

Analisi Comparativa per Riduzione Dimensionale

Approccio Lineare e Non-Lineare per
Riconoscimento e Verifica Facciale

Roberto Carriero, Massimiliano Leone

Laurea Magistrale in Ingegneria Informatica – Artificial Intelligence and Data Science

Docente: Prof.ssa Marina Popolizio

A.A. 2025/2026

Indice

Struttura della presentazione

1 Introduzione e Motivazioni

Contesto applicativo, maledizione della dimensionalità, obiettivi del progetto

3 Dataset e Preprocessing

LFW, analisi sbilanciamento, pipeline di preprocessing, faccia media

5 Risultati: Face Recognition

Protocollo sperimentale, risultati quantitativi, curve ROC e matrici di confusione

Obiettivo

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

2 Framework Matematico

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

4 Esperimenti: Riduzione Dimensionale

Studio di ablazione PCA, eigenfaces, addestramento AE, confronto di ricostruzione

6 Risultati: Face Verification

Setup di verifica, distanza euclidea vs similarità del coseno, distribuzione di similarità, heatmap

01

Obiettivi del Progetto

Fasi principali dello studio

1

Riduzione Dimensionale

PCA (SVD) vs Autoencoder

Confronto tra approccio lineare e non-lineare

↳ Analisi della varianza spiegata e MSE ricostruzione

2

Classificazione

SVM vs Rete Neurale (NN)

Valutazione di classificatori lineari e non-lineari su feature ridotte

✖ Accuratezza, F1-Score, curva ROC-AUC

3

Face Verification

Distanza Euclidea vs Similarità del Coseno

Confronto tra metriche di similarità per task di verifica

⚖️ EER, AUC, TAR, FAR

4

Studio di Ablazione

Analisi Componenti e Parametri

Variazioni al numero di componenti, architettura, iperparametri

🔍 Grid Search e Cross-Validation

?

Tesi della Ricerca

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

Applicazione

Sicurezza

Controllo degli accessi, sorveglianza, autenticazione biometrica

Mobile

Sblocco smartphone (Face ID), app di autenticazione

Sistemi Embedded

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 efficienza

3 Alta accuratezza

Precisione nel riconoscimento e robustezza alle variazioni

Approfondimento: Dispositivi Edge



Ampia Distribuzione

Dispositivi con memoria limitata



Bassa Latenza

Inferenza rapida in tempo reale



Consumo Energetico

Meno calcoli = meno energia



Storage Ridotto

Meno dati e più compatti

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 Accuratezza

Predizione corrette vs numero totale di esempi osservati

2 F1-Score Ponderato

Media pesata per gestire sbilanciamento classi

3 ROC-AUC

Classificazione multiclasse (OvR) con micro/macro averaging

4 Matrice di Confusione

Analisi dettagliata degli errori per classe



Scenario

Dataset con **K identità**: l'immagine in input viene associata a una di queste

Dataset

K = 7 classi

Task 2: Verifica Facciale (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

ROC / AUC

Area sottesa alla curva ROC di verifica

⚙️ TAR-FAR

True Accept Rate, fissato un certo False Accept Rate (es. 0.1%)



Coppia Genuina

Stessa identità → Similarità alta



Coppia Impostore

Identità diverse → Similarità bassa

Problema: Maledizione della Dimensionalità

Sfida matematica in spazi ad alta dimensionalità

⚠ Sfida Matematica

Esempio Concreto

Immagine **64 × 47 pixel – D = 3008** dimensioni

Ogni immagine è un punto in uno spazio 3008-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

Il volume si concentra nelle regioni estreme

→ Dati necessari crescono esponenzialmente

Per coprire lo spazio servono $O(e^D)$ campioni

Riepilogo

1. Distanze perdono significato

Tutti i punti diventano «equidistanti»

2. Classificatori perdono potere

Overfitting più probabile

3. Distanze non discriminanti

Similarità non significative



Soluzione: Riduzione Dimensionale

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

💡 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 un sottospazio
- ✓ **Preserva varianza** – Massimizza informazione contenuta
- ✓ **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** – Iterativa (discesa del gradiente)

PCA: Formulazione del Problema di Ottimizzazione

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 corrispondono 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 costituiscono 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 ai valori singolari per efficienza

⟳ Teorema SVD

Ogni matrice $X \in \mathbb{R}^{NxD}$ 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

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

</> Implementazione

1 Centra i dati

$X_c = X - \mu$ 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 vs Decomposizione agli Autovalori

Stabilità numerica ed efficienza computazionale. Evita il calcolo esplicito di Σ

Varianza Spiegata Cumulativa

Quanta informazione viene preservata?

% Varianza Spiegata

Per ogni componente i:

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

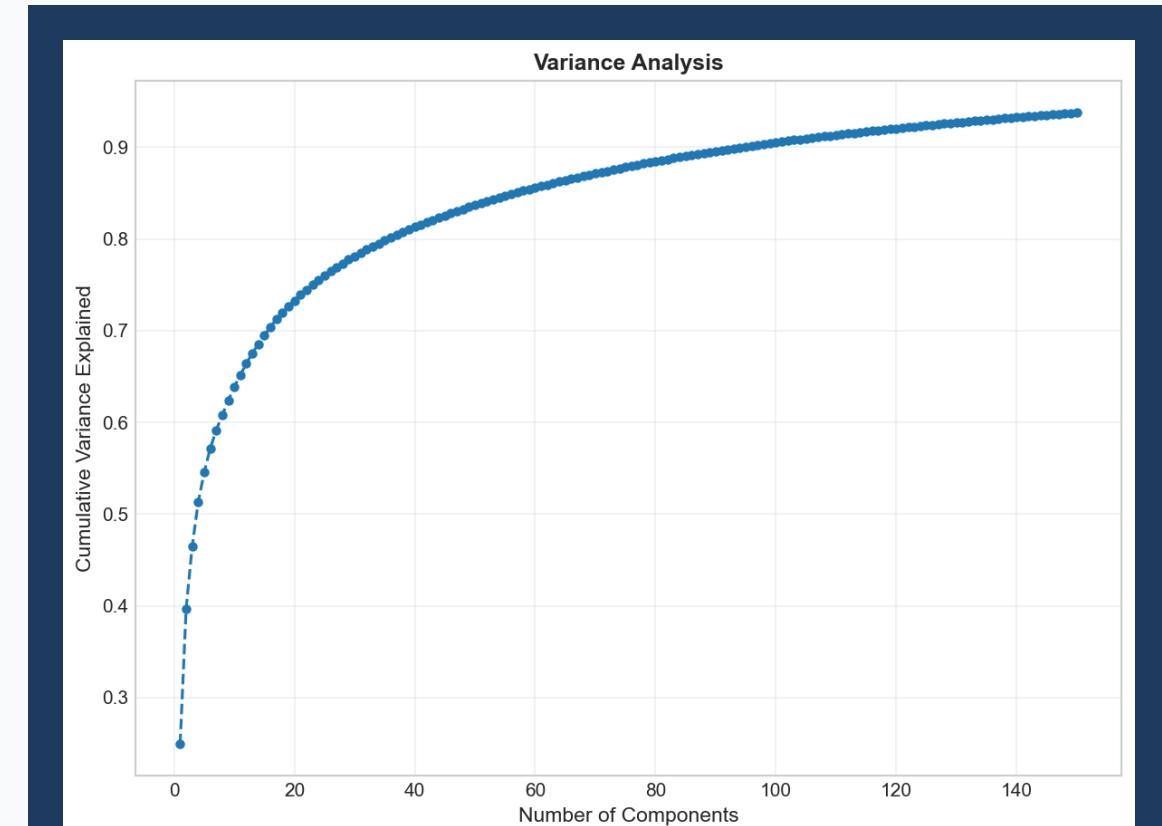
- **Frazione di varianza** catturata dalla componente i-esima
- ↳ Somma di tutti gli EVR = **1 (100%)**

