = meno energia



Storage Ridotto

Database più compatti

Il Problema: Curse of Dimensionality

Sfida matematica nell'high-dimensional space

⚠ Sfida Matematica

Esempio Concreto

Immagine **64 × 47 pixel** ⇒ **D = 3,008** dimensioni

Ogni immagine è un punto in uno spazio 3,008-dimensionale

→ Spazio delle feature intrinsecamente sparso

La maggior parte dello spazio è vuoto, i dati occupano una frazione infinitesimale

→ Volume concentrato sui bordi dell'iper cubo

In alta dimensione, il volume si concentra nelle regioni estreme

→ Dati necessari crescono esponenzialmente

Per coprire lo spazio servono $O(ed)$ campioni

Curse of Dimensionality

1. Distanze perdono significato

Tutti i punti diventano «equidistanti»

2. Classificatori perdono potere

Overfitting più probabile

3. Distanze non discriminanti

Impossibile distinguere similarità



Soluzione: Riduzione Dimensionale

Proiettare i dati in uno spazio di dimensione inferiore preservando l'informazione rilevante

1 Riduzione Dimensionale

PCA (SVD) vs Autoencoder

Confronto tra approccio lineare e non-lineare

↳ Analisi varianza spiegata e MSE ricostruzione

2 Classificazione

SVM vs Neural Network

Valutazione di classificatori lineari e non-lineari su feature ridotte

✖ Accuracy, F1-Score, ROC-AUC

3 Face Verification

Cosine vs Euclidean Similarity

Confronto tra metriche di similarità per task di verifica 1:1

⚖️ EER, AUC, TAR, FAR

4 Ablation Study

Analisi Componenti e Parametri

Variazioni al numero di componenti, architettura, iperparametri

🔍 Grid search e cross-validation

Domanda

Un metodo **non-lineare (Autoencoder)** supera sempre il **lineare (PCA)**, oppure esistono condizioni in cui la **semplicità vince**?

Task 1: Face Recognition (Matching 1:N)

Identificazione dell'identità tra N persone

Formulazione del Problema

Input

Immagine $X \in \mathbb{R}^D$

Output

Identità $\hat{y} \in \{1, \dots, K\}$

Decisione Ottima (MAP)

$$\hat{y} = \operatorname{argmax}_k P(Y = k | X = x)$$

Metriche di Valutazione

1 Accuracy

Predizione corrette vs numero totale di esempi osservati

2 F1-Score Weighted

Media pesata per gestire sbilanciamento classi

3 ROC-AUC

Multi-class One-vs-Rest con micro/macro averaging

4 Confusion Matrix

Analisi dettagliata errori per classe



Scenario

Database con **K identità**, immagine in input riconosciuta tra **N possibili**

Dataset

K = 7 classi

Task 2: Face Verification (Matching 1:1)

Verifica se due volti appartengono alla stessa persona

➡ Formulazione del Problema

Input

Coppia di immagini (x_1, x_2)

Output

Decisione binaria $d \in \{\text{Genuine, Impostor}\}$

Regola di Decisione

$d = \text{Genuine} \text{ se } S(x_1, x_2) \geq \tau$

$d = \text{Impostor} \text{ altrimenti}$

↖ Metriche Specifiche

⚖️ EER (Equal Error Rate)

Punto in cui FAR = FRR

Soglia ottima τ^* dove i due errori si equivalgono

📈 AUC ROC

Area Under Curve della curva ROC di verifica

⚙️ TAR, FAR

True Accept Rate a FAR fissato (es. TAR@FAR=0.1%)



Genuine Pair

Stessa identità → Similarità alta

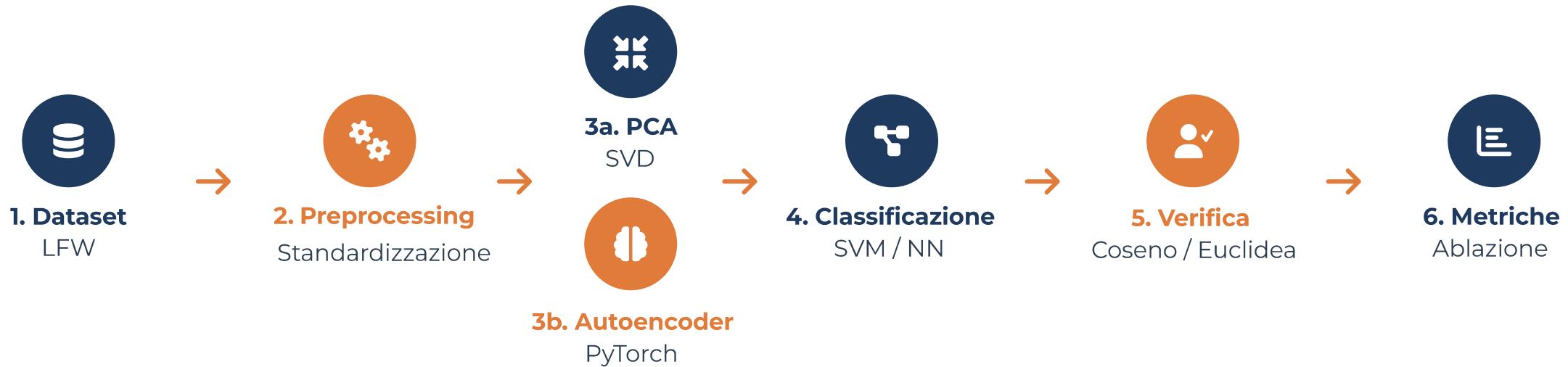


Impostor Pair

Identità diverse → Similarità bassa

Pipeline del Progetto

Flusso completo dall'acquisizione dati alla valutazione



Tecnologie Utilizzate

Python 3.14

NumPy, Pandas, Scikit-Learn, Matplotlib, PyTorch

SVD from scratch

Non PCA da Sci-kit Learn

Autoencoder

Fully-connected neural network con PyTorch

Grid Search / Cross Validation

5 Fold

💡 Definizione

Dato un dataset $X \in \mathbb{R}^{NxD}$, trovare una rappresentazione $Z \in \mathbb{R}^{Nxk}$ con $k \ll D$ che preservi l'informazione rilevante.

