

Face Recognition using PCA

COLAPIETRA Antonio Pio
OUKANI Youssef
TAVILLA Simone

Statistical and Mathematical Methods For AI
Prof. Marina Popolizio

Obiettivo

L'obiettivo di questo progetto è sviluppare un **sistema di riconoscimento facciale** in grado di associare un volto umano all'identità di una persona. Per il conseguimento di tale obiettivo verrà utilizzato un dataset di volti per addestrare degli algoritmi di Machine Learning oltre che varie tecniche per rendere efficiente il nostro task dal punto di vista computazionale.

Applicazioni pratiche:

- **Sicurezza**: sistemi di accesso biometrico, sorveglianza intelligente e identificazione di persone in aree pubbliche.
- **Organizzazione**: catalogazione automatica di foto in archivi digitali per applicazioni personali o professionali.



Sfide da affrontare

Teorema Garbage In Garbage Out:

un sistema informatico produrrà risultati errati o inutili se i dati in ingresso sono di scarsa qualità o inaccurati



Dopo un'attenta ricerca e scelta di un dataset adeguato, ci siamo scontrati con la gestione delle immagini che dovranno essere visualizzate come **matrici di pixel**, ciascuno dei quali rappresenta un colore attraverso valori numerici.



Un'altra sfida da affrontare è la valutazione dell'efficacia della Principal Component Analysis per effettuare una riduzione dell'**elevata dimensionalità** del dataset, preservandone le caratteristiche discriminanti necessarie per l'applicazione di algoritmi di Machine Learning.

Dataset

Per poter conseguire il nostro task nel modo migliore abbiamo scelto un dataset di immagini facciali creato dall'università di Yale chiamato "[Extended Yale Face Database B](#)". Il dataset comprende 2432 immagini di dimensioni 168x192 pixel:

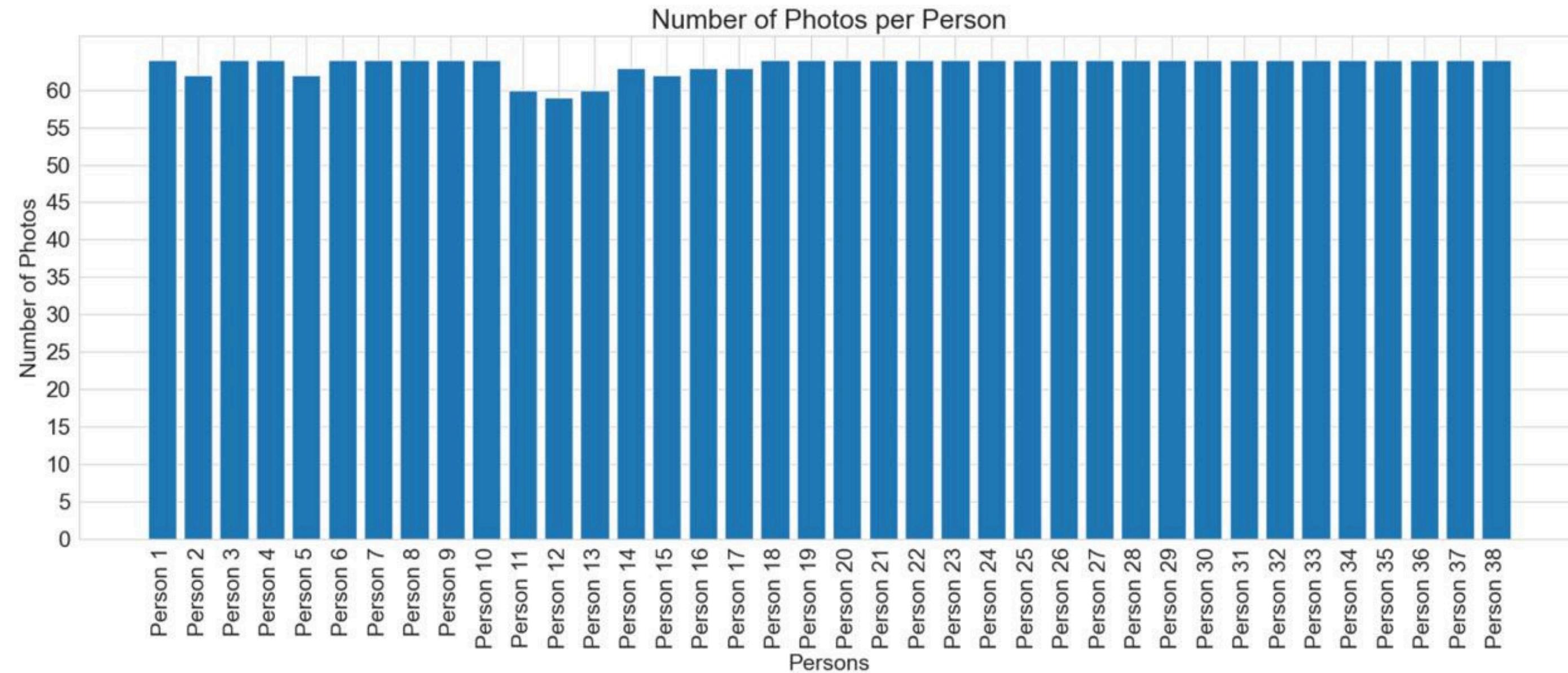
per ciascuno dei 38 soggetti sono state acquisite 64 immagini in 64 condizioni di illuminazione differenti. Tra queste 2432 immagini ci sono 22 immagini completamente nere che sono state aggiunte esclusivamente per permettere la corretta visualizzazione del dataset.



36 dei 38 soggetti considerati

Dataset

Dal grafico sottostante si può notare come varia il numero di immagini valide utilizzate per ciascun soggetto e come il numero di immagini minimo per ogni soggetto sia di 59 immagini.



Dataset



Per facilitare la comprensione dei dati utilizzati è stata riportato qui di fianco una griglia con le 64 immagini per uno dei 38 soggetti considerati. Come già anticipato, alcune di queste immagini sono nere, ma nonostante ciò non sono state considerate come dati.

Dataset

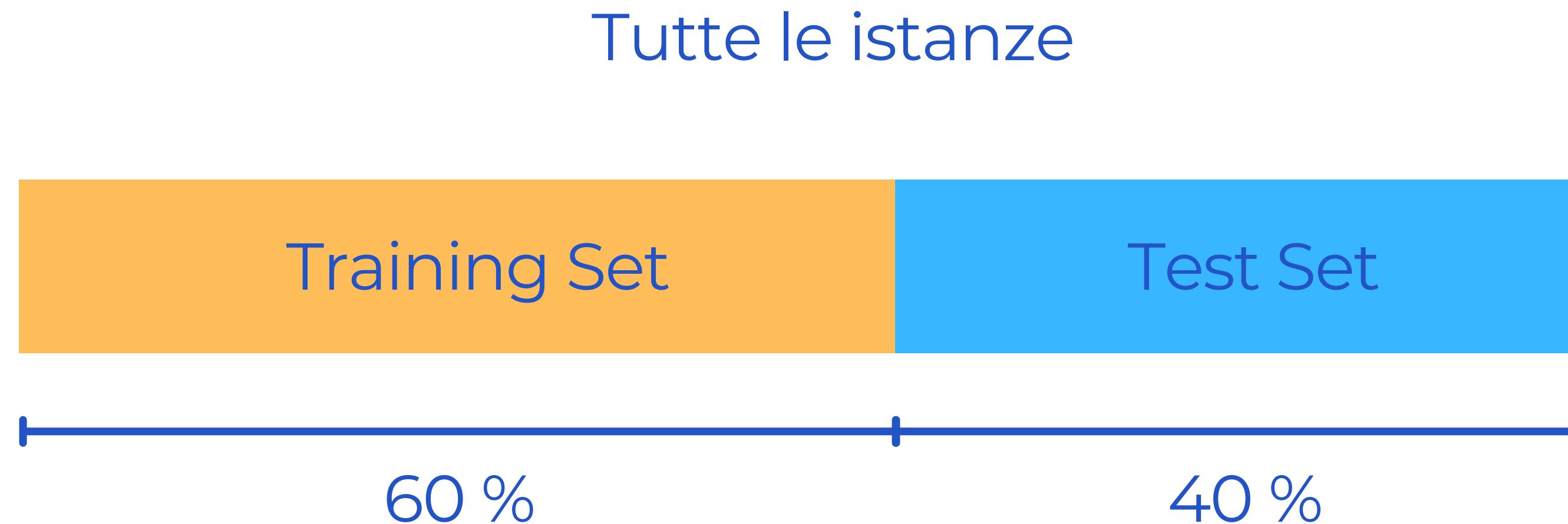
1	2	...	32256
80	11
94
...	2

2410 righe = 64 foto per
ognuno dei 38 individui

32256 colonne= 168 x 192 pixel

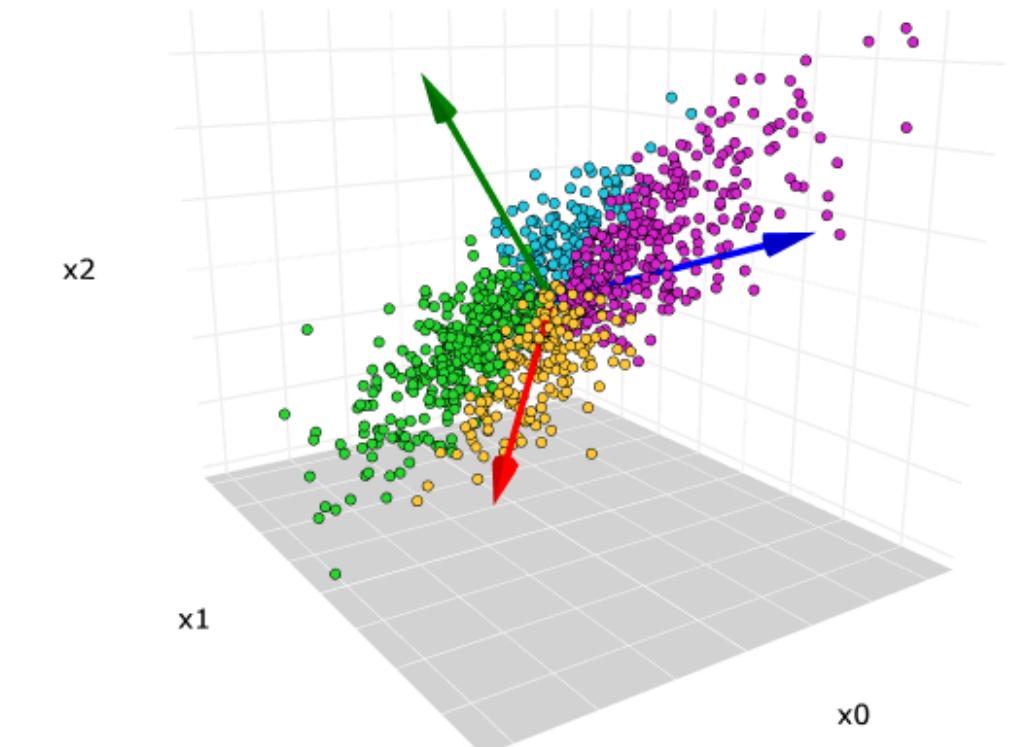
Splitting

Per evitare problemi dovuti al group bias è stato effettuato uno shuffle prima dello splitting. La tecnica di splitting utilizzata è l'Holdout Splitting con Stratify, abbiamo quindi diviso il dataset in due sottosinsiemi: uno per il training (pari al 60% dei dati) e uno per il test (pari al 40%).