↳ 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



Target

90-95% varianza

Autoencoder: Architettura

Rete neurale per riduzione 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:

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

⚙️ Dettagli Implementativi

⚡ Attivazione

LeakyReLU ($\alpha = 0.2$)

⟳ Optimizer

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

📦 Batch Norm

Dopo ogni layer

🛡️ Regolarizzazione

✗ Dropout

$p = 0.1$

.Weight Decay

$\lambda = 10^{-4}$

✋ Early Stopping

patience = 10

Confronto Teorico: PCA vs Autoencoder

Tabella comparativa dei due approcci

Caratteristica	PCA (SVD)	Autoencoder
Natura	Lineare	Non-lineare
Soluzione	Analitica (forma chiusa)	Iterativa (discesa del gradiente)
Ottimizzazione	Convessa	Non-convessa
Complessità	$O(N \cdot D \cdot \min(N, D))$	$O(E \cdot N \cdot \theta)$
Interpretabilità	Alta (eigenfaces)	Bassa (black-box)
Dati richiesti	Pochi	Molti
Capacità	Sottospazi lineari	Manifold non-lineari

Metriche di Similarità per Verifica

Confronto teorico tra le due distanze

» Similarità del Coseno

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

- ✓ Invariante alla magnitudine – Robusto alla 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
- ! Distanza effettiva nello spazio



Relazione Matematica

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

Pipeline di Analisi Completa

Flusso operativo al variare del numero di componenti



Tecnologie Utilizzate

Python 3.14

NumPy, Pandas, Scikit-Learn, Matplotlib, PyTorch

PCA from scratch

Implementata con SVD, senza libreria Sci-kit Learn

Autoencoder

Rete neurale con PyTorch

Grid Search / Cross Validation

5 Fold di validazione

Dataset: Labeled Faces in the Wild (LFW)

Stato dell'arte per il riconoscimento facciale

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

1288

Immagini Totali

1030

Train (80%)

258

Test (20%)

Analisi dello Sbilanciamento

Soluzioni per compensare lo sbilanciamento delle classi

⚠ Problema

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

Conseguenze

→ **Bias** verso la classe maggioritaria

→ **Sensibilità bassa** per classi minoritarie

→ **Metriche aggregate fuorvianti**

🔧 Soluzioni Adottate

☒ Suddivisione stratificata

Mantiene **proporzioni** tra train e test

⚖ Peso delle classi

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

Ἑ F1-Score ponderato

Metrica **robusta** allo sbilanciamento

Distribuzione delle Classi

Bush
530

Powell
236

Blair
144

Rumsfeld
121

Schroeder
109

Sharon
77

Chavez
71

Preprocessing

Dall'acquisizione alla standardizzazione



Standardizzazione (Z-score)

$$z_i = (x_i - \mu) / \sigma$$

Motivazioni

- ✓ **Stabilità numerica** SVD
- ✓ **Convergenza rete neurale** più rapida
- ✓ **Comparabilità** feature

Suddivisione Stratificata

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

Importanza

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

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

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

Iperparametri (Grid Search)

C ∈ {0.1, 1, 10}

Kernel ∈ {RBF, Linear}

γ ∈ {scale, auto}



Limitazione

Scalabilità limitata. Costoso su dataset grandi

Classificatori: Neural Network

Rete neurale con bilanciamento delle classi

Architettura

Input → [256] → [128] → [K]

- Batch Normalization – Stabilizza il training
- Dropout ($p = 0.1 - 0.2$) – Previene overfitting
- ReLU Activation Function – Non-linearietà

Cross-Entropy Loss (Pesata)

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

Pesi per sbilanciamento:

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

 Dà più peso alle classi minoritarie

Iperparametri (Grid Search)

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

Learning Rate
 $\{10^{-2}, 10^{-3}\}$

Dropout
 $\{0.1, 0.2\}$

Epochs
 $\{50, 100\}$

Ottimizzazione dei Modelli

Validazione incrociata e ricerca dei migliori iperparametri



5-Fold CV
Stratificata



Grid Search
Iperparametri



Ri-addestramento
Migliori Iperparametri



Valutazione
Test Set

Configurazioni Testate

Feature

PCA (100 dim) vs Autoencoder (100 dim)

Classificatori

SVM (RBF/Linear) vs Neural Network

Combinazioni Totali

4

Griglia di iperparametri 2 x 2

Faccia Media

Centroide dello spazio vettoriale dei volti

📍 Interpretazione Geometrica

La **faccia media** è il **centroide** dello spazio vettoriale:

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

Formulazione

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

🔍 Analisi Visiva



📍 Zone Nitide

Strutture comuni e allineate tra i volti

☁️ Zone Sfocate

Alta varianza **inter-soggetto**

〽️ PCA Sottrae μ

Componenti – **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 delle caratteristiche dei dati