⤵ Approccio Lineare (PCA)

$$Z = XW, W \in \mathbb{R}^{D \times k}$$

- ✓ **Trasformazione affine** - Proiezione lineare su sottospazio
- ✓ **Preserva varianza** - Massimizza informazione mantenuta
- ✓ **Soluzione analitica** - Chiusa e deterministica

⤶ Approccio Non-Lineare (AE)

$$Z = f_{\theta}(X), f: \mathbb{R}^D \rightarrow \mathbb{R}^k$$

- ✓ **Trasformazione non-lineare** - Funzioni di attivazione
- ✓ **Apprende manifold curvi** - Strutture non-lineari
- ✓ **Soluzione iterativa** - Gradient descent

Massimizzazione della varianza proiettata

◎ Massimizzazione della Varianza

Si cercano le direzioni nello spazio che massimizzino la varianza dei dati proiettati

Si utilizza la matrice di covarianza dei dati centrati

$$\Sigma = (1/(N-1)) \mathbf{X} \mathbf{c}_T \mathbf{X} \mathbf{c}$$

◆ Soluzione

Gli **autovettori** di Σ corrispondenti ai **k autovalori più grandi**:

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i, \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$$

 Gli autovettori \mathbf{v}_i sono le **componenti principali**

★ Proprietà

Problema Convesso

Soluzione **globalmente ottima e unica**

Ortogonalità

Le componenti sono tra loro ortogonali: $\mathbf{v}_i \perp \mathbf{v}_j$

Ordinamento

λ_1 cattura la massima varianza, λ_2 la seconda, etc.

Singular Value Decomposition (SVD)

Decomposizione matriciale per PCA efficiente

⟳ Teorema SVD

Ogni matrice $X \in \mathbb{R}^{N \times D}$ può essere decomposta come:

$$X = U\Sigma V_T$$

U Matrice Ortogonale

$$U \in \mathbb{R}^{N \times N}$$

Vettori singolari sinistri

Colonne = autovettori di $X X_T$

Σ Matrice Diagonale

$$\Sigma \in \mathbb{R}^{N \times D}$$

Valori singolari σ_i

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(N,D)} \geq 0$$

V Matrice Ortogonale

$$V \in \mathbb{R}^{D \times D}$$

Vettori singolari destri

Colonne = autovettori di $X_T X$

⟳ Relazione con Autovalori

$$\sigma_i = \sqrt{\lambda_i(N-1)}$$
 dove λ_i è l'autovalore di Σ (matrice di covarianza)

Implementazione PCA via SVD

Algoritmo implementato da zero

</> Algoritmo Implementato

1 Centra i dati

$$X_c = X - \mu \text{ dove } \mu = (1/N) \sum_i x_i$$

2 Calcola SVD

$$X_c = U \Sigma V^T$$

3 Seleziona V_k

Prime **k colonne** di V : $V_k \in \mathbb{R}^{D \times k}$

4 Proietta

$$Z = X_c V_k$$

⇄ Trasformazioni

↓ Proiezione

$$Z = X_c V_k \in \mathbb{R}^{N \times k}$$

Da D dimensioni a k dimensioni

↑ Ricostruzione

$$\hat{X} = Z V_k^T + \mu$$

Torna allo spazio originale



Vantaggi SVD rispetto a Eigendecomposition

Stabilità numerica · Efficienza $O(ND \min(N,D))$ · Evita calcolo esplicito di Σ

Explained Variance Ratio

Quanta informazione preserviamo?

% Varianza Spiegata

Per ogni componente i:

$$\text{EVR}_i = \sigma_i^2 / \sum_j \sigma_j^2$$

- ⓘ **Frazione di varianza** catturata dalla componente i-esima
- ↗ Somma di tutti gli EVR = **1 (100%)**

Varianza Cumulativa

Con k componenti:

$$\text{CVR}(k) = \sum_{i=1}^k \text{EVR}_i$$

- ⓘ **Varianza totale** preservata con k componenti
- ⓘ Utile per scegliere il numero ottimale di componenti

Criterio Elbow

↗ Punto di **inflessione** dove aggiungere componenti porta **benefici marginali**. Identifica il trade-off ottimale tra compressione e informazione

Target

90-95% varianza

Autoencoder: Architettura

Rete neurale per apprendimento non-lineare

Definizione

Un autoencoder $A = (f_\theta, g_\varphi)$ è composto da:

Encoder f_θ

$$f_\theta: \mathbb{R}^D \rightarrow \mathbb{R}^k$$

$$z = f_\theta(x) = \sigma(W_e x + b_e)$$

- **Comprime** l'input in rappresentazione latente
- Parametri: W_e, b_e

Decoder g_φ

$$g_\varphi: \mathbb{R}^k \rightarrow \mathbb{R}^D$$

$$\hat{x} = g_\varphi(z) = \sigma(W_d z + b_d)$$

- **Ricostruisce** l'input dalla rappresentazione
- Parametri: W_d, b_d



Architettura Implementata

$D = 3008 \rightarrow 256 \rightarrow 256 \rightarrow k=100 \rightarrow 256 \rightarrow 256 \rightarrow D = 3008$

Autoencoder: Funzione di Costo

Ottimizzazione non-convessa con regolarizzazione

- Problema di Ottimizzazione (Non-Convesso)

Minimizzare l'errore di ricostruzione:

$$L(\theta, \varphi) = \frac{1}{N} \sum_i \|x(i) - g_{\varphi}(f_{\theta}(x(i)))\|_2^2 + \lambda \|\theta, \varphi\|_2^2$$

⚙️ Dettagli Implementativi

⚡ Attivazione

LeakyReLU ($\alpha = 0.2$)

⟳ Optimizer

Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)

_HEAP Batch Norm

Dopo ogni layer

🛡️ Regolarizzazione

✗ Dropout

$p = 0.1$

.Weight Decay

$\lambda = 10^{-4}$

✋ Early Stopping

patience = 10

Confronto Teorico: PCA vs Autoencoder

Lineare vs Non-lineare: analisi comparativa

Caratteristica	PCA (SVD)	Autoencoder
Natura	Lineare	Non-lineare
Soluzione	Analitica (closed-form)	Iterativa (gradient descent)
Ottimizzazione	Convessa	Non-convessa
Complessità	$O(ND \min(N,D))$	$O(E \cdot N \cdot \theta)$
Interpretabilità	Alta (eigenfaces)	Bassa (black-box)
Dati richiesti	Minori	Elevati
Capacità	Sottospazi lineari	Manifold non-lineari

X¹ Formulazione Primale

$$\begin{aligned} \min & (1/2) \|w\|^2 + C(1/N) \sum \xi_i \\ \text{s.t. } & y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i \end{aligned}$$

- ✓ **Problema convesso** - Soluzione globale unica
- ✓ **Efficace su piccoli dataset**
- ✓ **Supporto teorico solido**

O Kernel RBF

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

Hyperparameters (Grid Search)

C ∈ {0.1, 1, 10}

Kernel ∈ {RBF, Linear}

γ ∈ {scale, auto}



Limitazione

Scaling O(N²) – O(N³)

Costoso su dataset grandi

Classificatori: Neural Network

Rete neurale fully-connected con class weighting

Architettura

Input → [256] → [128] → [K]

- Batch Normalization - Stabilizza training
- Dropout ($p = 0.1 - 0.2$) - Previene overfitting
- ReLU activations - Non-linearità

Cross-Entropy Loss (Pesata)

$$L = -\sum w_i y_i \sum y_k \log(\hat{p}_k)$$

Pesi per sbilanciamento:

$$w_k = N / (K \cdot n_k)$$

Da più peso alle classi minoritarie

Hyperparameters (Grid Search)