Principal Component Analysis

La Principal Component Analysis (PCA) è una tecnica di riduzione della dimensionalità che trasforma un dataset ad alta dimensionalità in un nuovo spazio a dimensione inferiore, preservando la massima varianza possibile. Il concetto chiave è dunque la ricerca di una base ortonormale per i dati, dove le prime componenti principali catturano la maggior parte della varianza del dataset. Tra i concetti fondamentali per la comprensione di questa tecnica ci sono:



Componenti principali

sono le direzioni degli autovettori della matrice di covarianza

Autovalori

rappresentano la quantità di varianza catturata da ciascuna componente

Varianza spiegata

misura la percentuale di informazione originale catturata da ciascuna componente principale

Principal Component Analysis

1. *Determinare le direzioni di massima varianza nei dati (componenti principali).*
2. *Proiettare i dati in un sottospazio di dimensione inferiore.*
3. *Decorrelare le variabili trasformando il dataset in nuove coordinate ortogonali.*
4. *Semplificare l'analisi statistica e migliorare le prestazioni di modelli ML.*

PCA preprocessing

1) Calcolo della media per ogni riga del dataset

$$\bar{x} = \frac{1}{n} \sum_{j=1}^n x_j$$

2) Calcolo Averaged Matrix

$$\bar{X} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{x}$$

3) Media centrata dei dati

$$B = X - \bar{X}$$

PCA

1) Calcolo matrice di Covarianza

$$\mathbf{C} = \mathbf{B}^\top \mathbf{B}$$

2) Calcolo degli autovalori e autovettori

$$\mathbf{C} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{D}$$

3) Fattorizzazione SVD

$$\mathbf{B} = \mathbf{U} \Sigma \mathbf{V}^\top$$

4) Definizione componenti principali

$$\mathbf{T} = \mathbf{B} \cdot \mathbf{V} = \mathbf{U} \Sigma$$

5) Selezione del numero di componenti principali

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i} \geq \text{soglia}$$

Eigenfaces

Le Eigenfaces costituiscono un metodo per il riconoscimento facciale basato sull'analisi delle componenti principali tramite Principal Component Analysis. Si tratta di una tecnica matematica che riduce la dimensionalità delle immagini dei volti, identificando le caratteristiche più rilevanti per distinguere un volto dall'altro. Grazie a questo approccio ogni immagine facciale può essere rappresentata come somma ponderata di queste Eigenfaces, semplificando significativamente il riconoscimento.

Le Eigenfaces, come le Principal Components, catturano le caratteristiche più rilevanti dei volti. Ogni Eigenface rappresenta una parte della variazione nei dati, spiegando ciò che la precedente non ha descritto.

Eigenface 1



Eigenface 2



Eigenface 3



Eigenface 4

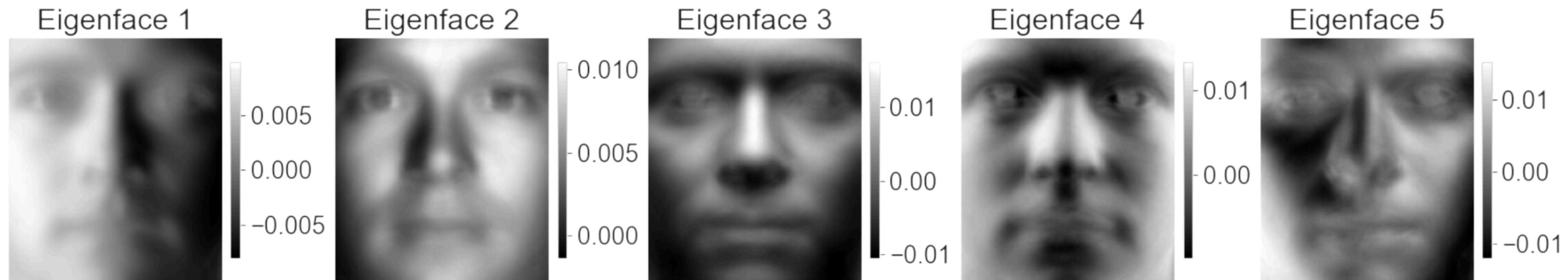


Eigenface 5



Eigenfaces

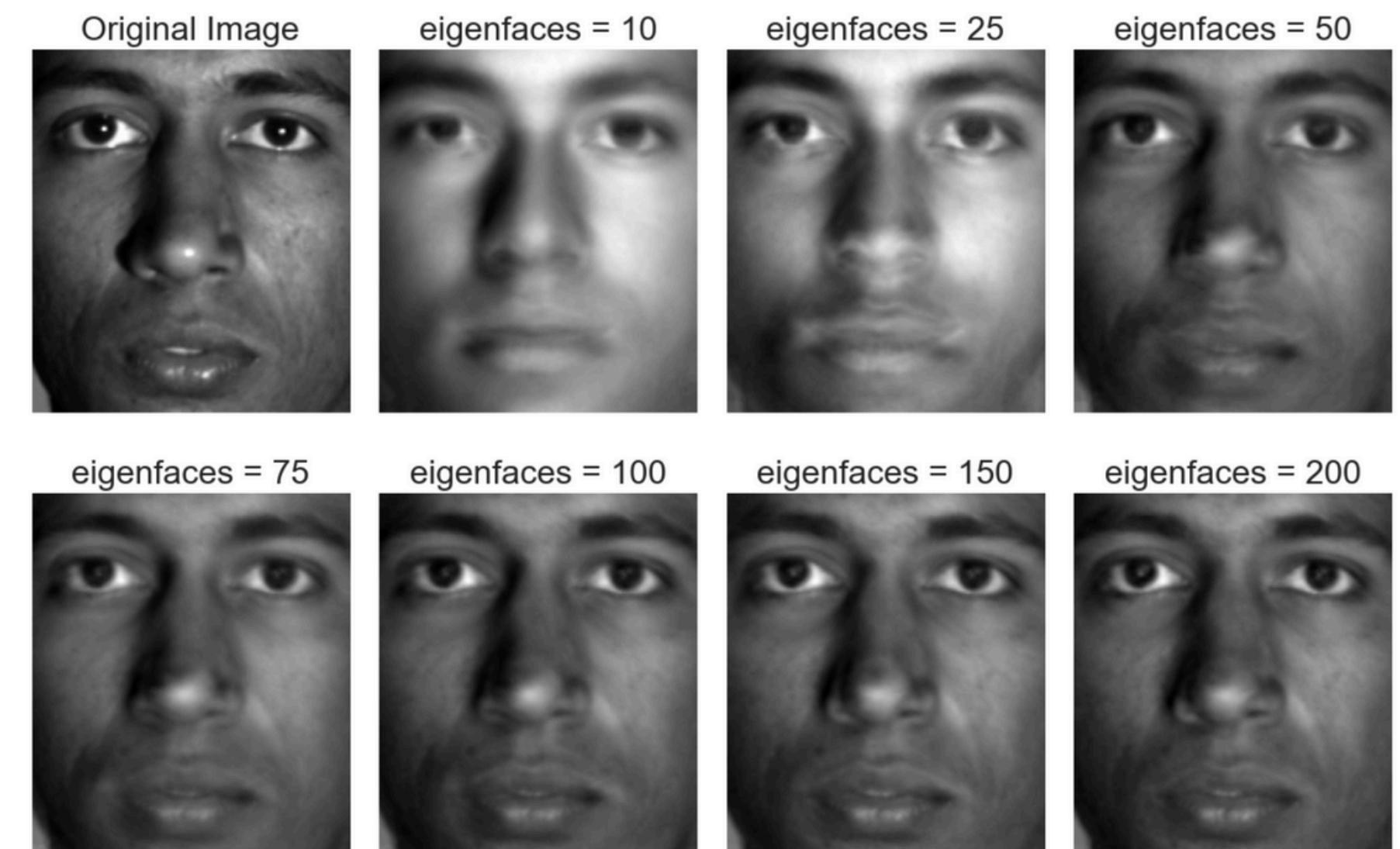
Nell'immagine seguente sono state riportate le prime 5 Eigenfaces ordinate in base alla varianza spiegata. Si può notare come le prime rappresentano le variazioni più significative nei volti, mentre le successive catturano dettagli meno rilevanti coerentemente con la riduzione progressiva della varianza spiegata.



Eigenfaces

Come già spiegato e come si può notare dalla figura, aumentare il numero di Eigenfaces utilizzate per la PCA comporta un aumento delle informazioni rappresentate e quindi avremo un'immagine con maggiore qualità.

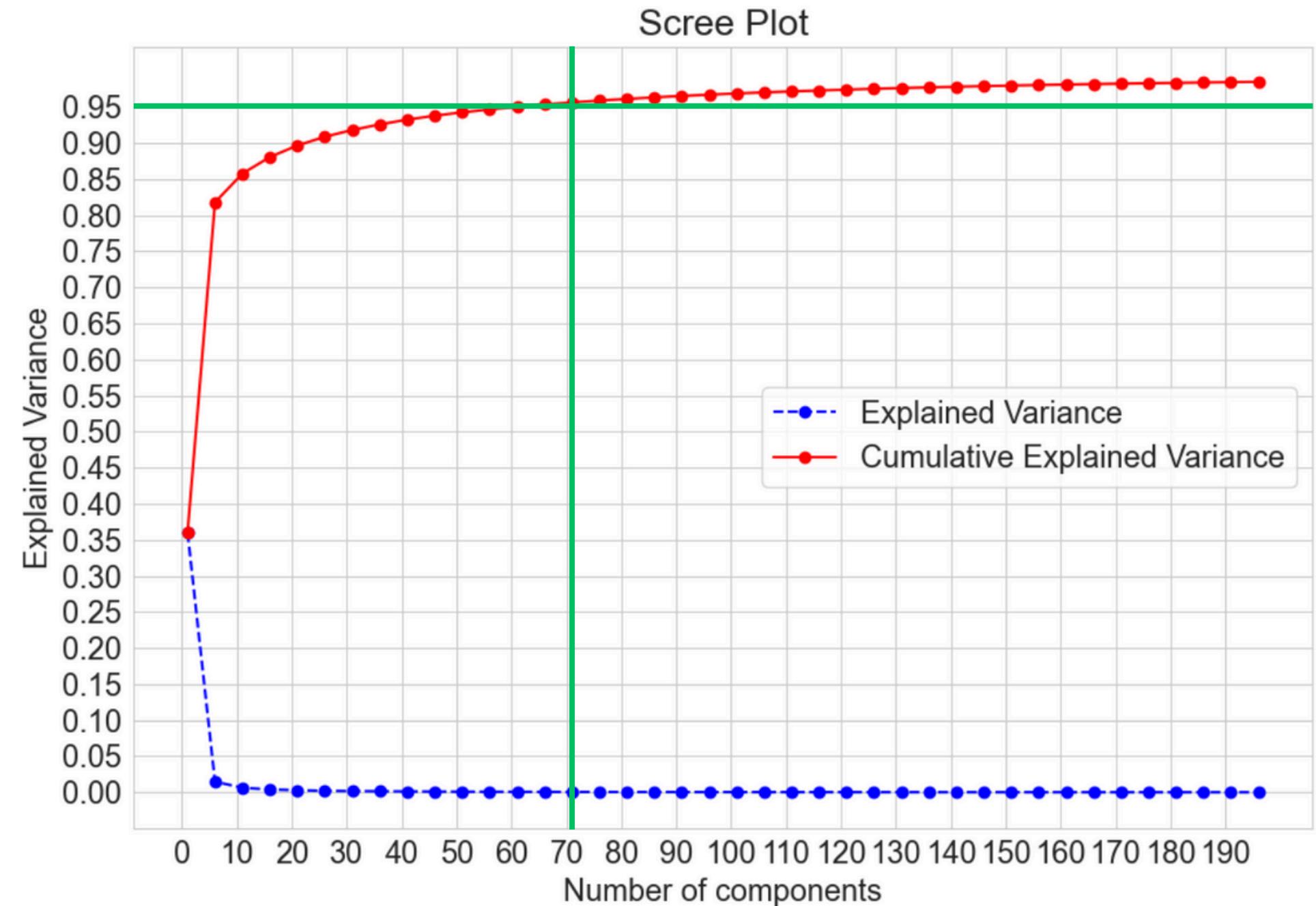
Si è quindi proceduto con una fase di ricerca del trade-off tra qualità e complessità computazionale.



Eigenfaces

Si può notare come la varianza spiegata dalla prima componente principale sia quasi del 35%, mentre per quelle successive la varianza spiegata tende allo 0%. D'altronde si può notare come la combinazione delle componenti principali comporti un aumento della varianza spiegata cumulativa, che ovviamente sarà del 100% solo nel caso in cui il numero di componenti è impostato pari al numero di dimensioni iniziale, ovvero 32256.

La scelta del numero di componenti è stata effettuata considerando [la soglia del 95%](#) della varianza spiegata cumulativa, quindi il numero di componenti scelto è 70.