Dimensioni

Immagine originale

125 × 94 pixel

Dopo ridimensionamento

64 × 47 pixel

Vettore delle feature

D = 3,008

Classi

K = 7

SplitOptions

Training Set

1,030

80%

Test Set

258

20%

Stratificato per classe

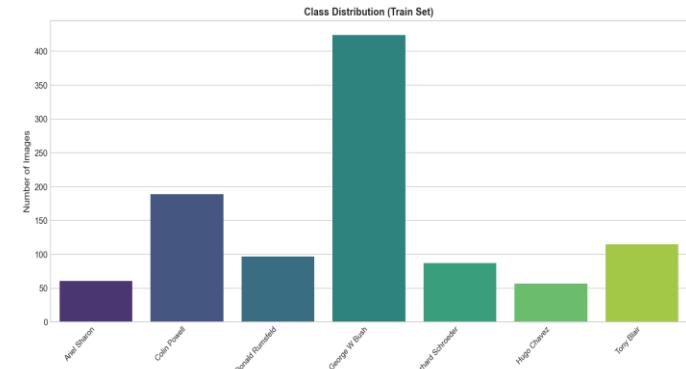
Pixel Statistics

Range originale

[0, 255]

Post-standardizzazione

$\mu = 0, \sigma = 1$



Compressione Target

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

Riduzione

≈ 97%

PCA: Studio di Ablazione

Analisi dell'impatto del numero di componenti

Osservazioni

1 Saturazione rapida

Dopo ~100 componenti i benefici diventano marginali

2 Correlazione accuratezza – varianza

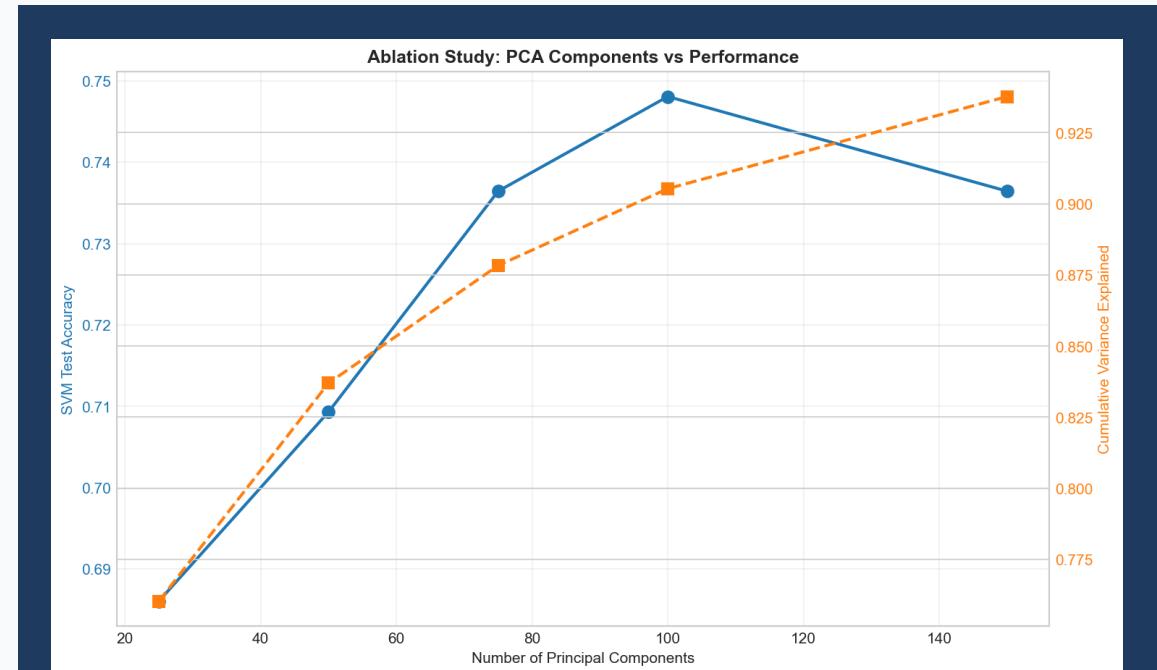
Maggiore varianza spiegata = migliore accuratezza