Hidden
 $\{(128,64), (256,128)\}$

LR
 $\{10^{-2}, 10^{-3}\}$

Dropout
 $\{0.1, 0.2\}$

Epochs
 $\{50, 100\}$

Metriche di Similarità per Verifica

Cosine vs Euclidean: confronto teorico

» Cosine Similarity

$$\text{Scos}(u,v) = (u \cdot v) / (\|u\|_2 \cdot \|v\|_2)$$
$$\in [-1, 1]$$

- ✓ Invariante alla magnitudine - Robustness a scala
- ✓ Robusto a variazioni di illuminazione
- ✓ Cattura "direzione semantica"

» Distanza Euclidea

$$d_{\text{Eucl}}(u,v) = \|u - v\|_2$$

- ! Sensibile alla scala - Magnitudine influisce
- ✓ Interpretazione geometrica diretta
- i Distanza effettiva nello spazio



Relazione Matematica

Per vettori unitari: $\|u - v\|_2^2 = 2(1 - \text{Scos}(u,v))$

Dataset: Labeled Faces in the Wild (LFW)

Benchmark standard per face recognition in-the-wild

Caratteristiche

- 🏆 **Benchmark standard** per face recognition in-the-wild
- ☒ **Variabilità naturale:** posa, illuminazione, espressione
- 📷 **Condizioni non controllate** - Sfida realistica

Filtro Applicato

Solo identità con **≥ 70 immagini** per garantire supporto statistico adeguato

Nota

Distribuzione classi **sbilanciata** - richiede tecniche di gestione

Statistiche Finali

7

Classi

1,288

Immagini Totali

1,030

Train (80%)

258

Test (20%)

Analisi dello Sbilanciamento

Problema e soluzioni adottate

⚠ Problema

La classe "**George W. Bush**" domina il dataset con oltre il 40% delle immagini

Conseguenze

- **Bias** verso la classe maggioritaria
- **Recall bassa** per classi minoritarie
- **Metriche aggregate fuorvianti**

🔧 Soluzioni Adottate

☒ Stratified Split

Mantiene **proporzioni** tra train e test

⚖ Class Weighting

$$w_k = N / (K \cdot n_k)$$

Ἑ F1-Weighted

Metrica **robusta** allo sbilanciamento

Distribuzione Classi

Bush
530

Powell
236

Blair
144

Rumsfeld
121

Schroeder
109

Sharon
77

Chavez
71

Pipeline di Preprocessing

Dall'acquisizione alla standardizzazione



Standardizzazione (Z-score)

$$Z_i = (X_i - \mu) / \sigma$$

Motivazioni

- ✓ Stabilità numerica SVD
- ✓ Convergenza NN più rapida
- ✓ Comparabilità feature

Stratified Split

$$n_{train_k} / N_{train} \approx n_{test_k} / N_{test}$$

Importanza

- ❗ Cruciale per dataset **sbilanciati**
- 🛡 Evita che classi rare siano **assenti nel test**
- ⚖️ Garantisce rappresentatività

Mean Face

Centroide dello spazio vettoriale dei volti

📍 Interpretazione Geometrica

La **mean face** è il **centroide** dello spazio vettoriale:

$$\mu \in \mathbb{R}^{3008}$$

Formula

$$\mu = (1/N) \sum_{i=1}^N x_i$$

🔍 Analisi Visiva

📍 Zone Nitide

Occhi, naso: strutture comuni e allineate tra i volti

☁️ Zone Sfocate

Capelli, contorni: alta varianza inter-soggetto

〽️ PCA Sottrae μ

Le componenti catturano le **deviazioni** dalla media

💡 Significato

La mean face rappresenta il **volto "medio"** del dataset. Ogni volto può essere visto come $\mu + \text{variazioni}$

Statistiche Descrittive del Dataset

Riepilogo dimensioni e caratteristiche

Dimensioni

Immagine originale

125 x 94 pixel

Dopo resize

64 x 47 pixel

Feature vector

D = 3,008

Classi

K = 7

SplitOptions

Training Set

1,030

80%

Test Set

258

20%

Stratificato per classe

Pixel Statistics

Range originale

[0, 255]

Post-standardizzazione

$\mu = 0, \sigma = 1$

Tipo

Grayscale



Compressione Target

Da **3,008** a **100** dimensioni

Riduzione

≈ 97%

PCA: Ablation Study

Analisi dell'impatto del numero di componenti

💡 Osservazioni

1 Saturazione rapida

Dopo **~100 componenti** i benefici diventano marginali

2 Correlazione accuracy-varianza

Maggiore varianza spiegata → migliore accuracy

3 Oltre 100: rumore

Componenti aggiuntive catturano **rumore non informativo**

Risultato Ottimale

Numero Componenti

$k = 100$

Varianza

90.2%

Accuracy SVM

80.6%



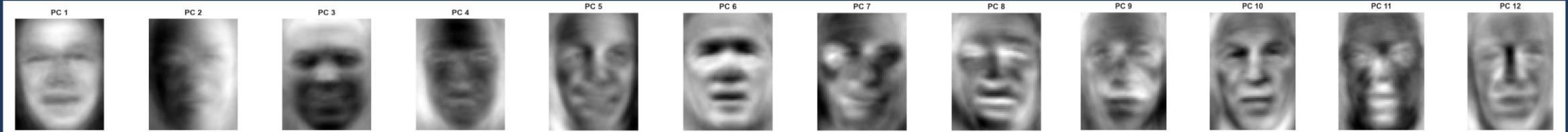
Conclusione

100 componenti rappresentano il punto ottimale: alta compressione (97%) con minima perdita di informazione discriminante

Eigenfaces: Visualizzazione

Le componenti principali come volti

Prime 12 Componenti Principali (Eigenfaces)