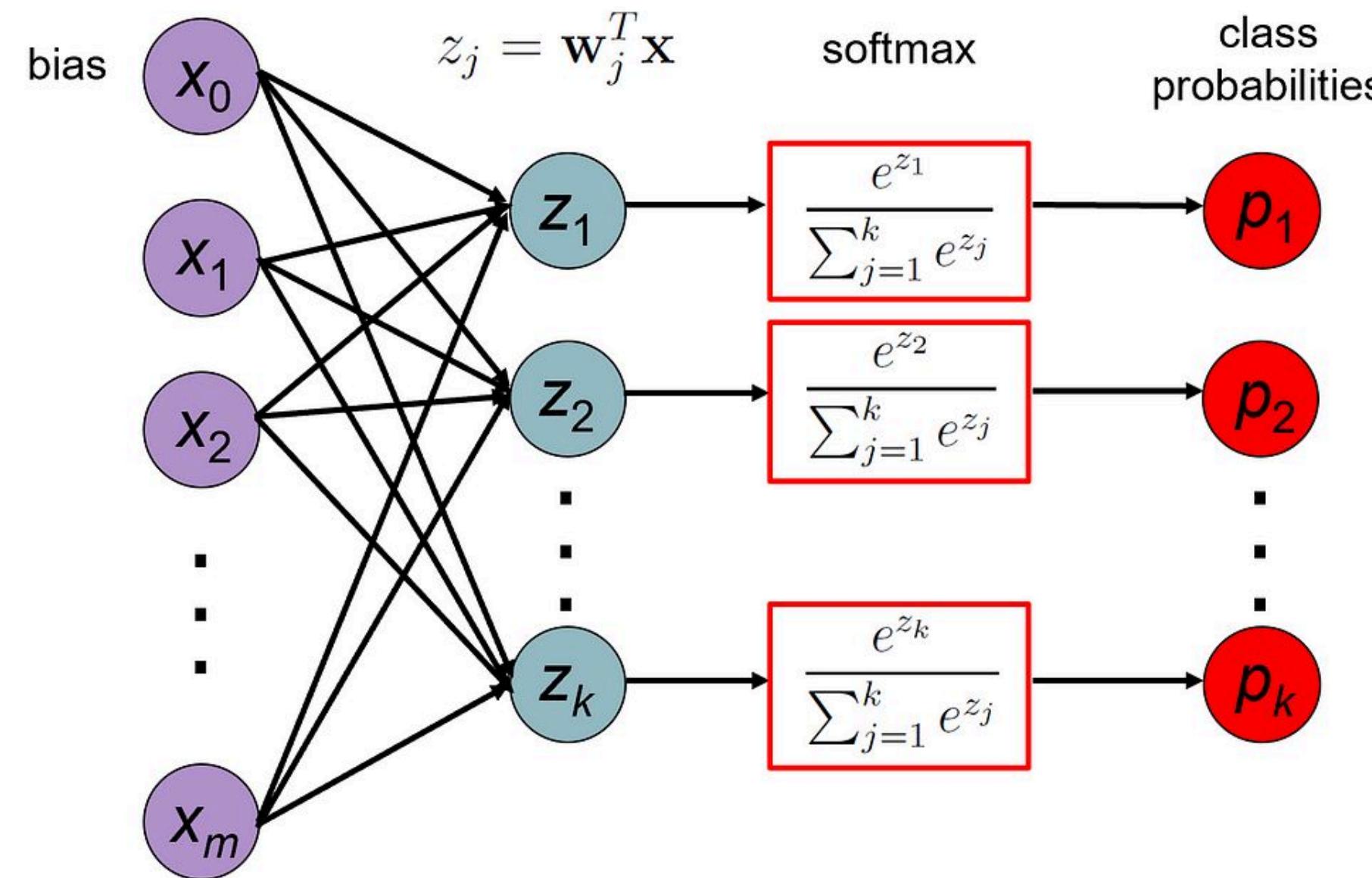


Framework



Modello

Abbiamo utilizzato la **Logistic Regression Multiclasse**, un modello di machine learning per effettuare task di classificazione e produce la probabilità che un dato appartenga a una certa classe.



Logistic Regression

Per estendere la Logistic Regression a C classi abbiamo utilizzato un approccio con Log-SoftMax, implementato con PyTorch. Quindi definiamo la combinazione lineare dei dati in input come:

$$z_k = \theta_k^T x + b_k$$

Successivamente questa combinazione viene trasformata in probabilità logaritmiche attraverso la funzione **Log-SoftMax**, dove C è il numero di classi da predire:

$$\text{Log-Softmax}(z_k) = z_k - \log \left(\sum_{j=1}^C e^{z_j} \right)$$

Per l'addestramento del modello, abbiamo utilizzato la **Negative Log-Likelihood** che calcola la perdita per una sola classe target (quella corretta) ed è definita come:

$$\text{NLL} = \mathcal{L} = - \sum_{i=1}^N \left(z_{\text{target},i} - \log \sum_{j=1}^C e^{z_{j,i}} \right)$$

Logistic Regression

L'ottimizzazione è stata gestita con **discesa del gradiente**: $\theta_j = \theta_j - \alpha \frac{\partial \mathcal{L}}{\partial \theta_j} \quad j = 0, \dots, n$

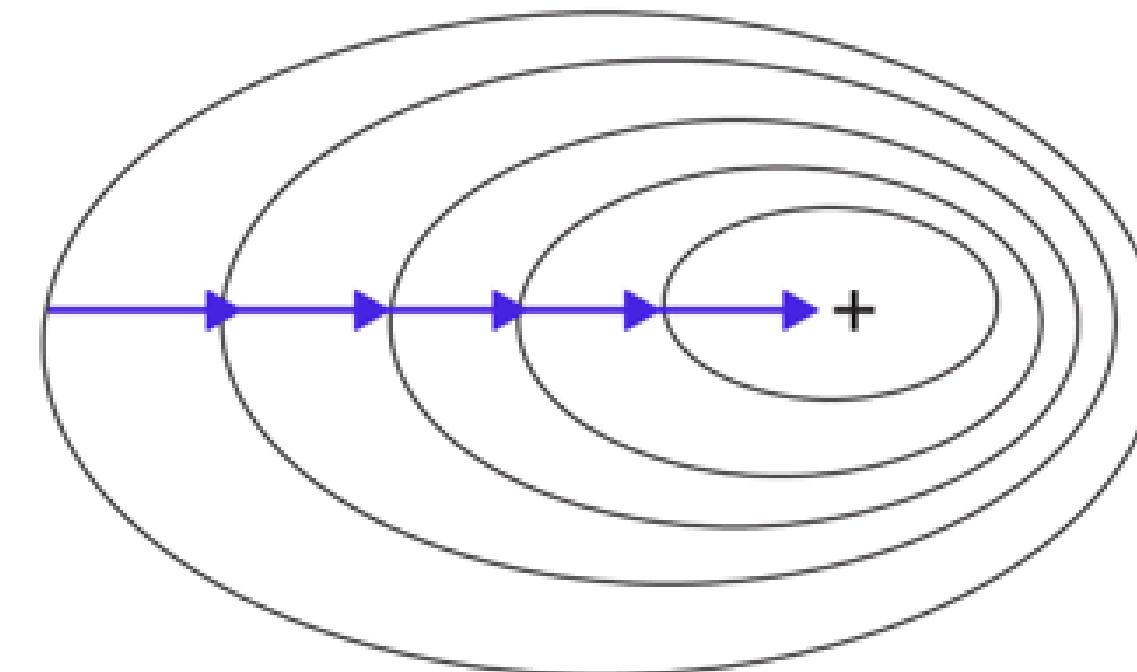
La derivata parziale può essere sviluppata con la **regola della catena**: $\frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{\partial \mathcal{L}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \theta_j}$

ed esplicitando i due termini: $\frac{\partial \mathcal{L}}{\partial z_k} = \text{Log-Softmax}(z_k) - y_k \rightarrow \frac{\partial \mathcal{L}}{\partial \theta_j} = (\text{Log-Softmax}(z_k) - y_k) \cdot x$

Full-Batch

Il Full Batch Gradient Descent è un algoritmo di ottimizzazione che, utilizzando l'intero dataset per calcolare il gradiente della funzione di costo ad ogni iterazione, garantisce una **convergenza stabile** e una maggiore precisione, ma risulta **lento** e richiede elevate risorse computazionali, soprattutto per dataset di grandi dimensioni.

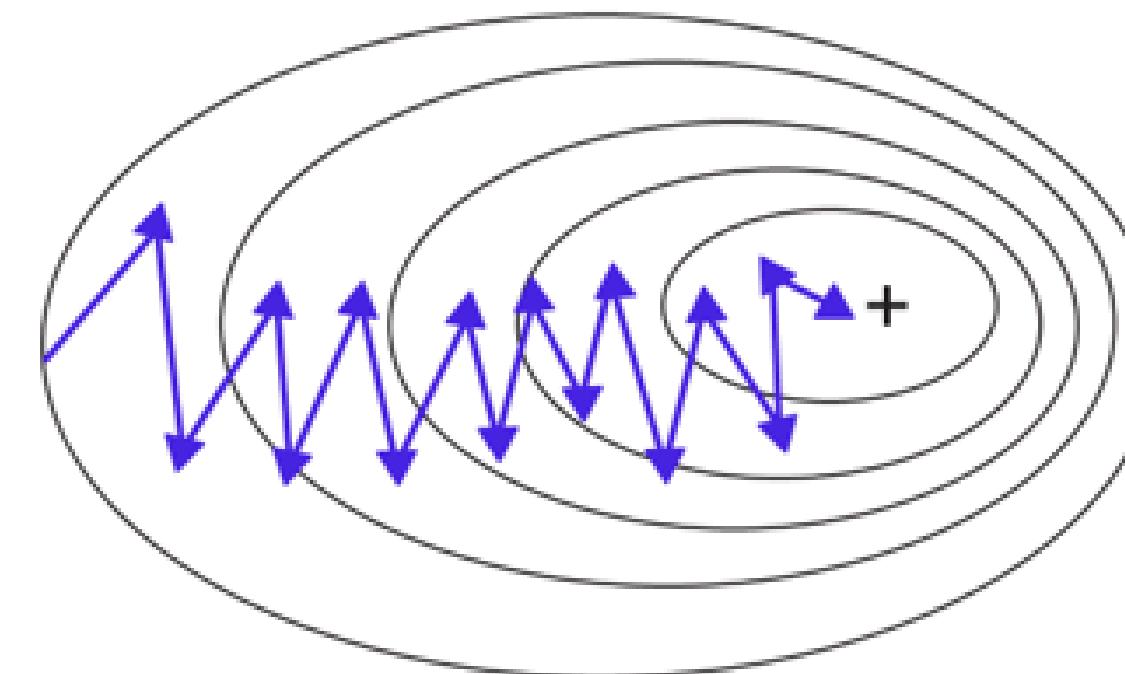
Batch Gradient Descent



SGD

Lo Stochastic Gradient Descent calcola il gradiente su un singolo campione per iterazione, **accelerando l'aggiornamento** dei pesi e riducendo il consumo di memoria, ma introduce **variabilità** nel percorso di convergenza e può richiedere più epoch per raggiungere un minimo, senza garantire la convergenza globale.

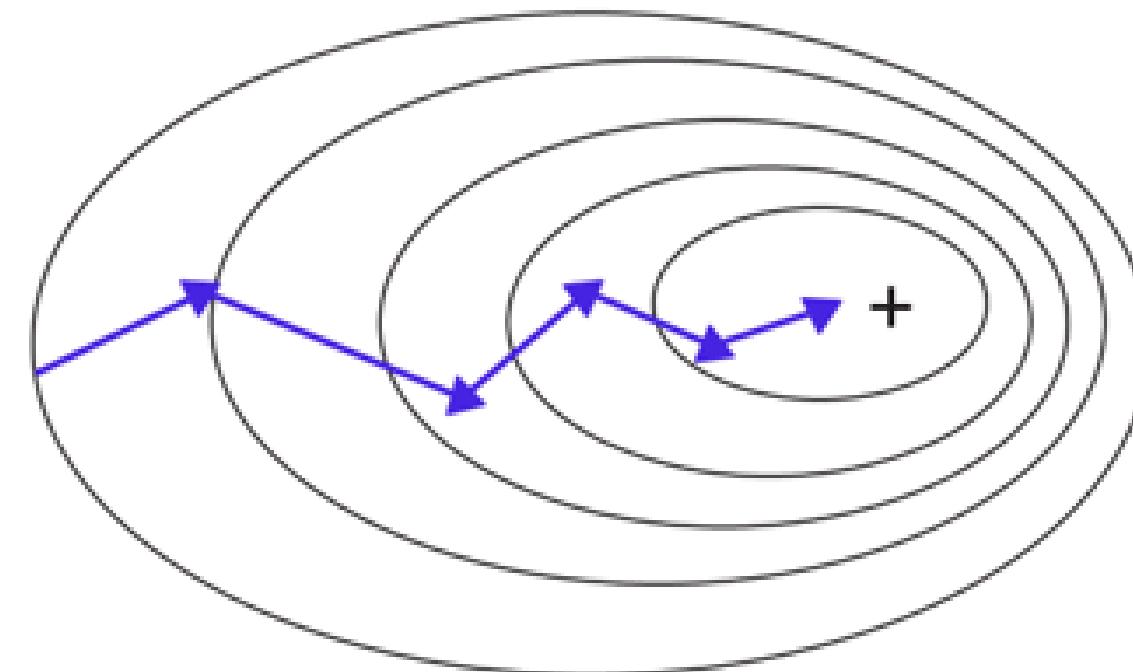
Stochastic Gradient Descent



Mini-Batch

Il Mini Batch Gradient Descent calcola il gradiente su un sottinsieme di dati per iterazione, **bilanciando precisione e velocità** grazie ad aggiornamenti più frequenti e un'efficienza computazionale ottimizzata, ma la sua convergenza dipende dalla dimensione del mini-batch e potrebbe non raggiungere il minimo globale.