3 Oltre k=100: rumore

Componenti aggiuntive catturano **rumore non informativo**

Conclusione

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



Numero Componenti
k = 100

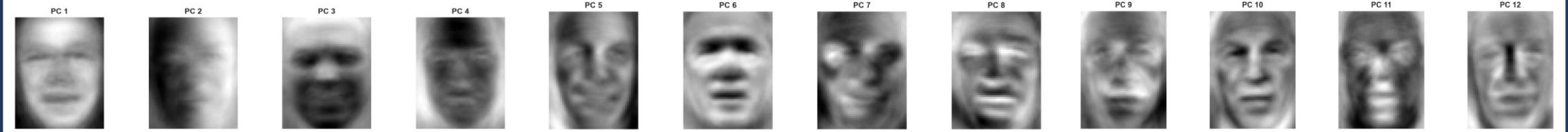
Varianza
90.2%

Accuratezza SVM
80.6%

Eigenfaces

Le componenti principali come volti

Prime 12 Componenti Principali



interpretazione

PC 1-2

Illuminazione globale – Variazione di luce

PC 3-6

Geometria – Simmetria, posizione delle feature

PC 7-12

Dettagli fini – Texture, micro-variazioni

Base Ortonormale

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

Ricostruzione

$$x \approx \mu + \sum_{i=1}^k c_i v_i$$

💡 Ogni volto è una combinazione lineare delle eigenfaces

Autoencoder: Addestramento

Convergenza e stabilità del modello

Convergenza

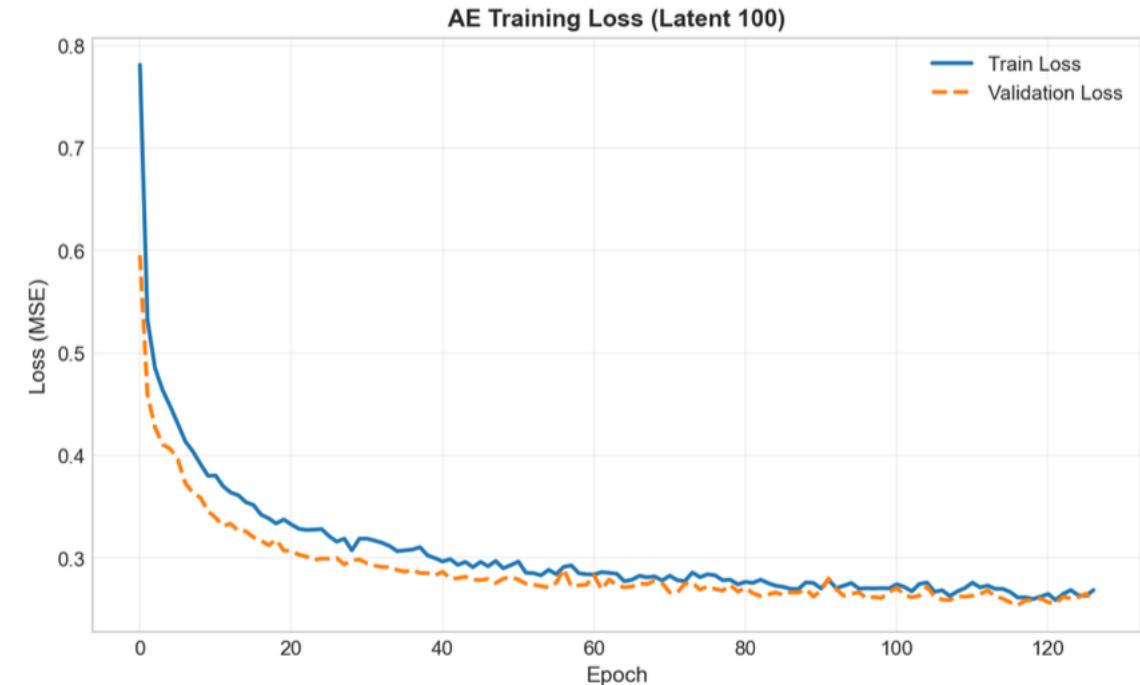
✓ **Convergenza stabile** – Nessuna oscillazione

⚖️ **Validation Loss ≈ Train Loss** – No overfitting evidente

✋ **Early stopping** – epoch 60

Architettura

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



Training Time



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

Autoencoder: Studio di Ablazione

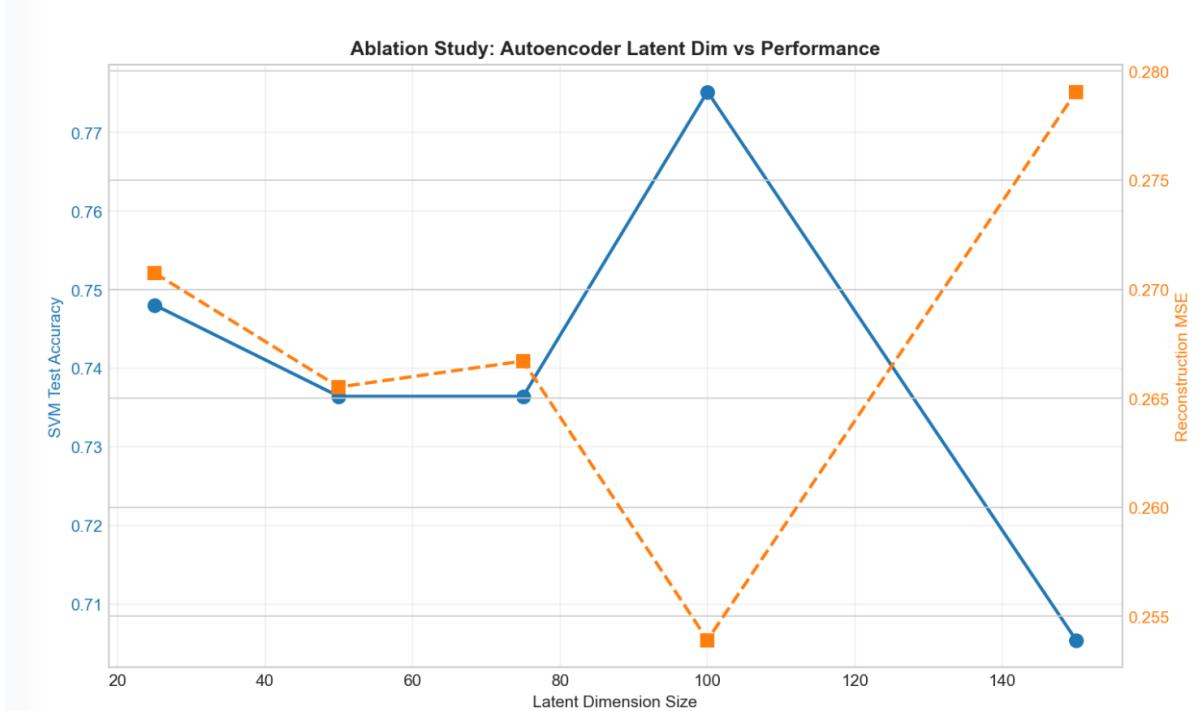
Relazione tra dimensione latente e prestazioni

Osservazione

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

Interpretazione

- ✓ Ricostruzione migliore – più informazione preservata
- ⚠ Oltre k=100: sovra-parameterizzazione



Trade-off Ottimale

k=100 bilancia rappresentatività e generalizzazione. Più componenti non migliorano le performance ma aumentano il rischio di overfitting

Errore di Ricostruzione

Confronto dei volti ricostruiti dopo la riduzione dimensionale (Errore Quadratico Medio, MSE)