interpretazione

PC 1-2

Illuminazione globale - Variazioni di luce

PC 3-6

Geometria - Simmetria, posizione feature

PC 7-12

Dettagli fini - Texture, micro-variazioni

Base Ortonormale

Le **eigenfaces** costituiscono una **base ortonormale** dello spazio dei volti

Ricostruzione

$$\mathbf{x} \approx \boldsymbol{\mu} + \sum_{i=1}^k c_i \mathbf{v}_i$$

Ogni volto è combinazione lineare delle eigenfaces

Autoencoder: Training

Convergenza e stabilità del modello

Convergenza

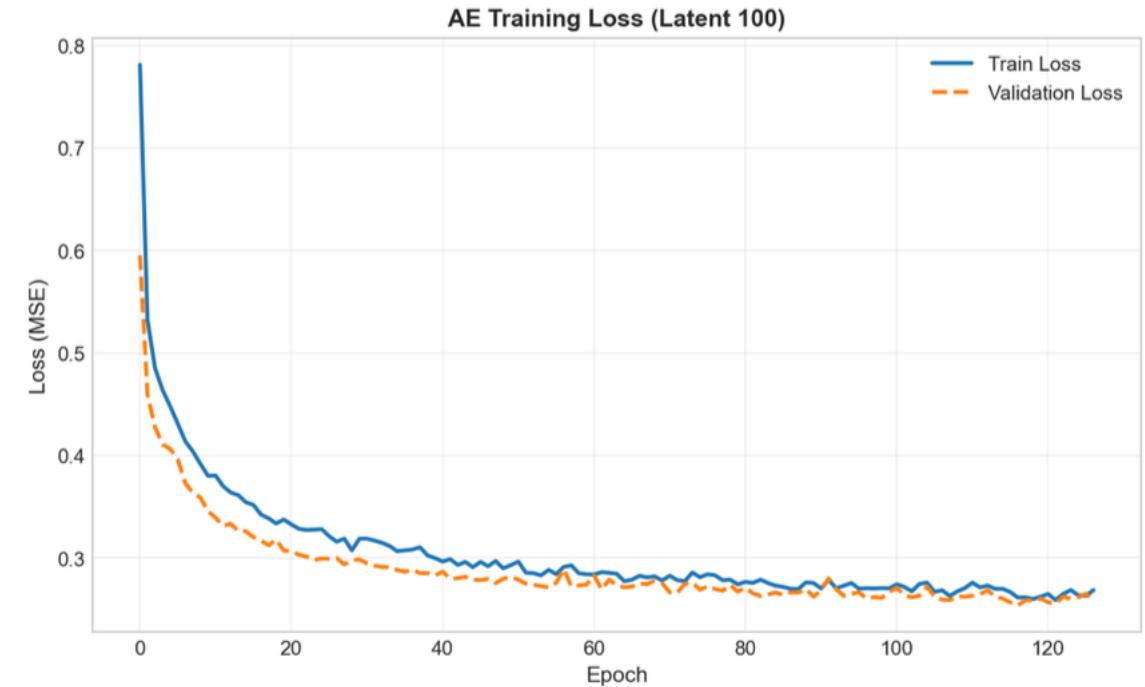
✓ **Convergenza stabile** - Nessuna oscillazione

⚖️ **Val ≈ Train** - No overfitting evidente

✋ **Early stopping** - ~epoch 60

Architettura

3008 → 256 → 128 → 100 → 128 → 256 → 3008



Training Time



~5 minuti su CPU per convergenza completa con early stopping (~2 minuti su GPU)

Autoencoder: Ablation Study

Relazione tra dimensione latente e performance

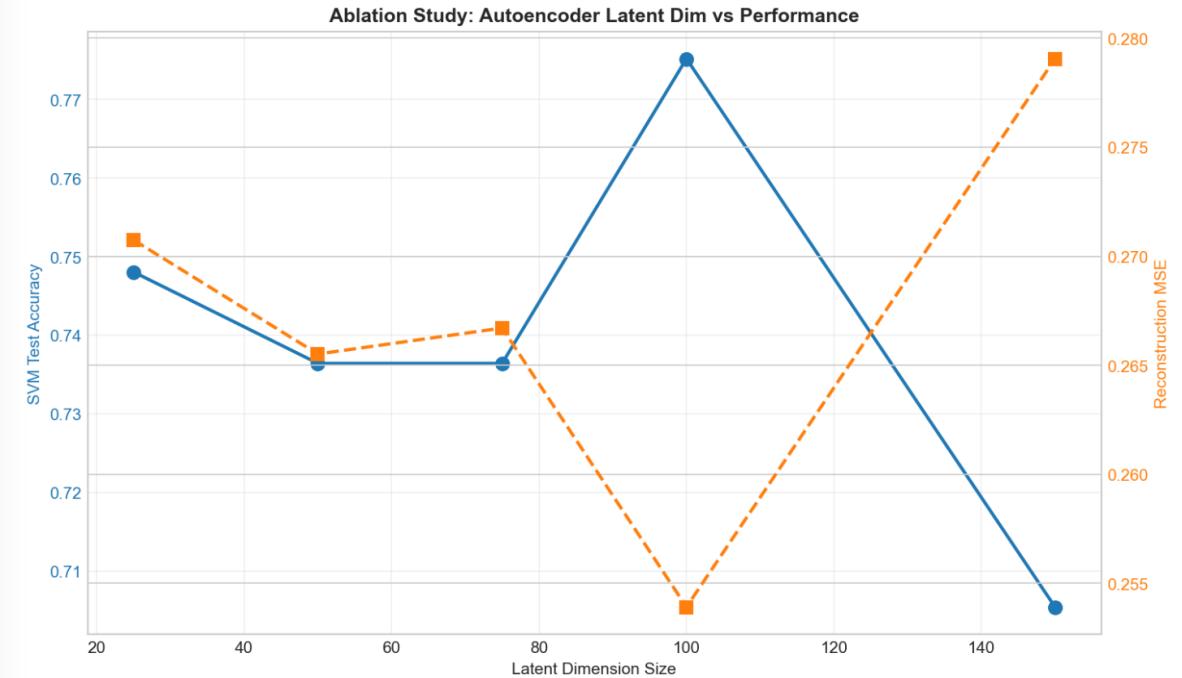
Osservazione

Il minimo MSE coincide con il massimo accuracy (dim = 100)

Interpretazione

✓ Ricostruzione migliore \Rightarrow più informazione preservata

⚠ Oltre 100: sovra-parameterizzazione



Trade-off Ottimale

$k=100$ bilancia rappresentatività e generalizzazione. Dimensioni maggiori non migliorano performance ma aumentano rischio overfitting

Confronto Ricostruzione

Qualità visiva: PCA vs Autoencoder

PCA

- ✓ **Geometricamente fedele** - Preserva struttura

- ⚠ **Artefatti alta frequenza** - Rumore nei dettagli

- ✓ **Preserva texture** - Mantieni pattern originali

MSE = 0.089

Autoencoder

- 〰 **Effetto smoothing** - Immagini più morbide

- ▀ **MSE "media" incertezze** - Approccio conservativo

- ✗ **Perde micro-dettagli** - Dettagli fini sfocati

MSE = 0.127 (smoothed)