Mini-Batch Gradient Descent



Matrici di confusione

Le matrici di confusione sono uno strumento essenziale per valutare le prestazioni di un modello di classificazione, in particolare nei problemi di classificazione multiclasse, infatti permettono di identificare le classi su cui il modello è meno preciso così da poter intervenire con strategie di miglioramento. Nello specifico viene effettuato un confronto tra le etichette reali e le etichette predette per tutte le classi in forma tabellare.

Per un problema con n classi, la matrice ha dimensione $n \times n$, con le righe che rappresentano le classi reali e le colonne quelle predette. Gli elementi sulla diagonale principale indicano i casi correttamente classificati, i restanti elementi sono errori di classificazione.

		Predicted value		
		Positive	Negative	
Actual value	Positive	TP	FN	
	Negative	FP	TN	
Positive	TP	FN	Negative	
	FP	TN		

Metriche di valutazione

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

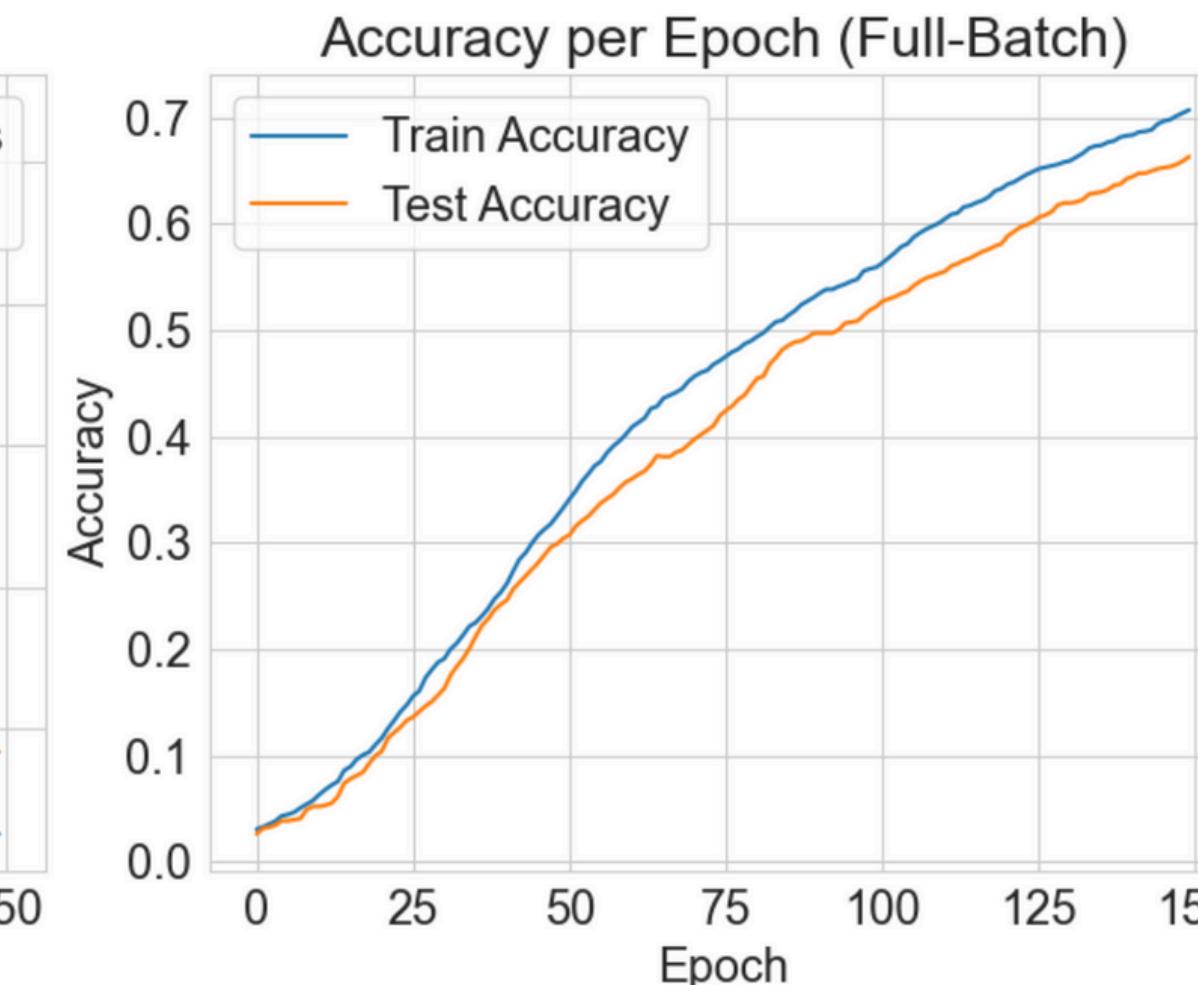
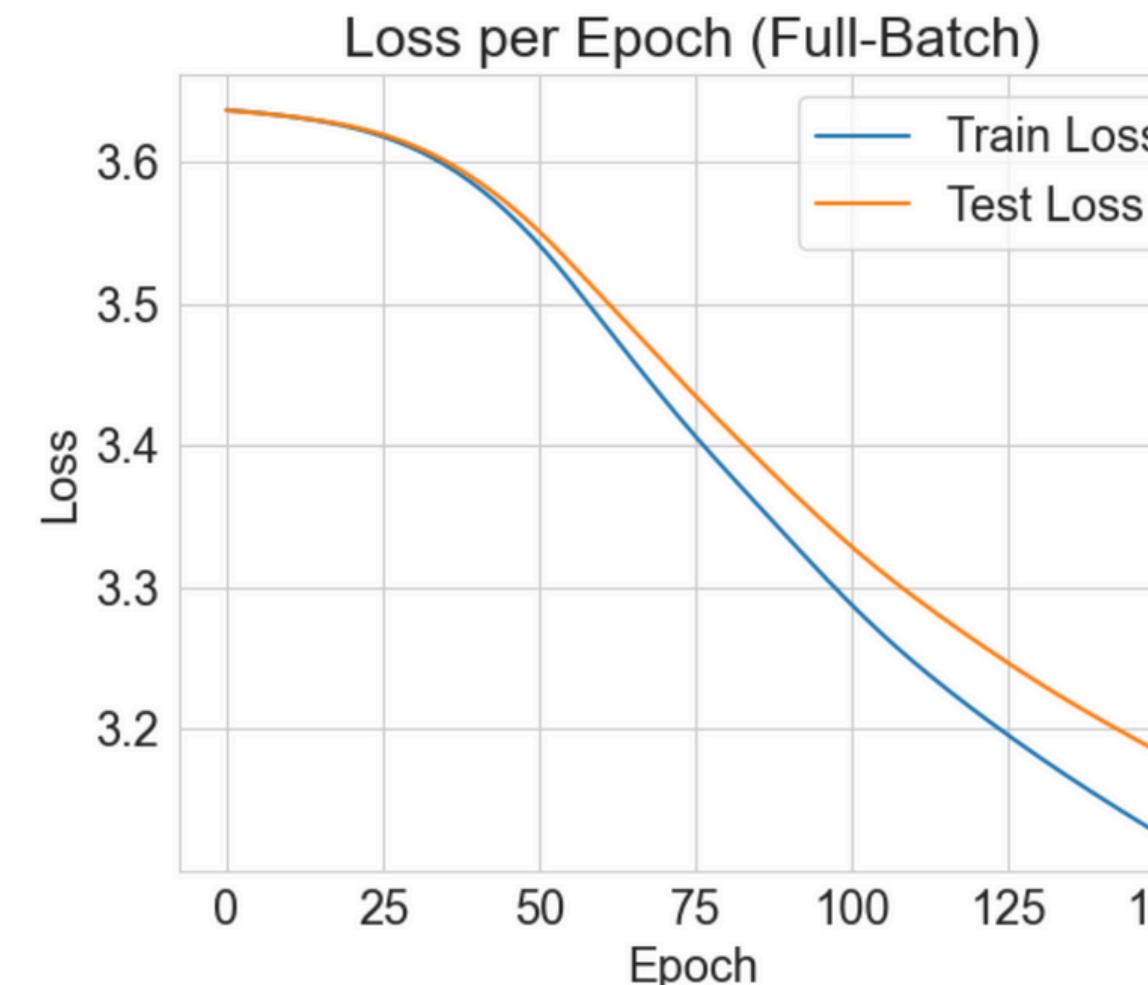
$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Analisi dei risultati

Logistic Regression con PCA e approccio **Full-Batch**

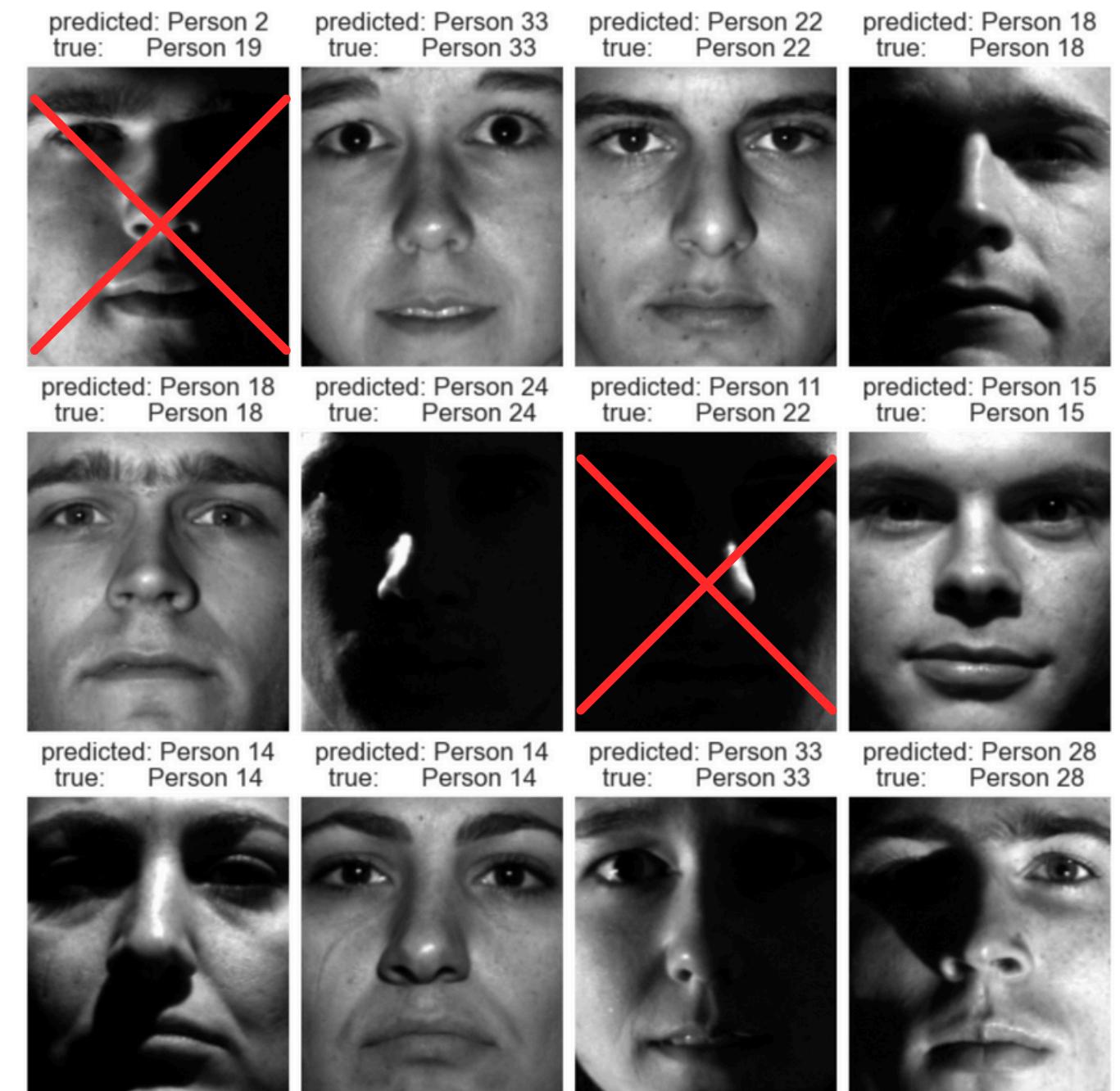
Loss function

Dai grafici sottostanti si può notare come l'andamento della loss e dell'accuracy siano coerenti con i risultati teorici. La loss sul test set risulta maggiore di quella sul train set con un gap che le rende quasi parallele, sintomo di un modello non affetto da overfitting, inoltre non ci sono oscillazioni anomale. Le curve dell'accuracy crescono in modo progressivo e regolare, indicando che il modello sta apprendendo senza instabilità, inoltre il divario tra accuracy di training e test è limitato, suggerendo che il modello generalizza bene senza eccessivo overfitting.