PCA

- ✓ **Geometricamente fedele** – Preserva struttura

- ⚠ **Artefatti ad alte frequenze** – Rumore nei dettagli

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

MSE = 0.089

Autoencoder

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

- Cal **MSE "media" incertezze** – Approccio conservativo

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

MSE = 0.127

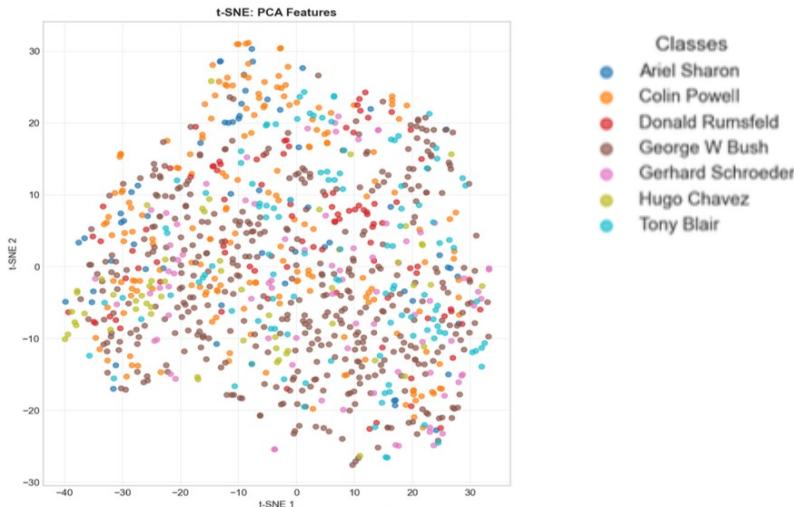
Originale vs PCA vs Autoencoder



t-SNE: Visualizzazione Spazio Latente

Struttura dei cluster nello spazio ridotto

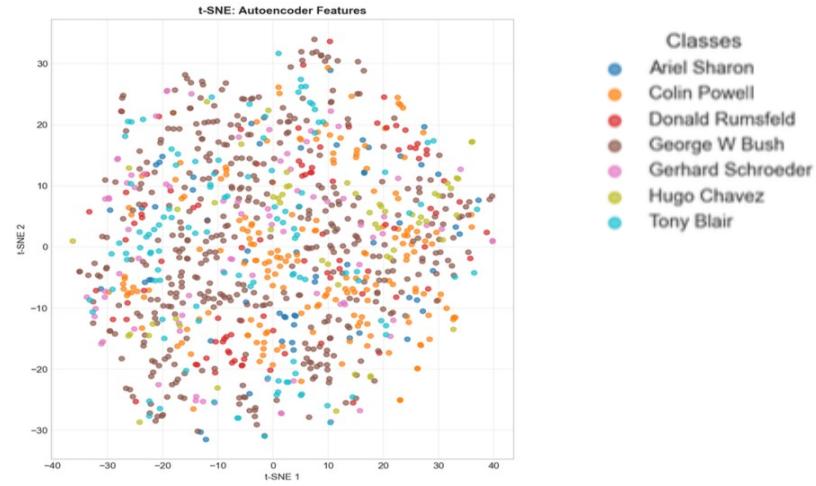
PCA



✓ **Cluster compatti** – Classi ben raggruppate

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

Autoencoder



❖ **Cluster più diffusi** – Maggiore dispersione

❖ **Sovraposizioni** – Classi meno distinte



Interpretazione

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

PCA vs Autoencoder: Confronto Qualitativo

Tabella riepilogativa delle prestazioni

Aspetto	PCA	Autoencoder
Accuratezza (SVM)	80.6%	79.8%
Errore di Ricostruzione (MSE)	0.089	0.127
Varianza spiegata	90.2%	N/A
Interpretabilità	Eigenfaces	Black-box
Tempo di Addestramento	< 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

PCA vs Autoencoder: Confronto Quantitativo

Prestazione dei 4 modelli testati

Modello	Feature	Accuratezza	F1-Score	ROC-AUC	95% Int. Confid.	CV Fold Acc.
Rete Neurale	PCA	0.849	0.852	0.981	[0.80, 0.89]	0.859
Rete Neurale	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

Perché 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 dei Dati

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

3 Approccio Ibrido

Rete Neurale su feature PCA consente decisioni non-lineari su feature lineari robuste