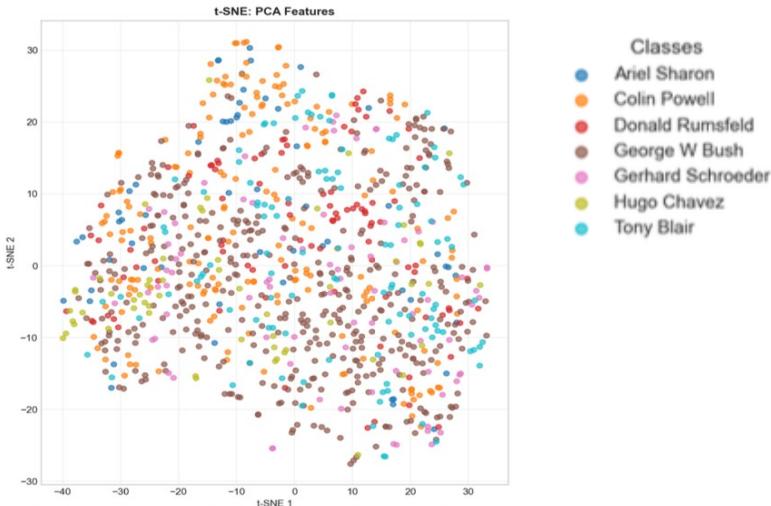
Originale vs PCA vs Autoencoder



t-SNE: Visualizzazione Spazio Latente

Struttura dei cluster nello spazio ridotto

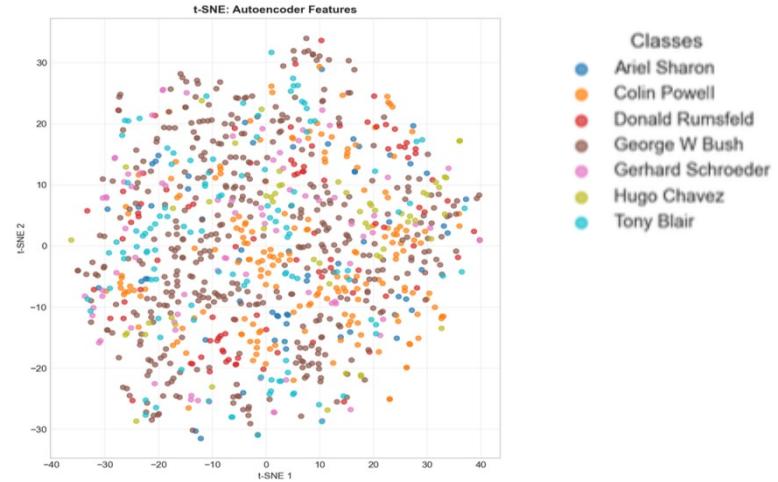
PCA



✓ **Cluster compatti** - Classi ben raggruppate

❖ **Separati** - Confini inter-classe chiari

Autoencoder



❖ **Cluster più diffusi** - Maggiore dispersione

❖ **Overlap** - Classi meno distinte



Interpretazione

PCA produce uno **spazio latente più strutturato** con cluster separati, mentre l'AE genera rappresentazioni più **diffuse e sovrapposte**

Confronto Qualitativo: PCA vs Autoencoder

Tabella riepilogativa delle performance

Aspetto	PCA	Autoencoder
Accuracy (SVM)	80.6%	79.8%
Ricostruzione MSE	0.089	0.127 (smoothed)
Varianza spiegata	90.2%	N/A
Interpretabilità	Eigenfaces	Black-box
Training time	<1 sec	~2 min
Cluster t-SNE	Compatti	Diffusi



Osservazione Chiave

Per questo dataset, **PCA è competitivo** con l'Autoencoder nonostante la linearità. La semplicità vince quando i dati hanno struttura lineare

Protocollo Sperimentale

Validazione rigorosa con cross-validation



Configurazioni Testate

Feature

PCA (100 dim) vs Autoencoder (100 dim)

Classificatori

SVM (RBF/Linear) vs Neural Network

Totale Combinazioni

4

Griglia di iperparametri 2 x 2

Risultati Quantitativi

Performance dei 4 modelli testati

Model	Features	Accuracy	F1 (W)	ROC-AUC	95% CI	CV
Neural Net	PCA	0.849	0.852	0.981	[0.80, 0.89]	0.859
Neural Net	Autoencoder	0.818	0.820	0.972	[0.77, 0.86]	0.798
SVM	PCA	0.806	0.802	0.980	[0.76, 0.85]	0.815
SVM	Autoencoder	0.798	0.794	0.971	[0.75, 0.84]	0.779



Vincitore

Neural Network + PCA

Accuracy ~**85%**

F1 85.2% | ROC-AUC 98.1%

Perché NN + PCA Vince?

Analisi del risultato contro-intuitivo

💡 Risultato Contro-intuitivo

Come mai il metodo **lineare (PCA)** batte quello **non-lineare (AE)**?

1 Dataset Allineato

LFW ha volti **già centrati**. La variabilità principale è **lineare** (illuminazione, posa).

2 Scarsità Dati

1,288 samples insufficienti per addestrare AE complessi. Rischio di **overfitting**.

3 Approccio Ibrido

PCA feature + NN classifier = best of both worlds. Feature lineari robuste + decisioni non-lineari.

✂ Occam's Razor

Principio applicato al ML

«Un modello **semplice** che spiega i dati è migliore del modello **complesso** che overfitta: la semplicità è la più grande complessità.»