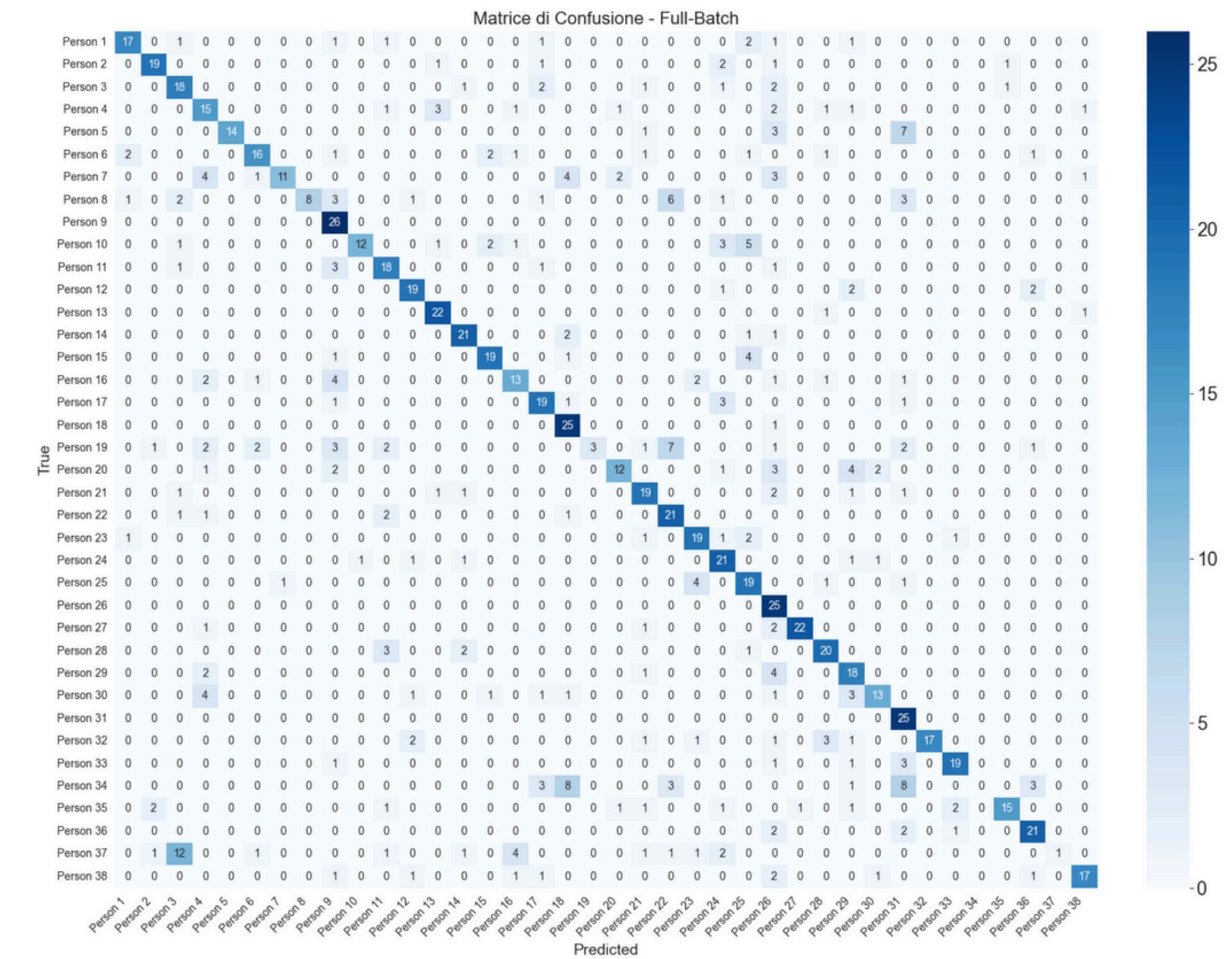
Predizioni sul test set

Nell'immagine di fianco sono state riportate 12 immagini prelevate dal test set e le relative predizioni effettuate dal modello. Si può notare come i risultati delle predizioni siano soddisfacenti dato che solo 2 immagini su 12 sono state predette in modo errato.



Matrice di confusione

Si può notare come la maggior parte dei valori alti si trova sulla diagonale principale, infatti il modello ha classificato correttamente la maggior parte degli esempi. Nonostante ciò sono presenti alcune previsioni errate (valori fuori dalla diagonale), ma il numero di errori è relativamente contenuto, inoltre alcune righe mostrano una maggiore dispersione, suggerendo che alcune classi vengono confuse con altre più frequentemente.

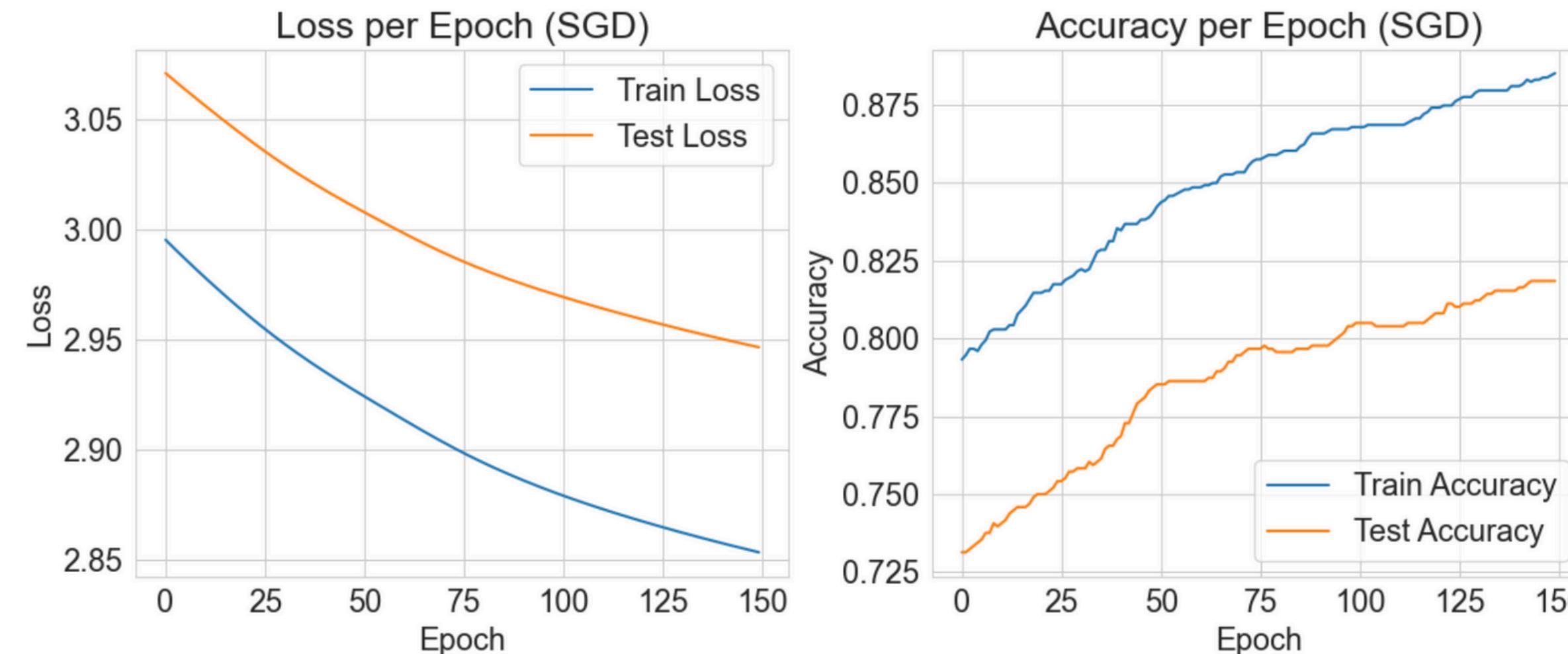


Analisi dei risultati

Logistic Regression con PCA e approccio **SGD**

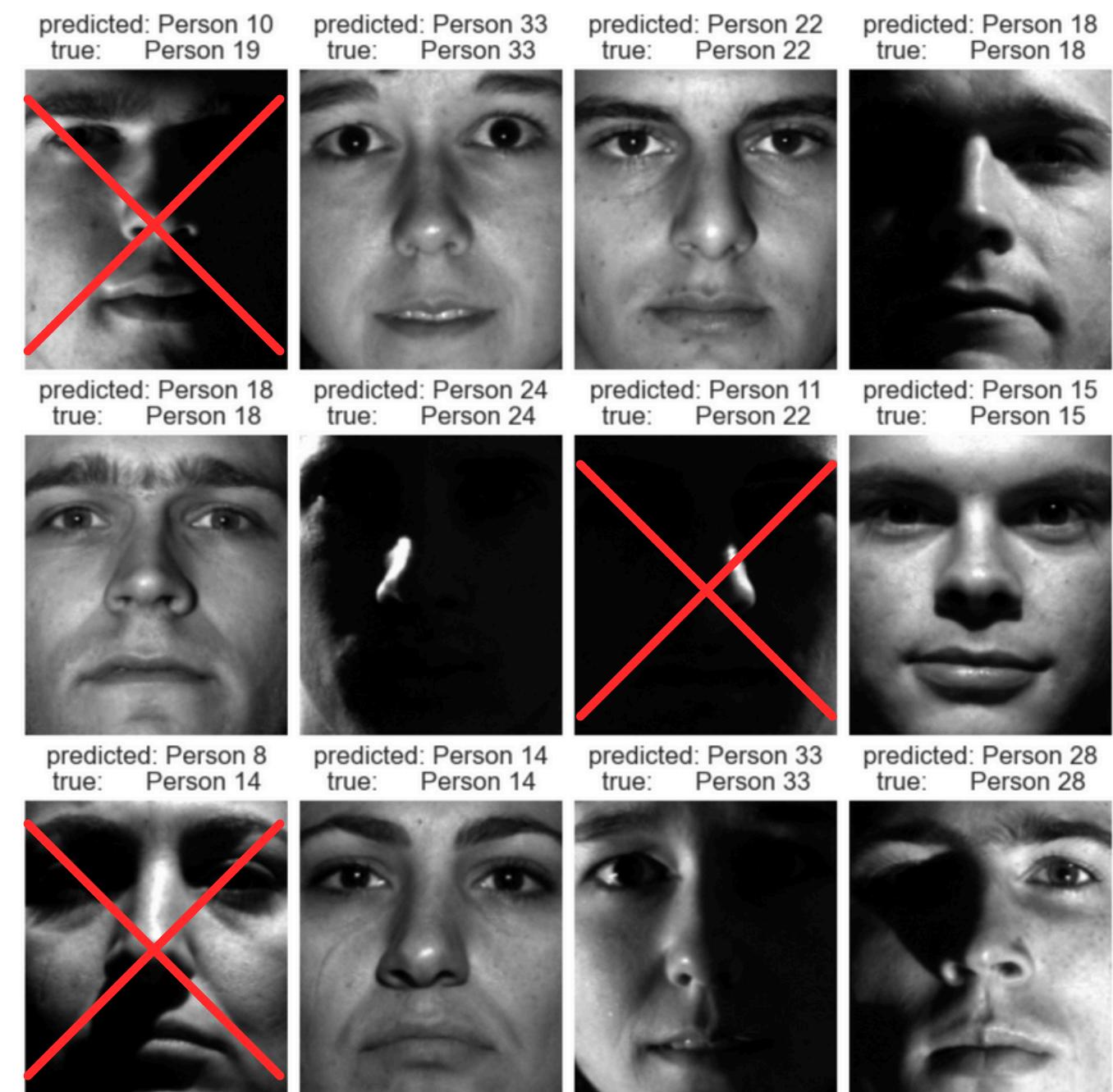
Loss function

Così come per il Full-Batch, si può notare come l'andamento della loss e dell'accuracy siano coerenti con i risultati teorici. La loss sul test set risulta maggiore di quella sul train set ma con un gap maggiore rispetto al caso con Full-Batch, inoltre non ci sono oscillazioni anomale o segni di divergenza. Le curve dell'accuracy crescono in modo progressivo e regolare, anche se possiamo notare come si raggiunga una convergenza per l'accuracy di test attorno all'82% di accuracy. Il divario tra accuracy di training e test è limitato ma maggiore rispetto al caso con Full-Batch, quindi il modello generalizza bene e non c'è overfitting.