✂ Rasoio di Occam

Principio applicato al Machine Learning

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

SVM su PCA: Curva ROC

Prestazione multiclass con approccio One vs Rest

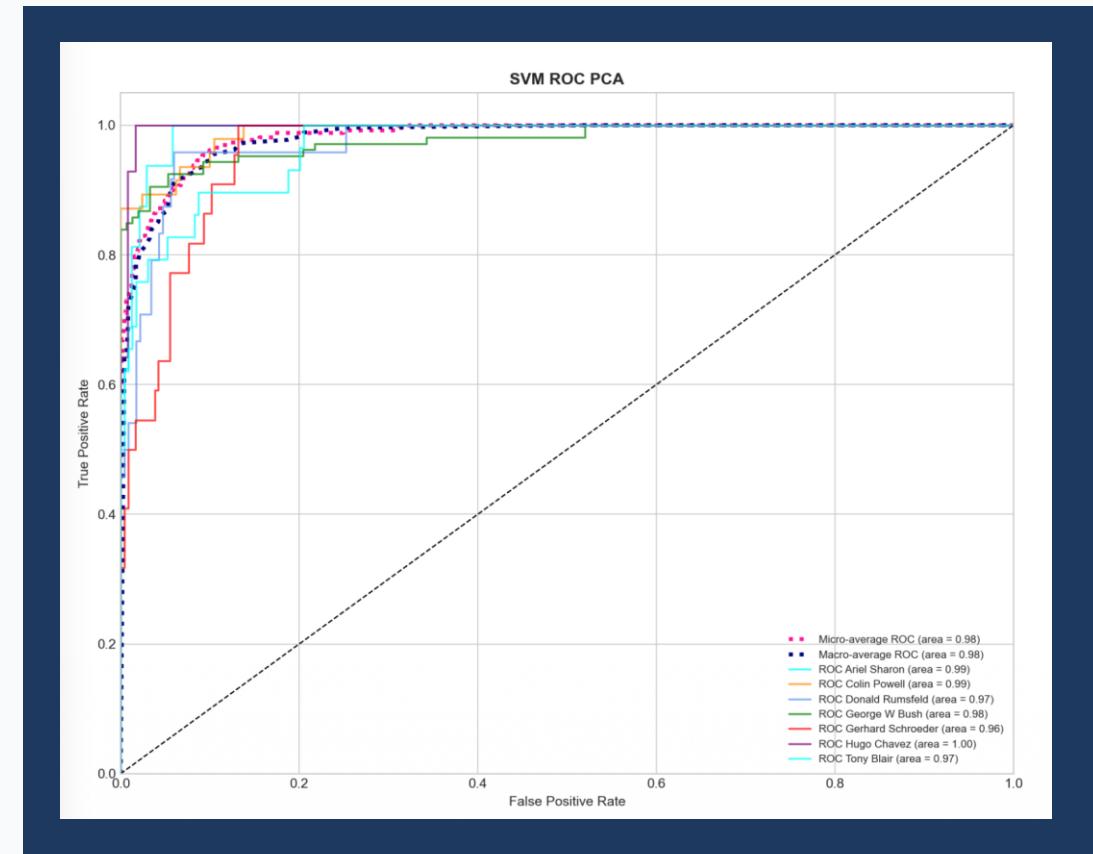
↳ Interpretazione

✓ 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

Rete Neurale su PCA: Curva ROC

Migliori prestazioni

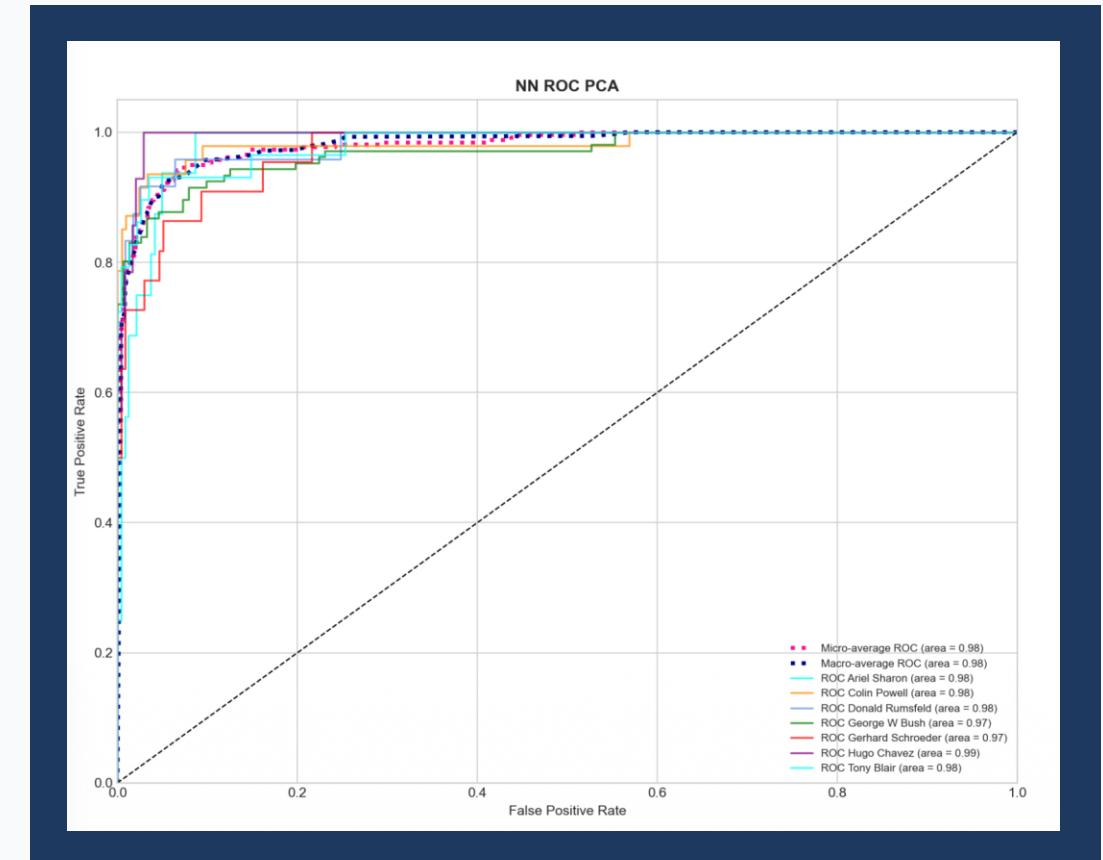
↑ Miglioramenti rispetto a SVM

↗ Prestazioni superiori ed alto TPR

➡ Confini decisionali **più flessibili**

人群 Migliore gestione **classi difficili**

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



Modello Migliore

Neural Network + PCA

0.981

AUC

Rete Neurale su PCA: Matrice di Confusione

Analisi dettagliata degli errori

Pattern Errori

✓ Diagonale dominante – Predizioni corrette

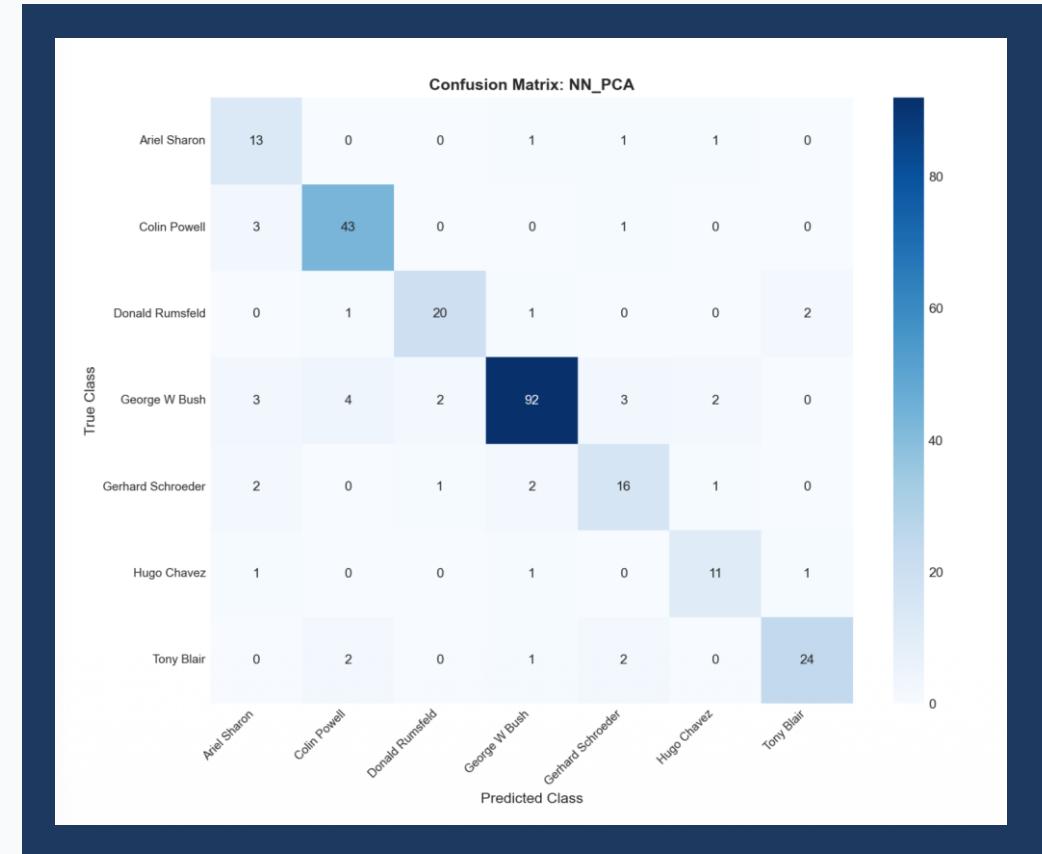
➡ Confusione: Powell – Bush

Sensibilità per Classe

Bush 93%

Blair 88%

Powell 79%



Nota



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

Rete Neurale su Autoencoder: Matrice di Confusione

Confronto con il modello PCA

≠ Confronto con 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 di Verifica

Protocollo per verifica facciale

Protocollo

1 Generazione Coppie

1,000 coppie bilanciate:

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

2 Calcolo Similarità

Per ogni coppia calcolare **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

Euclidea vs Coseno: Confronto

Quale metrica di similarità performa meglio?