ROC Curve: SVM su PCA

Performance multi-class con One-vs-Rest

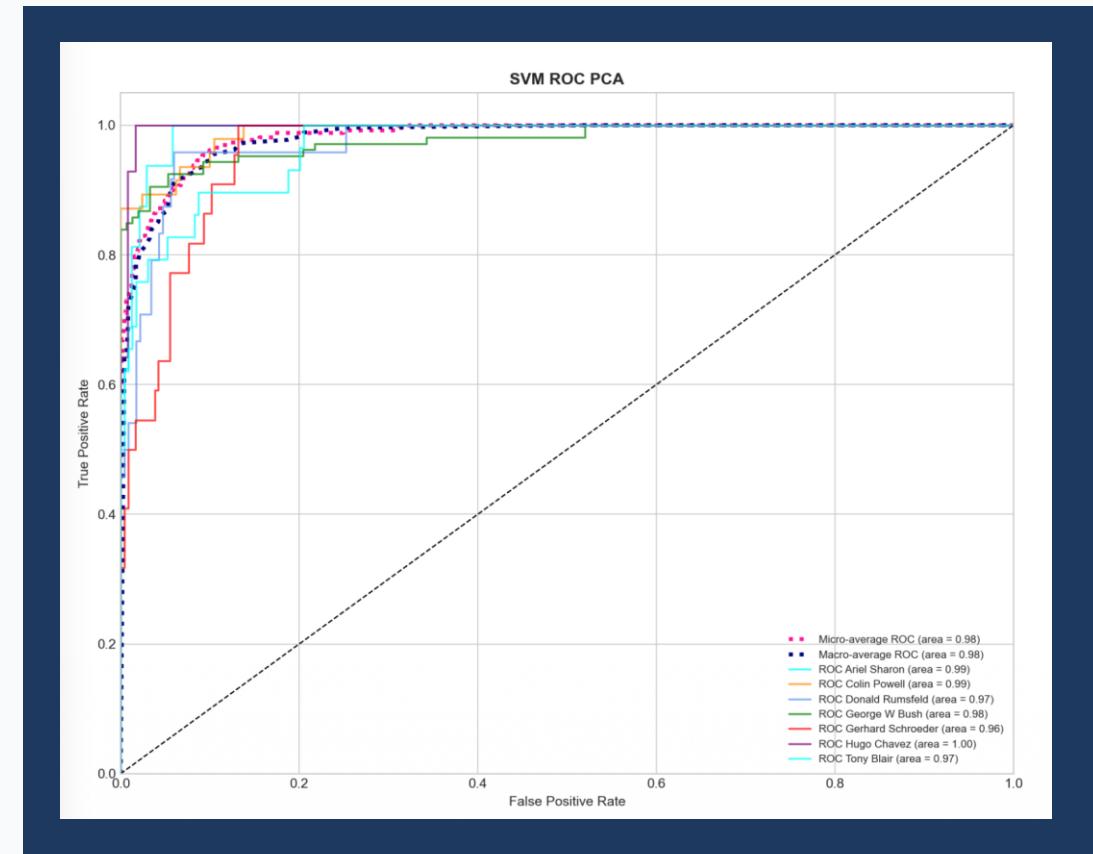
↳ Interpretazione

✓ Tutte le classi: AUC ≥ 0.95

★ Hugo Chavez: AUC ≈ 1.00

👍 Separabilità quasi lineare

Schema **One-vs-Rest**
con micro/macro averaging



Significato



Elevato AUC indica **ottima capacità discriminativa** del modello su tutte le classi

ROC Curve: Neural Network su PCA

Miglioramenti rispetto a SVM

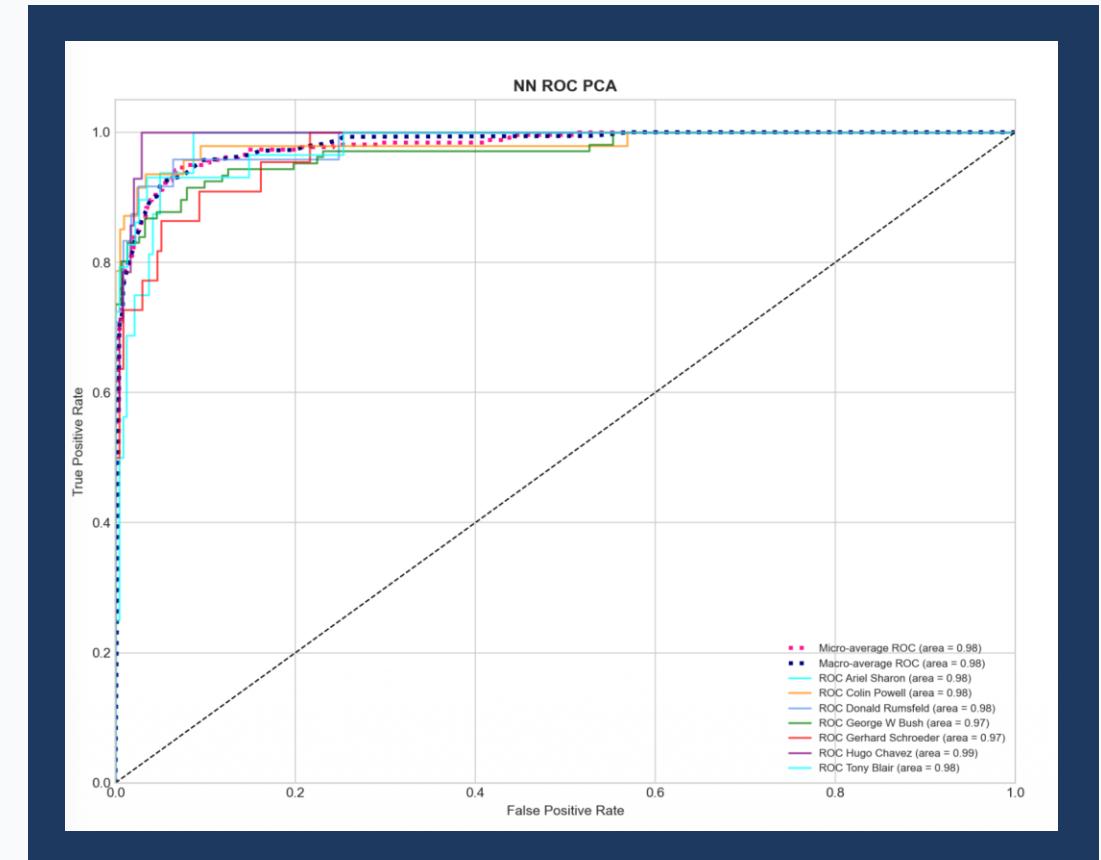
↑ Miglioramenti vs SVM

↳ Performance superiore ad alto TPR

↳ Confini decisionali più flessibili

↳ Migliore gestione classi difficili

Schema **One-vs-Rest**
con micro/macro averaging



Best Model

Neural Network + PCA

0.981

AUC

Confusion Matrix: NN + PCA (Best)