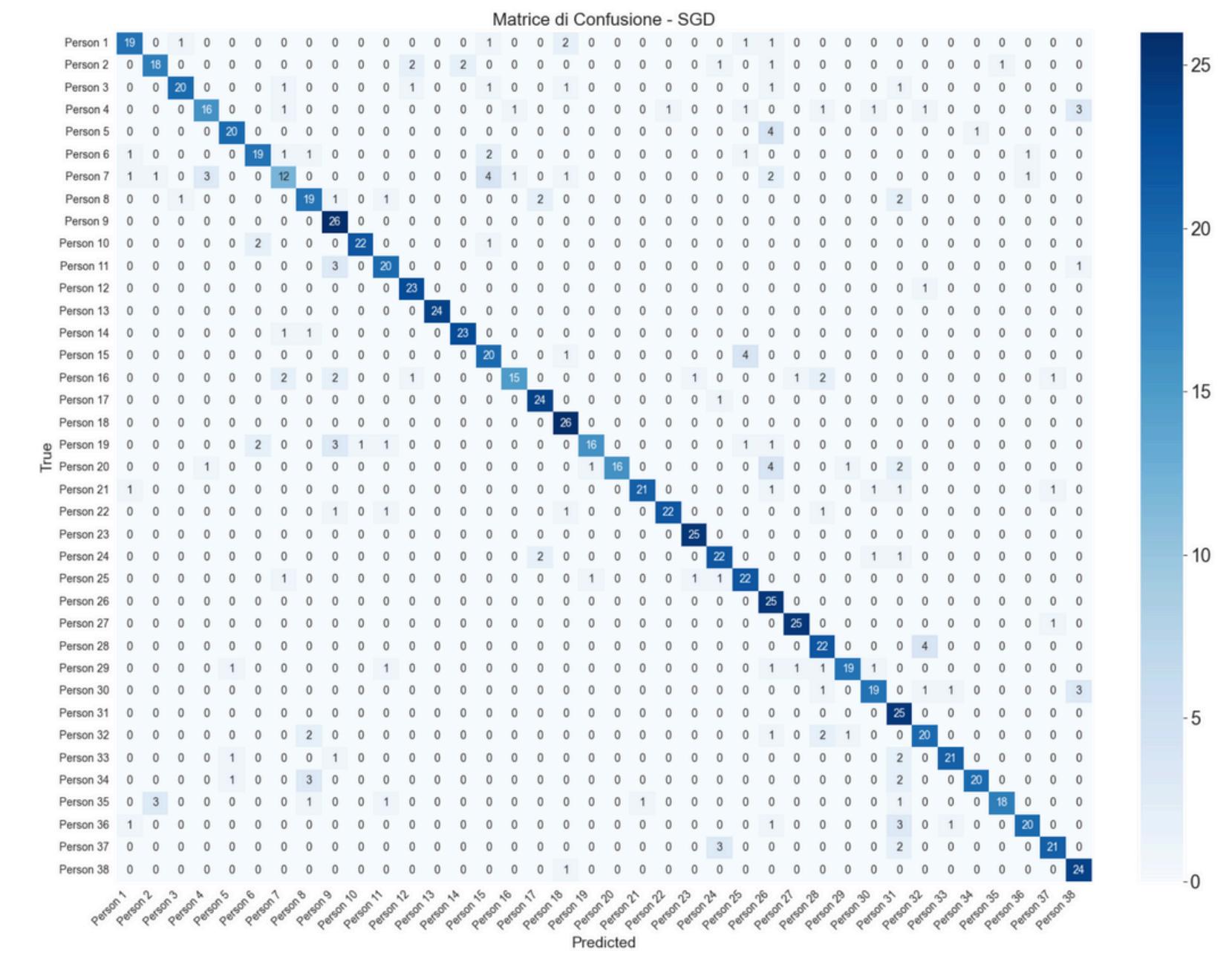
Predizioni sul test set

Nell'immagine di fianco sono state riportate 12 immagini prelevate dal test set e le relative predizioni effettuate dal modello. Si può notare come i risultati delle predizioni siano soddisfacenti dato che solo 3 immagini su 12 sono state predette in modo errato.



Matrice di confusione

Si può notare come la maggior parte dei valori alti si trova sulla diagonale principale, infatti il modello ha classificato correttamente la maggior parte degli esempi. Nonostante ciò sono presenti alcune previsioni errate (valori fuori dalla diagonale) ma il numero di errori è contenuto e minore rispetto all'approccio con Full-Batch.

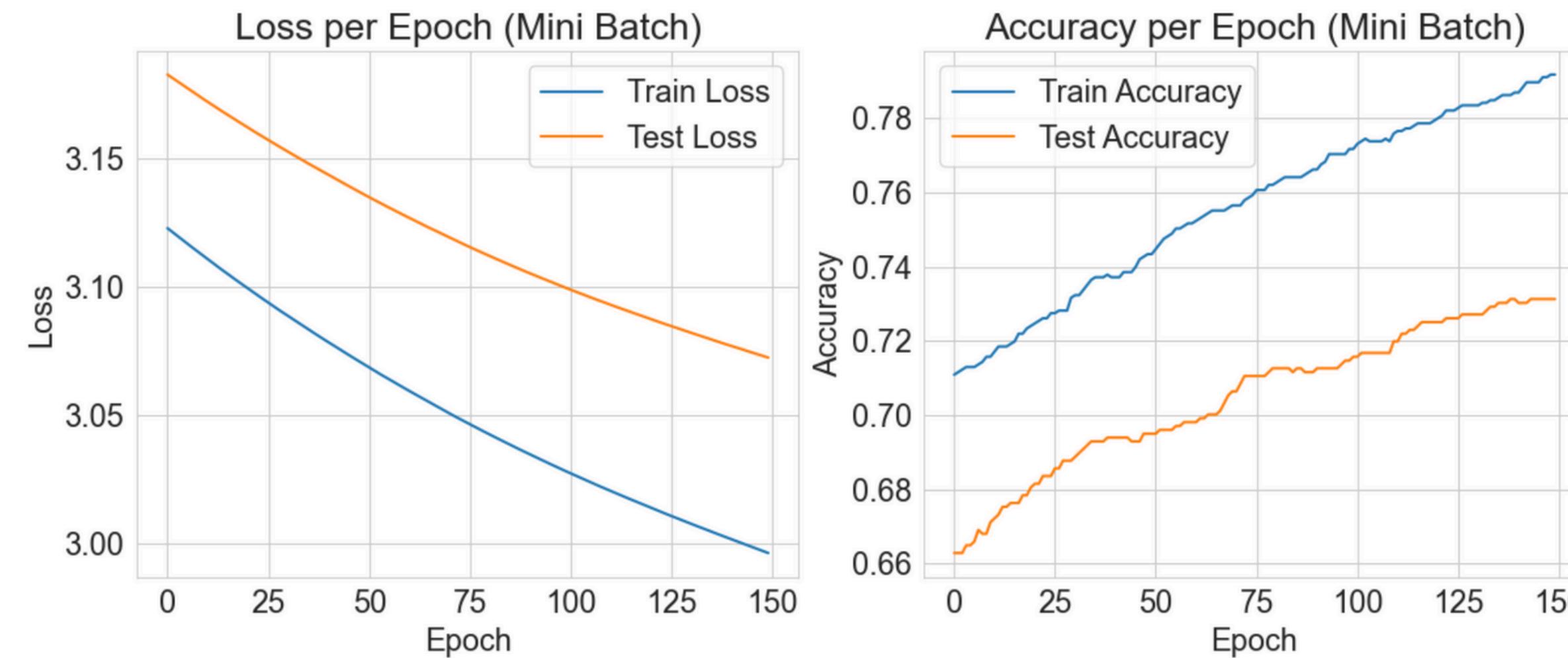


Analisi dei risultati

Logistic Regression con PCA e approccio **Mini-Batch**

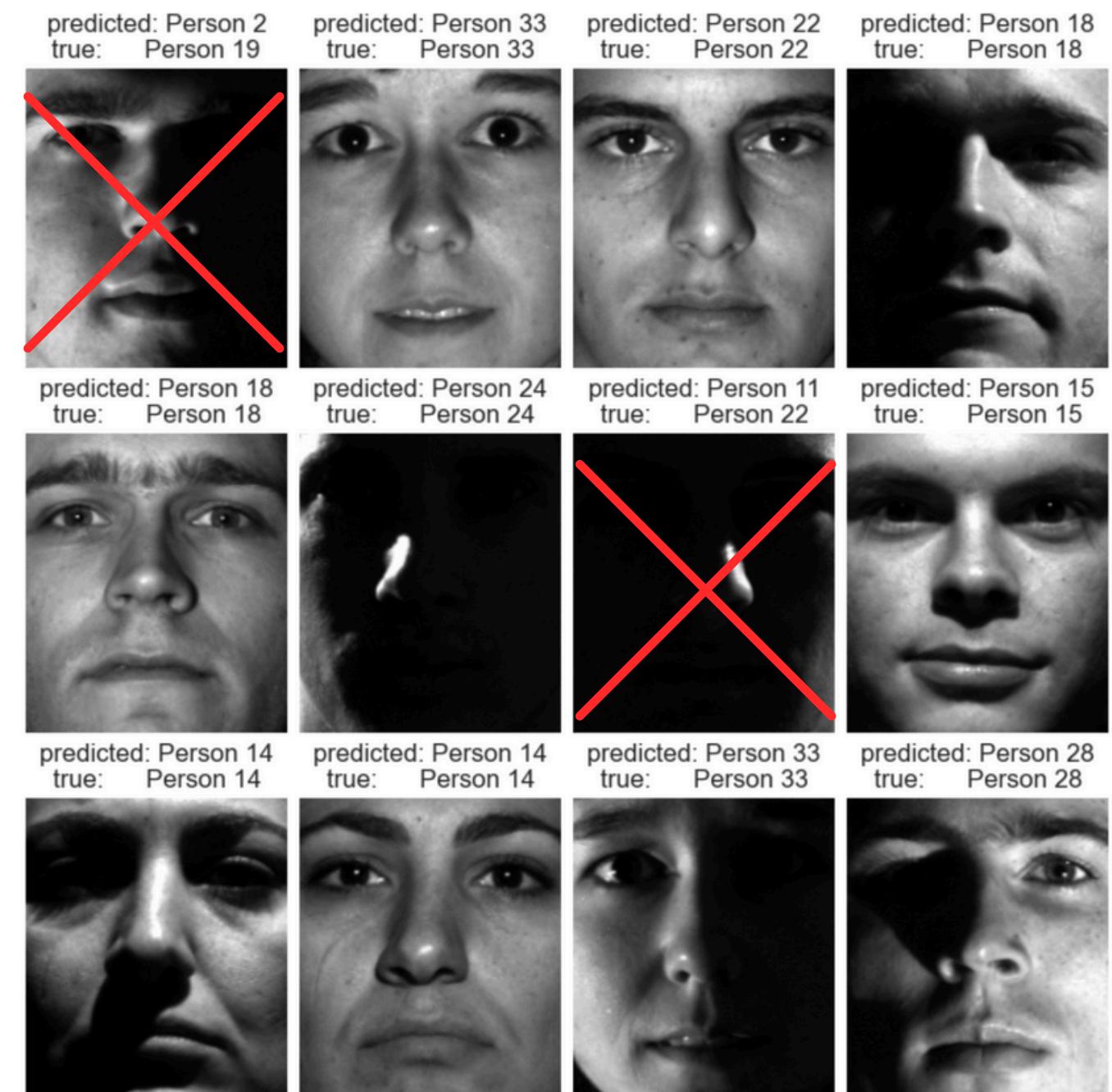
Loss function

Si osserva una costante diminuzione della loss per entrambi i set, tuttavia la perdita sul test rimane più alta rispetto a quella del training. L'accuratezza aumenta progressivamente per entrambi i set, con il training che raggiunge valori più elevati rispetto al test, il che potrebbe indicare un certo overfitting. Il comportamento generale è coerente con l'atteso miglioramento delle prestazioni del modello, ma la distanza tra training e test suggerisce che potrebbe essere utile regolarizzare meglio il modello.