» Similarità del Coseno

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

✓ Invariante alla scala

Risultati (PCA)

AUC
0.94

EER
8.2%

» Distanza Euclidea

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

✗ Sensibile alla magnitudine

Risultati (PCA)

AUC
0.91

EER
11.5%



Vincitore

Similarità del coseno performa meglio per verifica facciale (AUC 0.94 vs 0.91, EER 8.2% vs 11.5%)

Distribuzione di Similarità

Separazione Genuini vs Impostori

Interpretazione

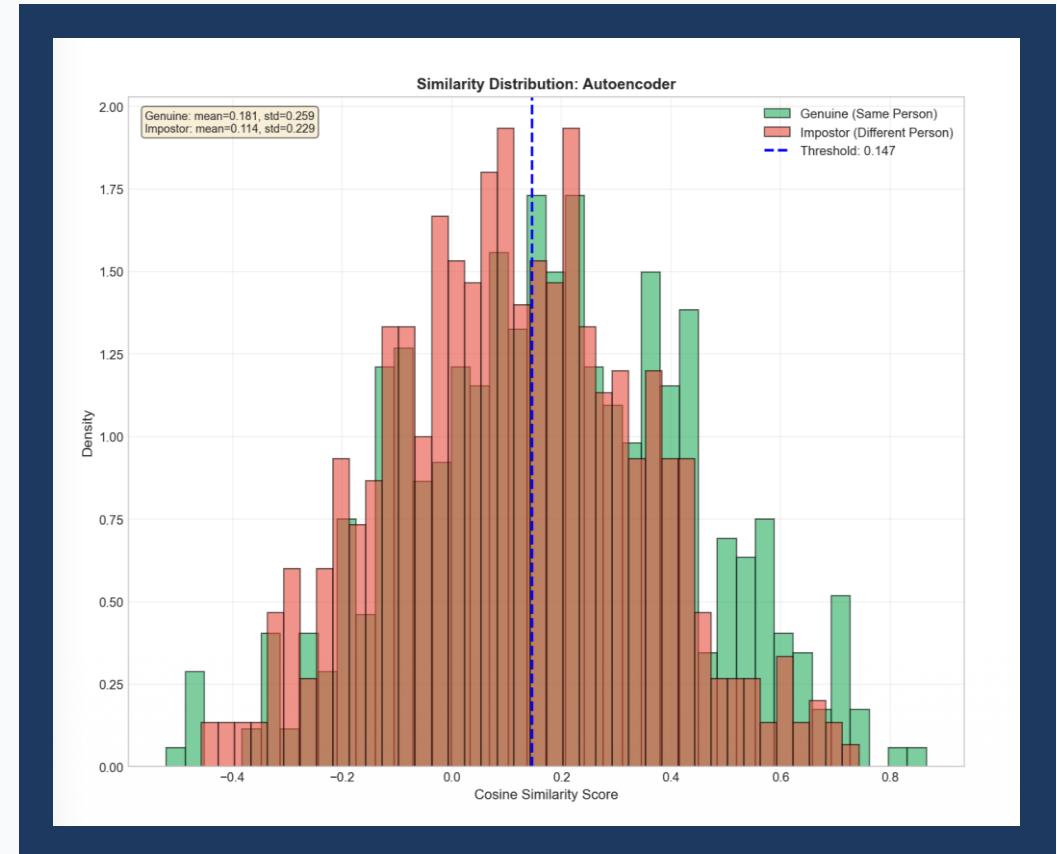
✓ **Separazione netta** tra distribuzioni

✗ **Soglia:** $\tau^* \approx 0.147$

✗ **Sovraposizioni:** 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

Visualizzazione coppie genuine e impostori

✓ Coppie Genuine

Coppie con **stessa identità** – Similarità alta

Genuine
Score: 0.728



Genuine
Score: 0.759



Genuine
Score: 0.760



Genuine
Score: 0.811



✗ Coppie Impostori

Coppie con **identità diverse** – Similarità bassa

Impostor
Score: 0.654



Impostor
Score: 0.670



Impostor
Score: 0.702



Impostor
Score: 0.743



Heatmap Inter-Classe: 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 rispetto alla PCA