Analisi dettagliata degli errori

Pattern Errori

✓ Diagonale dominante - Predizioni corrette

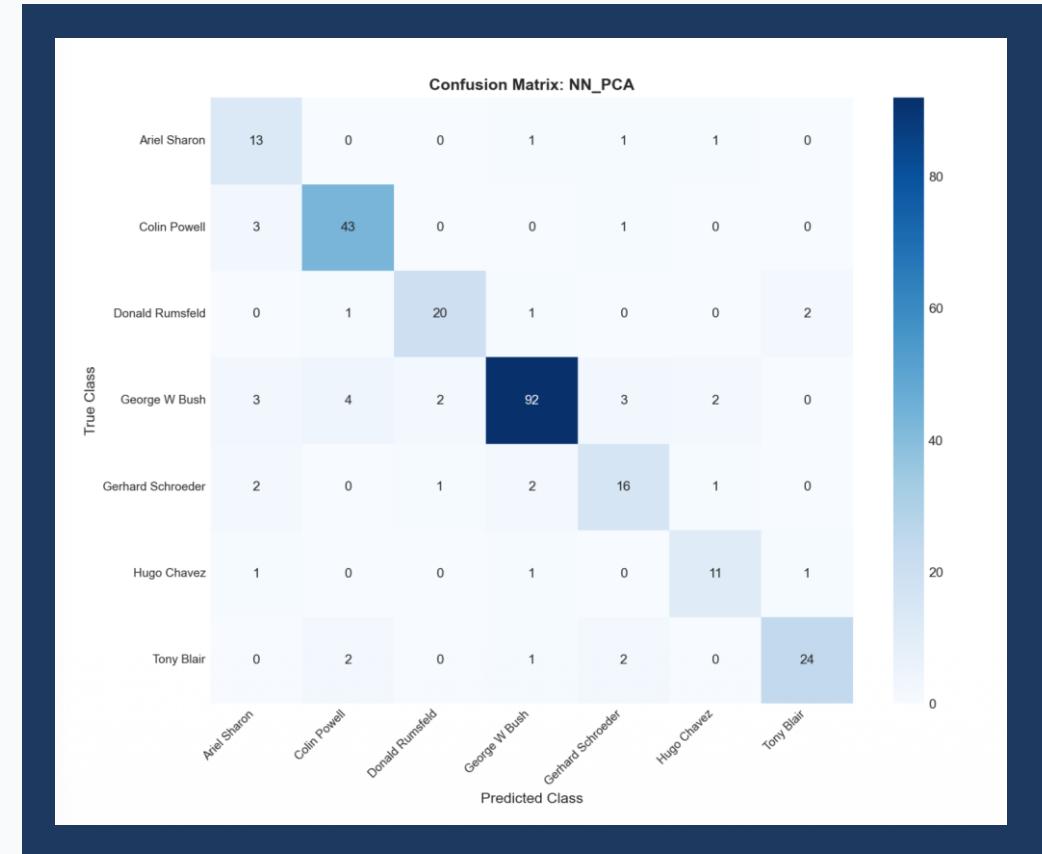
➡ Confusione: Powell . Bush

Recall per Classe

Bush 93%

Blair 88%

Powell 79%



Nota

i La confusione tra Powell e Bush è dovuta alla **somiglianza visiva** e alla **dominanza della classe Bush** nel dataset

Confusion Matrix: NN + Autoencoder

Confronto con il modello PCA

≠ Confronto vs PCA

✖ Più elementi fuori diagonale

人群 Confusione classi minoritarie

〽 Gap: ~3% accuracy

Ipotesi

AE **comprime troppo**, perdendo **discriminatività inter-classe**



Conclusione

L'Autoencoder, pur essendo più complesso, **non riesce a catturare** la struttura discriminante del dataset quanto la PCA

Setup Verifica

Protocollo per face verification 1:1

Protocollo

1 Generazione Coppie

1,000 coppie bilanciate:

- 500 **Genuine** (stessa identità)
- 500 **Impostor** (identità diverse)

2 Calcolo Similarità

Per ogni coppia calcola **similarità** nello spazio latente

3 Determinazione Soglia

Soglia ottima τ^* al punto **EER**

Equal Error Rate (EER)

Punto in cui:

$$\text{FAR}(\tau^*) = \text{FRR}(\tau^*)$$

FAR (False Accept Rate)

Impostor accettato

FRR (False Reject Rate)

Genuine rifiutato



Metriche Testate

Cosine Similarity vs **Euclidean Distance** per determinare quale metrica è più robusta

Cosine vs Euclidean: Confronto

Quale metrica performa meglio?

» Cosine Similarity

$$S = (\mathbf{z}_1 \cdot \mathbf{z}_2) / (\|\mathbf{z}_1\| \cdot \|\mathbf{z}_2\|)$$

✓ Invariante alla scala

Risultati (PCA)

AUC
0.94

EER
8.2%

» Euclidean Distance

$$d = \|\mathbf{z}_1 - \mathbf{z}_2\|_2$$

✗ Sensibile alla magnitudine

Risultati (PCA)

AUC
0.91

EER
11.5%



Vincitore

Cosine Similarity performa meglio per face verification (AUC 0.94 vs 0.91, EER 8.2% vs 11.5%)

Distribuzione Similarità

Separazione Genuine vs Impostor

Interpretazione

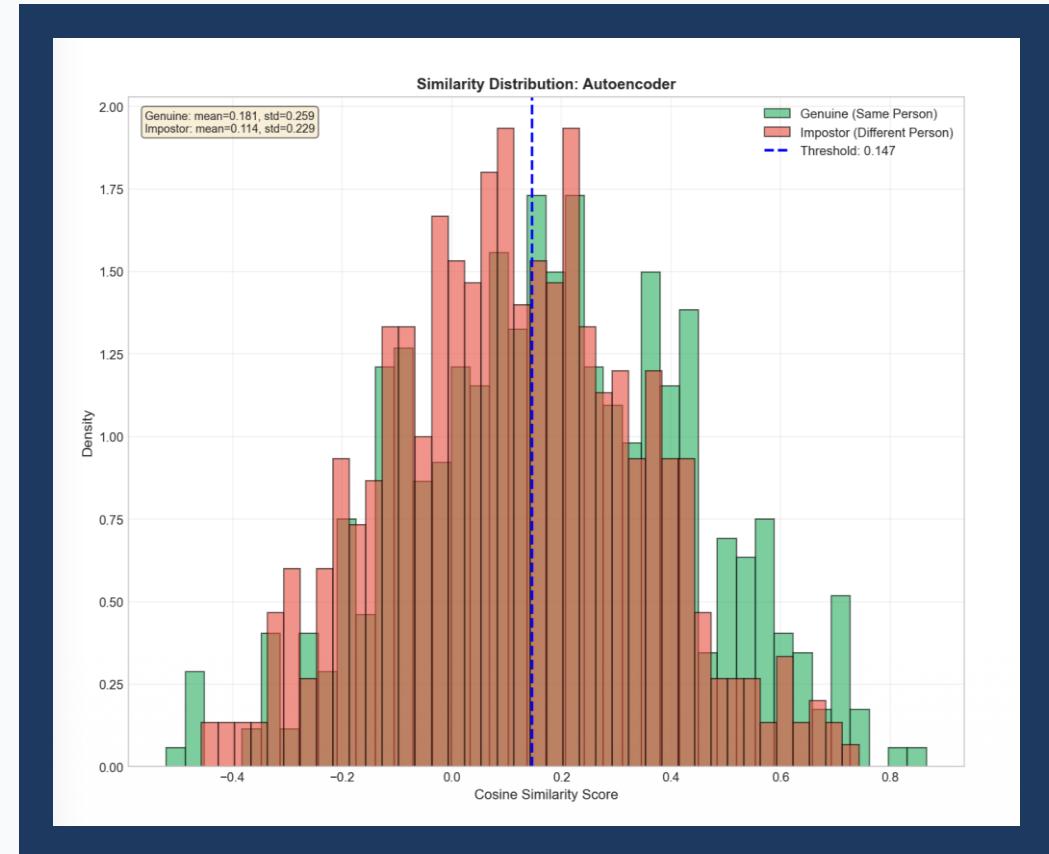
- ✓ Separazione netta tra distribuzioni

- ✗ Soglia: $\tau^* \approx 0.147$

- ✗ Overlap = errore irriducibile

EER $\approx 8\%$

Sistema per sicurezza media



Significato

EER = 8% indica un sistema **utilizzabile per sicurezza media** (es. accesso smartphone) ma non per applicazioni critiche