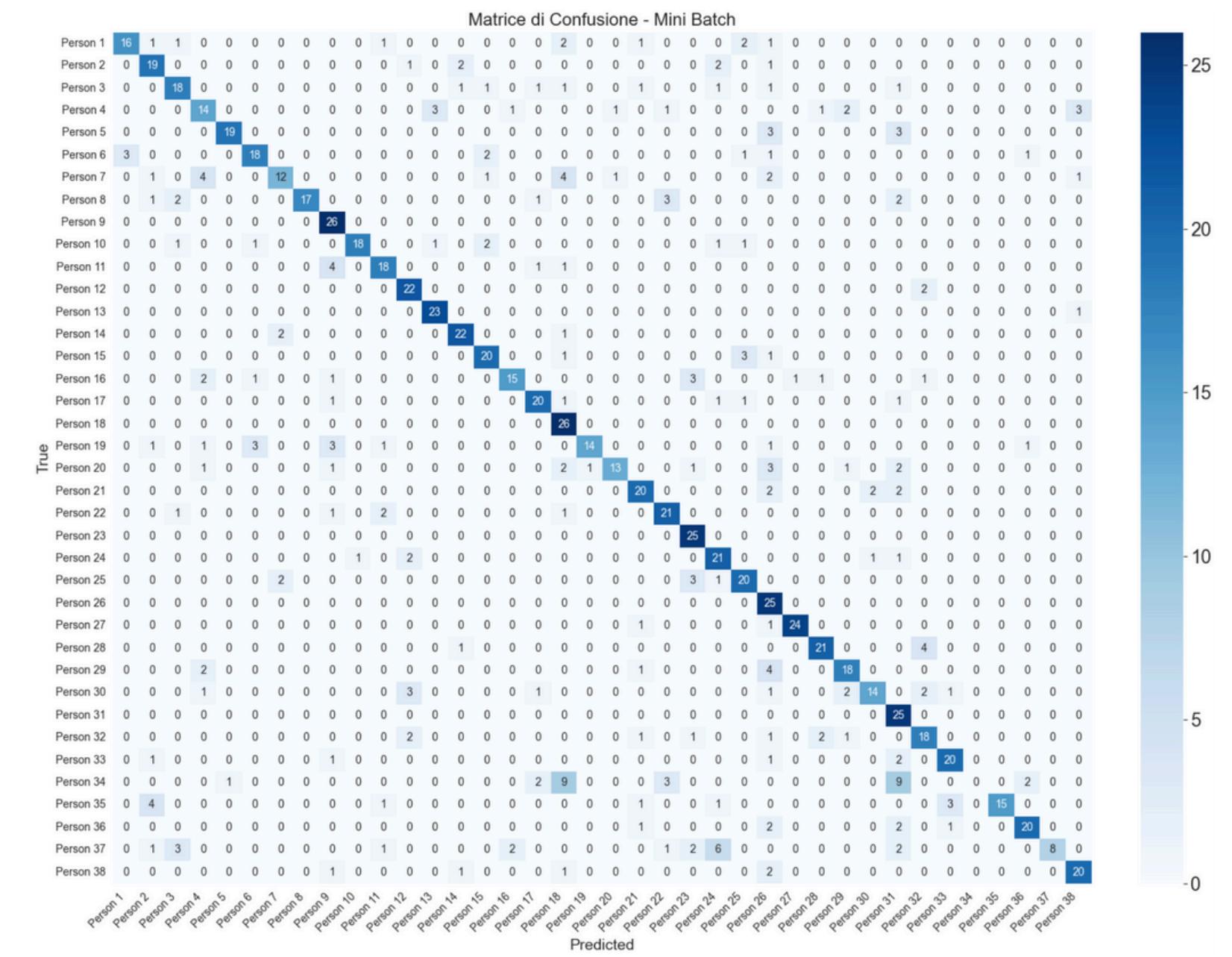
Predizioni sul test set

Nell'immagine di fianco sono state riportate 12 immagini prelevate dal test set e le relative predizioni effettuate dal modello. Anche in questo caso i risultati delle predizioni siano soddisfacenti dato che solo 2 immagini su 12 sono state predette in modo errato.



Matrice di confusione

Si può notare come la maggior parte dei valori alti si trova sulla diagonale principale, infatti il modello ha classificato correttamente la maggior parte degli esempi. Nonostante ciò sono presenti varie previsioni errate (valori fuori dalla diagonale), ma il numero di errori è contenuto, similmente agli approcci con Full-Batch e SGD.



Confronto dei risultati

Dall'analisi comparativa riportata di seguito si può notare come l'SGD, a parità di epoche, ottenga le migliori prestazioni, seguito dal Mini Batch e infine dal Full Batch. Questo risultato è coerente con la teoria, secondo cui l'SGD, grazie all'aggiornamento più frequente dei parametri, permette di raggiungere minimi locali più rapidamente e con maggiore efficacia, migliorando la generalizzazione del modello. Il Mini Batch, rappresentando un compromesso tra SGD e Full Batch, mostra un miglioramento significativo rispetto al Full Batch, il quale risulta meno performante probabilmente a causa della sua minore capacità di adattamento durante l'addestramento.

Metodo	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)
Full Batch	0,69	0,73	0,69	0,68
Mini Batch	0,76	0,77	0,76	0,75
SGD	0,83	0,85	0,83	0,83

T-Test

Il T-Test è una procedura statistica il cui fine è stabilire se la differenza tra una coppia di osservazioni è 0, dove le osservazioni ad esempio possono essere tutti i valori delle accuracy ottenuti da due modelli differenti variando le soglie di decisione.

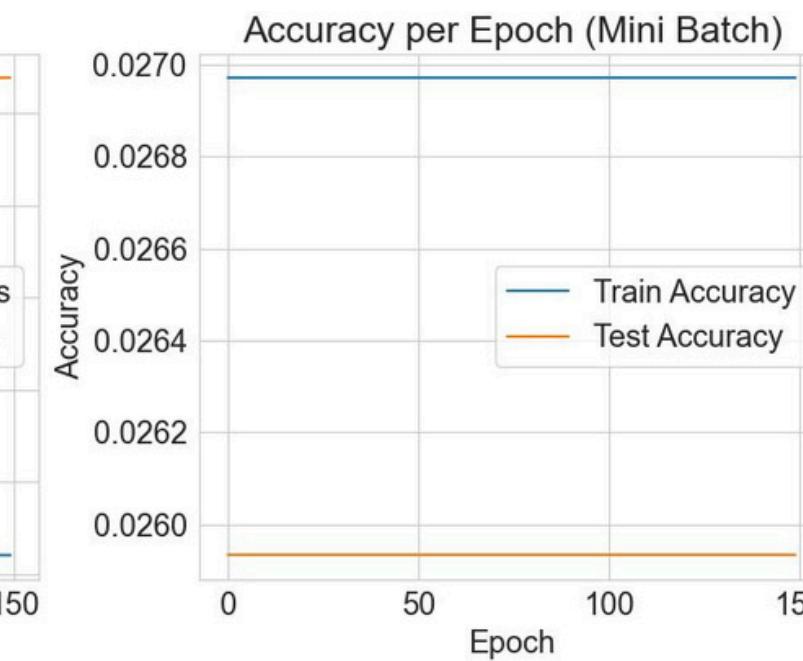
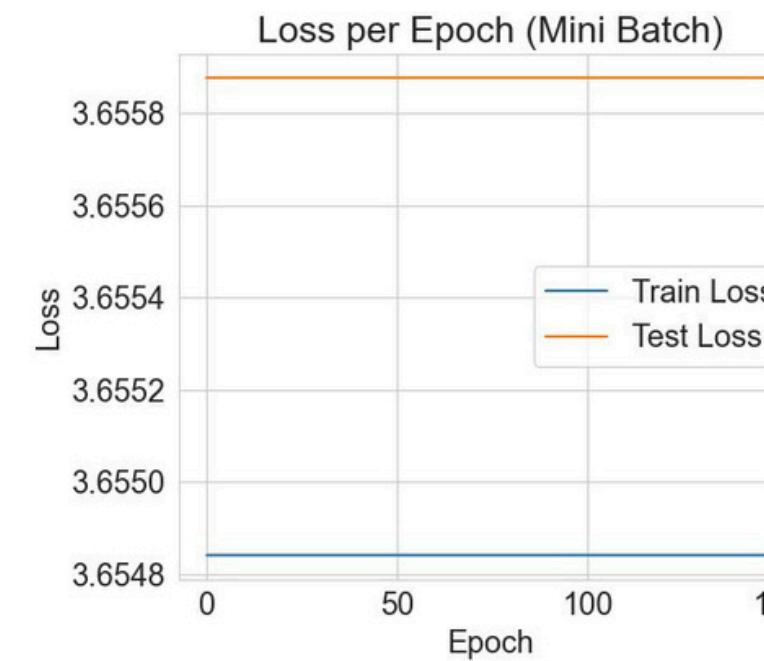
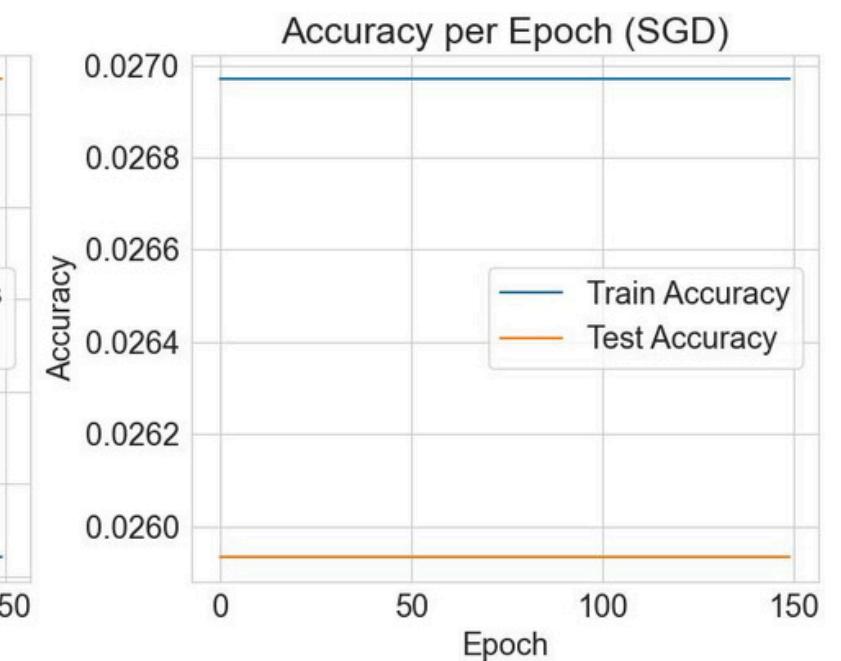
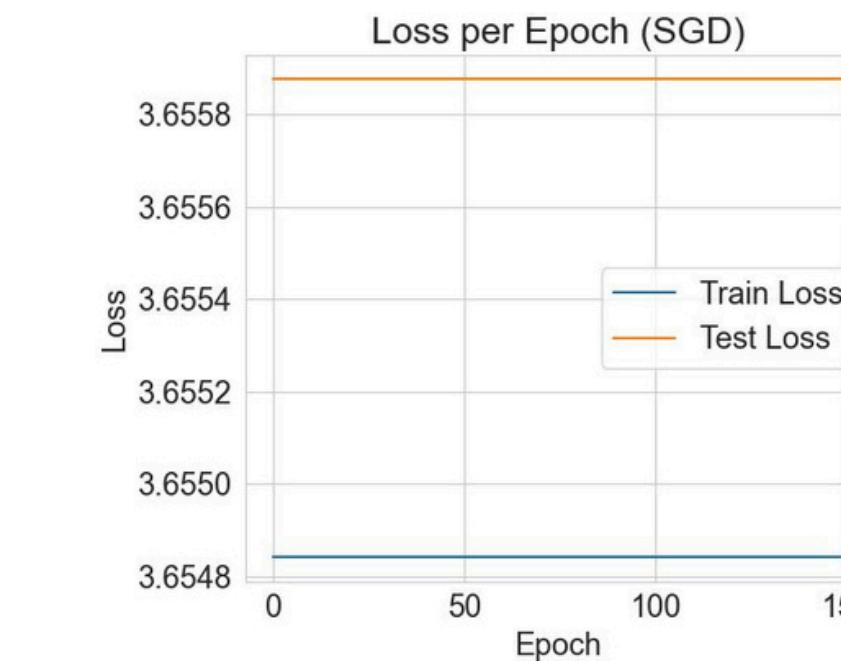
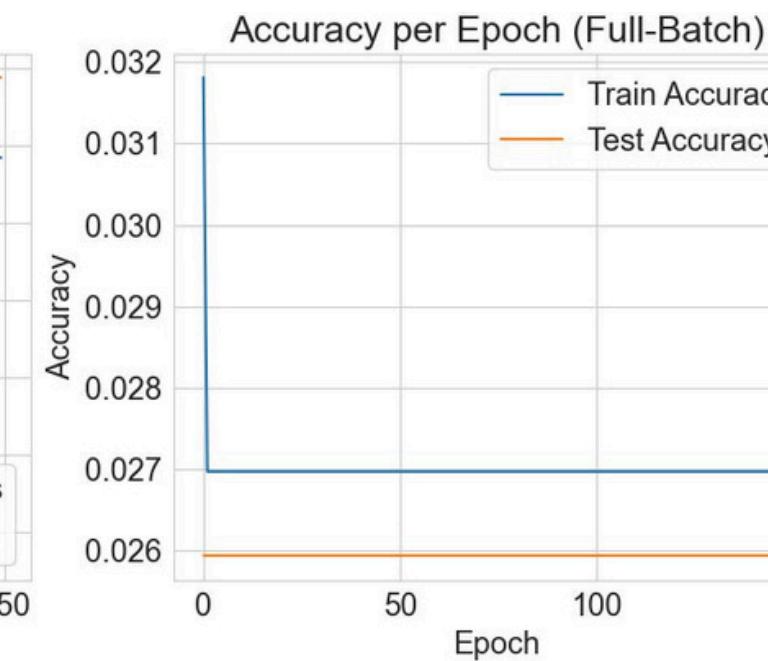
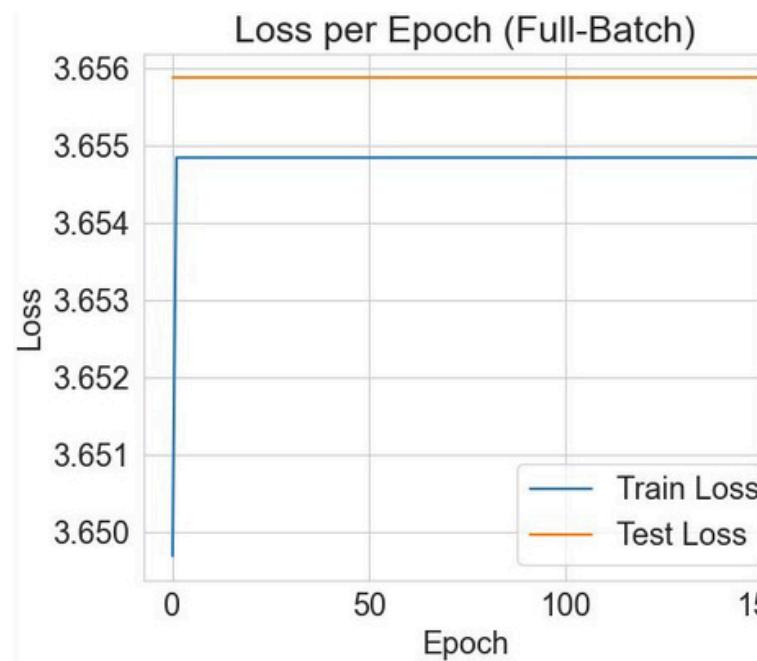
Il T-test si basa sulla ipotesi nulla: i due sistemi di learning hanno la stessa accuratezza. Affinché i due risultati abbiano significato, bisogna dimostrare il rifiuto dell'ipotesi nulla, e cioè che uno dei due sistemi è più accurato dell'altro. Infatti, sotto l'ipotesi nulla, tutte le differenze osservabili sono spiegate da una variazione random. L'obiettivo è quello di determinare la probabilità p che la differenza media tra i due set di osservazioni supporti l'ipotesi nulla. Se p è sufficientemente piccolo (tipicamente < 0.05), allora l'ipotesi nulla è rifiutata.