≠ Confronto con PCA

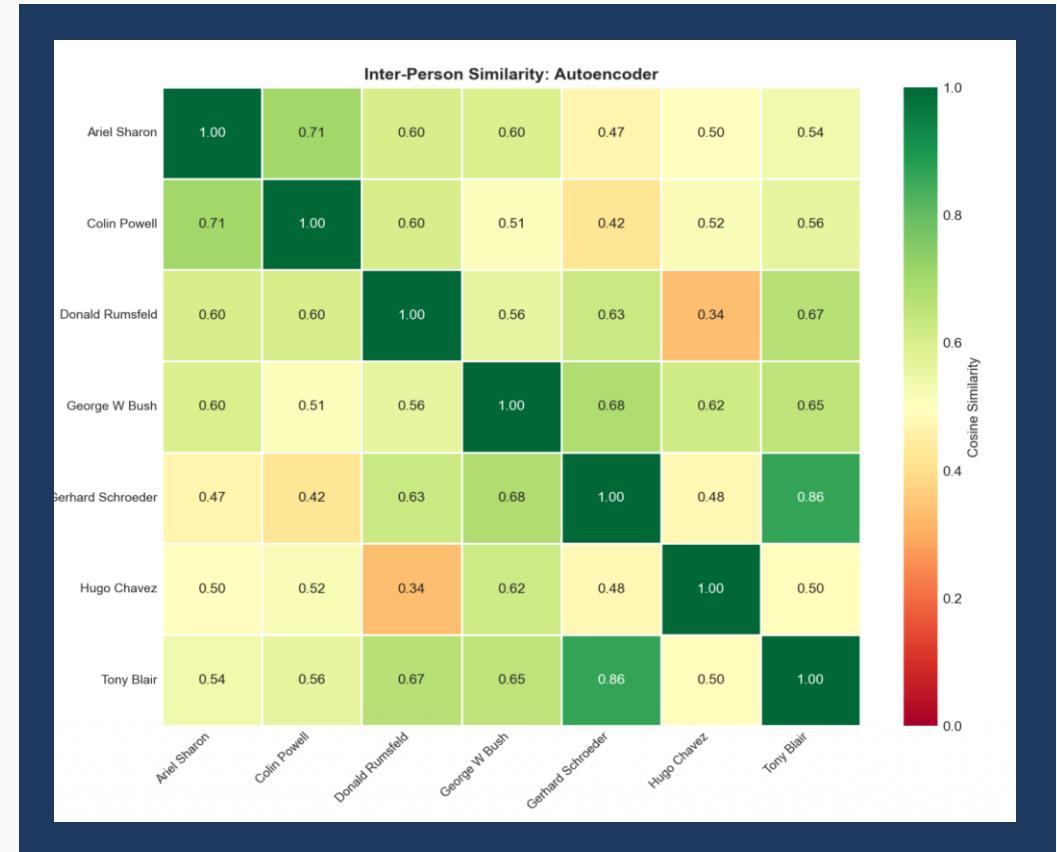
👉 Colori più chiari off-diagonal

👉 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 il potere discriminante

Verifica Facciale: Riepilogo

Confronto completo delle metriche ottenute

Feature	Similarità / Distanza	AUC	EER	Soglia
PCA	Coseno	0.94	8.2%	0.147
PCA	Euclidea	0.91	11.5%	12.3
Autoencoder	Coseno	0.89	12.1%	0.203
Autoencoder	Euclidea	0.86	15.8%	8.7



Migliore

PCA + Similarità del Coseno



Conclusione

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

💡 Il gap tra PCA e AE è più marcato nella verifica che nel riconoscimento facciale, confermando la maggiore rappresentatività delle feature PCA

Sintesi dei Risultati

Punti chiave dell'analisi comparativa

1 Compressione 97%

Da **3008 a 100 dimensioni**, accuratezza > 84%. Dimostrazione dell'efficacia della 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 è la combinazione ottimale. Unisce le caratteristiche migliori di entrambi gli approcci

4 Cosine > Euclidean

Per la verifica facciale, **l'invarianza alla scala dei dati** è fondamentale. La similarità del coseno performa meglio

5 EER 8%

Sistema utilizzabile per sicurezza media

85%

Accuratezza Migliore

Criticità e Sviluppi Futuri

Limitazioni e direzioni di ricerca

⚠ Limitazioni

- ✖ **Autoencoder:** ignora struttura 2D delle immagini
- ⌚ **Dataset limitato:** 1288 samples insufficienti per AE complessi
- ⚖️ **Sbilanciamento:** impatta la sensibilità delle classi minoritarie

🔑 Direzioni Future

- .GridView **Autoencoder convoluzionale:** sfrutta correlazioni spaziali
- Cloud **Transfer Learning:** pre-training su dataset grandi
- Image **Stress Test:** testare i modelli con rumore (gaussiano o S-P)

💡 Prospettiva

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

1

Non sempre non-lineare è meglio di lineare

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

2

La riduzione dimensionale è cruciale

97% compressione con minima perdita

3

Gli Approcci ibridi possono essere vincenti

Rete neurale su feature PCA si è rivelata la ccombinazione ottimale

4

La metrica conta

Similarità del coseno e distanza Euclidea hanno comportamenti diversi. Scegliere in base al task

5

La validazione è essenziale

Valutare **validazione incrociata** e **intervalli di confidenza** per risultati affidabili

Extra: Classificazione sui Dati Originali

Analisi delle prestazioni sulle immagini prima della riduzione dimensionale

I modelli **SVM** e **NN** sono stati **ri-addestrati** (utilizzando i **migliori iperparametri**) sulle **immagini originali** pre-riduzione dimensionale (`X_train`) e ne sono state valutate le prestazioni (`X_test`)

Modello	Accuratezza	Precisione	Sensibilità	F1-Score	ROC-AUC
SVM	0.810	0.816	0.810	0.801	0.983
Rete Neurale	0.848	0.858	0.849	0.850	0.975

💡 Commento

La riduzione dimensionale consente di **mantenere prestazioni comparabili ai dati originali**. La PCA risulta efficace per la Rete Neurale, **migliorando di poco l'accuratezza**, mentre per SVM le **prestazioni restano quasi invariate**, confermando la robustezza dei modelli anche in uno spazio a dimensionalità ridotta

💡 Per osservare in dettaglio la fase di addestramento dei modelli al variare degli iperparametri, consultare il file **log.txt** con l'output del terminale

Bibliografia

Fonti della ricerca

Abdullah, M. I., Islam, M. N., Bala, D., Hossain, M. A., Rahman, M. A., & Islam, M. S. (2025). Efficient Face Detection and Recognition with PCA and Eigenfaces: A Comprehensive Analysis. *International Journal of Advanced Networking and Applications*, 17(1), 6705–6718.

Devi, S., Saini, J., Kaur, H., & Kaur, H. (2024). Face Classification & Recognition Using Stacking Ensemble Learning with Principal Component Analysis Approach. In *2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, Trichirappalli, India: IEEE, pp. 1–8. doi: 10.1109/ICEEICT61591.2024.10718476

Huang, G., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. rep., ott. 2008.

Kayal, S. (2013). Face verification experiments on the LFW database with simple features, metrics and classifiers. In *nDS '13; Proceedings of the 8th International Workshop on Multidimensional Systems*, pp. 1–6.

Meedeniya, D. A., & Ratnaweera, D. A. A. C. (2007). Enhanced face recognition through variation of Principle Component Analysis (PCA). In *2007 International Conference on Industrial and Information Systems*, Peradeniya, Sri Lanka: IEEE, pp. 347–352. doi: 10.1109/ICIINFS.2007.4579200

Wang, Y., Yao, H., & Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomputing*, 184, 232–242. doi: 10.1016/j.neucom.2015.08.104



Grazie per l'attenzione



Codice e documentazione disponibili nel **repository**

<https://git.new/Gb69fEp>