Esempi di Verifica

Genuine pairs vs Impostor pairs

✓ Genuine Pairs

Copie con **stessa identità** → Similarità alta

Genuine
Score: 0.728



Genuine
Score: 0.759



Genuine
Score: 0.760



Genuine
Score: 0.811



✗ Impostor Pairs

Copie con **identità diverse** → Similarità bassa

Impostor
Score: 0.654



Impostor
Score: 0.670



Impostor
Score: 0.702



Impostor
Score: 0.743



Heatmap Inter-Class: PCA

Similarità media tra classi diverse

Analisi

■ Diagonale verde scuro (≈ 1) - Alta auto-similarità

■ Off-diagonal rosso (< 0) - Bassa similarità inter-classe

❖ Classi proiettate in direzioni ortogonali

Interpretazione

Ottima separabilità inter-classe



Conclusione

PCA produce feature con **ottima separabilità inter-classe**, ideale per classificazione e verifica

Heatmap Inter-Class: Autoencoder

Confronto con PCA

≠ Confronto vs PCA

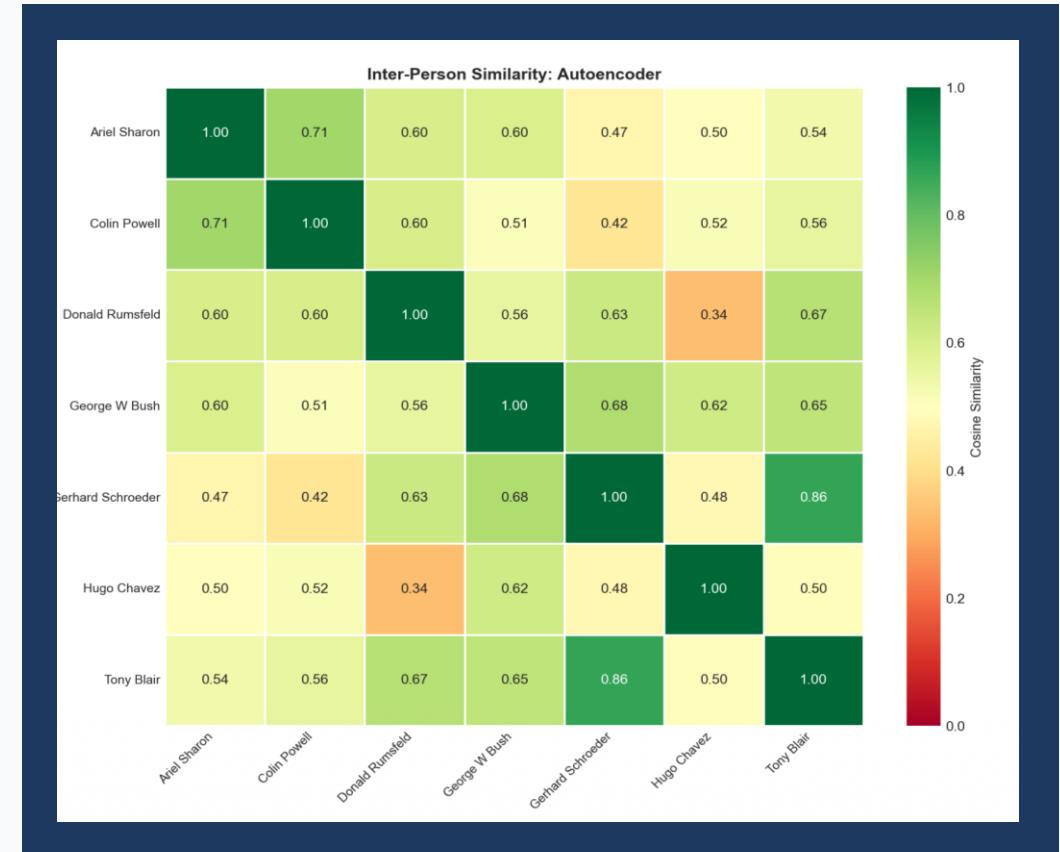
👉 Colori più chiari off-diagonal

👉 Similarità inter-classe maggiore

👉 Meno separabilità

Interpretazione

AE mappa identità diverse in **regioni più vicine** nello spazio latente



Problema

L'Autoencoder produce feature con maggior sovrapposizione tra classi, riducendo la discriminatività

Riepilogo Verification

Confronto completo delle metriche

Features	Metric	AUC	EER	Threshold
PCA	Cosine	0.94	8.2%	0.147
PCA	Euclidean	0.91	11.5%	12.3
Autoencoder	Cosine	0.89	12.1%	0.203
Autoencoder	Euclidean	0.86	15.8%	8.7



Best

PCA + Cosine Similarity



Conclusione

Per la verifica, le **feature lineari sono più robuste**

- Il gap tra PCA e AE è più marcato in verification che in recognition, confermando la maggiore discriminatività delle feature PCA

Sintesi dei Risultati

Punti chiave dell'analisi comparativa

1 Compressione 97%

Da **3,008 a 100 dimensioni**, accuracy >84%. Dimostrazione efficacia riduzione dimensionale.

2 Linear Wins

PCA supera Autoencoder su dataset piccolo e allineato. La semplicità vince quando i dati lo consentono.

3 Hybrid Power

Feature PCA + Classificatore NN = combinazione ottima. Best of both worlds.

4 Cosine > Euclidean

Per verification, **invarianza alla scala** è cruciale. Cosine similarity performa meglio.

5 EER 8%

Sistema utilizzabile per sicurezza media

85%

Best Accuracy

Criticità e Sviluppi Futuri

Limitazioni e direzioni di ricerca

⚠ Limitazioni

- ✖ **AE Fully-Connected:** ignora struttura 2D delle immagini
- ⌚ **Dataset limitato:** 1,288 samples insufficienti per AE complessi
- ⚖️ **Sbilanciamento:** impatta recall classi minoritarie
- 🚫 **No data augmentation:** manca variabilità artificiale

🔑 Direzioni Future

- .GridView **Convolutional AE:** sfruttare correlazioni spaziali locali
- 👉 **Triplet Loss:** ottimizzare direttamente distanze
- 🖼️ **Data Augmentation:** rotazioni, flip, illuminazione
- ☁️ **Transfer Learning:** pre-training su dataset grandi



Prospettiva

Con **ConvAE** e **più dati**, l'approccio non-lineare potrebbe superare il lineare. Il potenziale dell'AE non è stato pienamente sfruttato

Concetti Chiave

5 messaggi da ricordare

1

Non sempre non-lineare > lineare

Dipende dai dati. Dataset piccoli e allineati favoriscono **metodi semplici**

2

Dimensionality reduction è cruciale

97% **compressione** con minima perdita

3

Approcci ibridi possono essere vincenti

PCA feature + NN classifier = combinazione ottima (il meglio di entrambi gli approcci)

4

Metrica conta

Cosine vs Euclidean fa la differenza. Scegliere in base al task

5

Validazione rigorosa è essenziale

Valutare **Cross-validation** e **confidence intervals** per risultati affidabili



Contributo Principale

Analisi comparativa rigorosa con **ablation study** che dimostra l'efficacia della **PCA** su dataset di face recognition **pre-allineati**



Grazie per l'attenzione



Codice e documentazione disponibili nel **repository**

<https://git.new/Gb69fEp>