Confronto	t_stat	p_value
Full Batch vs Mini Batch	-18,9057	1,30779E-41
Full Batch vs SGD	-22,7862	8,00965E-51
Mini Batch vs SGD	-32,7365	2,01E-98

Ha senso usare la PCA?

Analisi dei risultati ottenuti senza l'applicazione della PCA

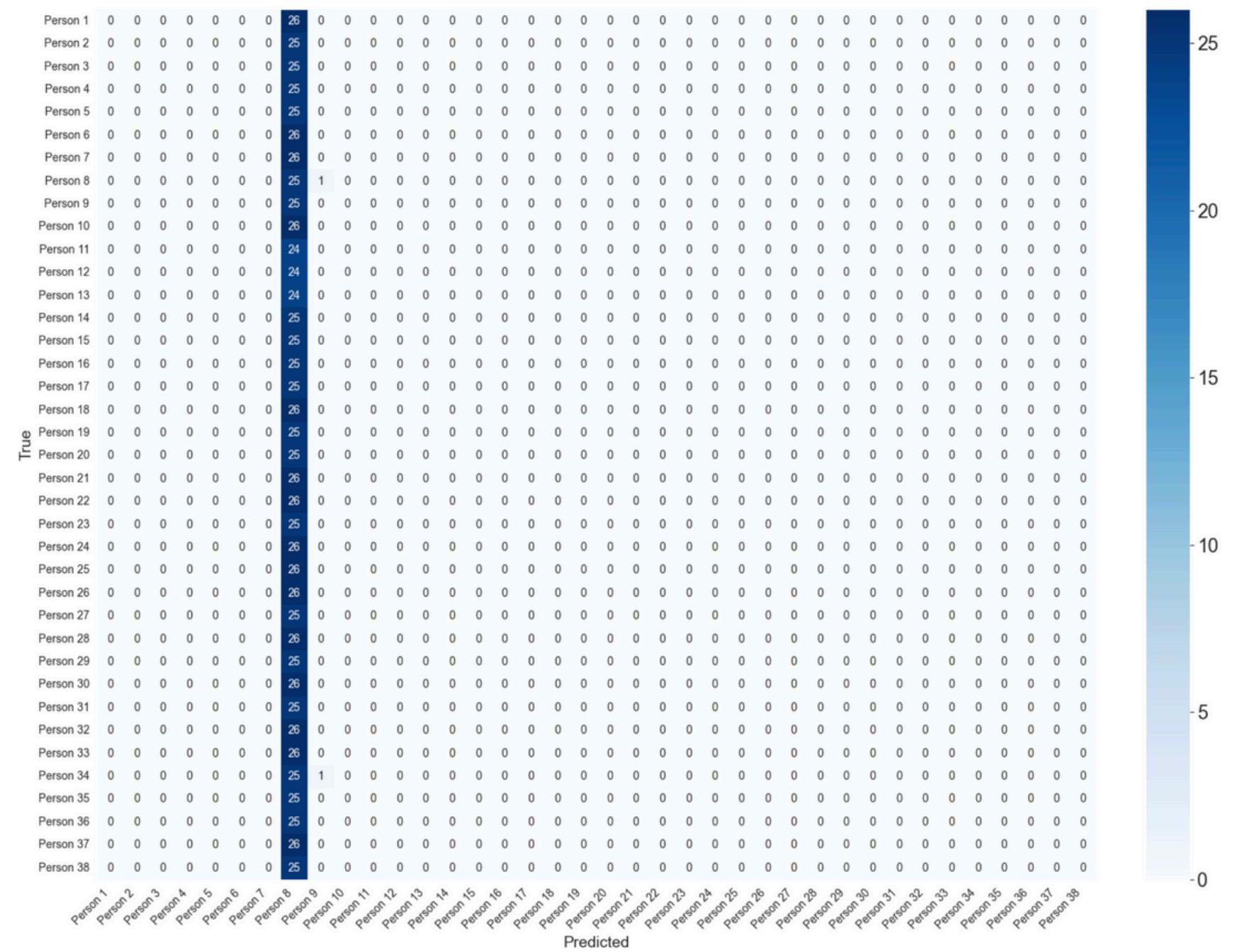
Loss function



Matrice di confusione

La matrice di confusione è la medesima per tutti i 3 approcci del gradient descent. Da questo grafico si può notare chiaramente come questo modello, senza la riduzione della dimensionalità, non sia in grado di separare le classi in modo chiaro. Le immagini sono ad alta dimensionalità e la regressione logistica fatica a trovare decision boundary lineari efficaci.

Infatti qualsiasi immagine venga predetta come la classe “Person 28”, rendendo queste predizioni totalmente inutili.



Tempi di esecuzione

Si può notare come l'utilizzo della PCA con un numero maggiori di componenti comporti un aumento dei tempi di esecuzione. Infatti riducendo il numero di componenti principali selezionate (in questo caso, 70 invece di 200), la parte di calcolo legata alla riduzione dimensionale viene semplificata.

	200 components	70 components
Calcolo PCA	1,308 s	0,821 s

Tempi di esecuzione

Nell'analisi del tempo di esecuzione per ogni epoca bisogna innanzitutto considerare **il numero di operazioni coinvolte**: nel caso del Full Batch, abbiamo una singola operazione vettorizzata che calcola il gradiente sull'intero dataset e un singolo aggiornamento dei parametri del modello, invece l'SGD esegue un'operazione separata per ciascun campione, seguita da un aggiornamento dei parametri per ogni singola iterazione. Anche se l'operazione vettorizzata del Full Batch può essere leggermente più lenta rispetto a un singolo calcolo di SGD, il fatto che l'SGD esegua molte più operazioni in una singola epoca rende inevitabile che risulti più lento a livello di tempo di calcolo per epoca.

Inoltre bisogna anche considerare la velocità con cui i diversi metodi avvicinano i parametri all'obiettivo finale, ovvero il minimo della funzione di perdita, quindi bisogna analizzare questi approcci rispetto alla convergenza e al comportamento dei learning rate.

	Full Batch	Mini Batch	SGD
PCA	1,34 s	2,29 s	27,23 s
NO-PCA	10,29 s	19,50 s	296,46 s

Tempi di esecuzione

Nella trattazione della **convergenza** bisogna evidenziare un vantaggio隐含的在SGD中：当参数更新时，如果模型立即根据看到的数据点进行更新，它可能会更快地接近目标（就像可能暂时偏离正确的方向一样）；因此，在同一 epoch 内的后续更新将受益于这种即时移动。

另一个关键方面是 **learning rate** 的作用：较高的 learning rate 可以通过更长的步幅来补偿较少的更新频率，从而加快收敛速度。然而，过高的学习率可能导致不稳定性，如振荡或直接错过最小值。相比之下，SGD 使用较小的步幅，有助于保持稳定性，但通常需要更多的 epoch 才能达到目标。

	Full Batch	Mini Batch	SGD
PCA	1,32 s	0,22 s	1,24 s
NO-PCA	10,31 s	1,99 s	15,26 s

Tempi di esecuzione

Infine, sebbene l'SGD possa essere vantaggioso in termini di velocità di convergenza in alcune situazioni (ovvero ridurre il numero di epoche necessarie), il costo computazionale più alto per epoca può comunque renderlo più lento in termini di tempo totale di esecuzione rispetto al Full Batch, specialmente se il dataset è di dimensioni tali che un'operazione vettorizzata non diventa proibitiva.

In questo caso specifico, il Full Batch sta funzionando molto bene, dirigendosi direttamente verso l'obiettivo con precisione e senza richiedere troppe epoche. Questo spiega perché, anche considerando lo stesso numero di epoche, il Full Batch risulti complessivamente più veloce in termini di tempo di esecuzione rispetto all'SGD.

	Full Batch	Mini Batch	SGD
PCA	1,32 s	0,22 s	1,24 s
NO-PCA	10,31 s	1,99 s	15,26 s

Conclusioni

Face Recognition using PCA

Conclusioni

L'obiettivo di questo progetto è quello di dimostrare come **l'utilizzo della PCA** sia fondamentale per ottenere delle predizioni soddisfacenti e accurate per questo task. Tuttavia bisogna evidenziare come per questo progetto sia stato utilizzato il modello di classificazione **più semplice in assoluto**, a causa di limitazioni computazionali, infatti i risultati sarebbero sicuramente migliori con l'utilizzo di modelli più complessi come SVM o Neural Network. Inoltre è fondamentale notare come la scelta di uno dei 3 approcci al Gradient Descent comporti delle differenze significative in termini di risultati ottenuti.

Grazie a questo progetto abbiamo avuto la possibilità di mettere in pratica i concetti teorici studiati durante il corso, riuscendo a comprendere quanto sia importante l'applicazione delle tecniche di riduzione della dimensionalità per ottenere dei risultati validi e soddisfacenti.

Grazie per l'attenzione