doc

Valentina Piscopo

Agosto 2025

Sommario

Background: Nel machine learning, l'*effetto Rashomon* denota la presenza — per uno stesso compito e lo stesso split dei dati — di molteplici modelli tra loro diversi (architetture, pesi, regole interne) che ottengono prestazioni comparabili; questa molteplicità implica che anche le spiegazioni possano divergere pur a parità di accuratezza.

Metodi: È stato costruito un Rashomon set di 10 reti neurali convolutional (CNN) con accuratezza entro l'1% sul dataset MNIST. Per un campione di 10 immagini di test, sono state generate spiegazioni locali utilizzando tre metodi: Saliency, Integrated Gradients (IG) e LIME. Sono state quindi valutate la similarità tra le spiegazioni (tramite SSIM, correlazione di Pearson e Spearman, similarità del coseno e MAE) e la loro fedeltà (utilizzando le curve MoRF e la metrica AOPC).

Risultati: La variabilità nelle spiegazioni è risultata essere principalmente guidata dalla scelta del metodo di XAI, non dal modello specifico all'interno del Rashomon set. La similarità inter-modello (stesso metodo, modelli diversi) è sistematicamente più alta della similarità intra-modello (metodi diversi, stesso modello). In termini di fedeltà, LIME ha ottenuto il punteggio AOPC più alto (0.7993), indicando che le feature che identifica sono le più decisive per la previsione del modello. Integrated Gradients ha mostrato un buon compromesso tra fedeltà (AOPC = 0.7511) e coerenza.

Conclusioni: I risultati dimostrano che, in questo contesto, la scelta dell'algoritmo di spiegazione ha un impatto maggiore sulla spiegazione risultante rispetto alla variazione dei parametri del modello. Importante sottolineare l'importanza di valutare sia la similarità che la fedeltà per validare i metodi XAI, specialmente in scenari dove coesistono modelli equivalentemente accurati.

Indice

1 | Introduzione

Quando ci affidiamo ai metodi di spiegazione per i modelli di *machine learning*, sorge una domanda cruciale: *fissato un metodo di spiegazione, quanto sono stabili (sez. 5.2.1) le spiegazioni al variare del modello all'interno del Rashomon set, e come possiamo quantificarne la variabilità?*

Nel contesto del *Rashomon effect* [10, 11], in cui coesistono molteplici modelli con prestazioni equivalenti ma parametri differenti, questa domanda assume un significato ancora più rilevante: se due modelli ottengono lo stesso livello di accuratezza, possiamo aspettarci che spieghino le proprie decisioni nello stesso modo?

Per affrontare questo problema, si adotta un approccio in tre fasi:

1. Costruire un setup sperimentale con più modelli ugualmente accurati, appartenenti a un *Rashomon set*, e analizzarli con diversi metodi di spiegazione.

2. Applicare più metriche di similarità (ad esempio *Structural Similarity Index* — SSIM, correlazione di Pearson, similarità coseno, ecc.) alle spiegazioni prodotte.

3. Confrontare i risultati per capire come varia la spiegazione in funzione sia del modello che del metodo scelto.

Tuttavia, questo approccio presenta alcune criticità:

- Due metodi di spiegazione, applicati allo stesso input, possono evidenziare regioni o feature completamente diverse come rilevanti.

- Metriche diverse possono portare a conclusioni contrastanti sul grado di similarità.

- L'assenza di un *ground truth* della "spiegazione corretta" rende impossibile dichiarare in assoluto quale metodo sia "migliore".

Per questo motivo, la sola similarità non è sufficiente a giudicare la qualità di una spiegazione. È necessario integrarla con un'analisi della *fedeltà*, cioè della capacità della spiegazione di individuare feature realmente decisive per la predizione del modello.

Un approccio comunemente adottato è l'analisi *Most Relevant First* (MoRF) [16], in cui si mascherano progressivamente le feature più importanti secondo la spiegazione e si osserva la velocità con cui decresce la confidenza del modello. Ad esempio, in un classificatore di immagini, se rimuovere la regione indicata come

più rilevante provoca un crollo immediato della probabilità predetta per la classe corretta, la spiegazione è considerata fedele; al contrario, un impatto minimo sulla predizione indicherebbe una spiegazione poco utile.

In letteratura non esiste una metrica unica e definitiva per la valutazione delle spiegazioni [1], ma piuttosto un insieme di prospettive complementari. Viene adottata quindi una doppia prospettiva:

- analisi della **similarità** tra spiegazioni (intra-modello e inter-modello);
- analisi della **fedeltà** tramite MoRF e *Area Over the Perturbation Curve* (AOPC).

2 | Introduzione all'implementazione sperimentale

L'esperimento è stato strutturato come una pipeline in più fasi, ciascuna con un obiettivo specifico, seguendo approcci già consolidati in letteratura [1, 10, 11]:

1. **Costruzione di un Rashomon set**: una raccolta di modelli differenti, ma tutti con prestazioni simili sullo stesso compito di classificazione, in linea con la definizione proposta da Fisher et al. [5].

2. **Applicazione di metodi di spiegazione**: generazione di spiegazioni locali per ciascun modello, su un insieme di dati di test, utilizzando diversi algoritmi XAI, tra cui Saliency [19], Integrated Gradients [22] e LIME [15].

3. **Valutazione della similarità delle spiegazioni**: confronto quantitativo tra le spiegazioni prodotte da modelli e metodi differenti, tramite varie metriche di similarità già impiegate in studi precedenti [2, 16].

4. **Valutazione della fedeltà delle spiegazioni**: misurazione di quanto le feature individuate dalle spiegazioni siano realmente determinanti per le predizioni dei modelli, mediante tecniche come le curve *MoRF* [16].

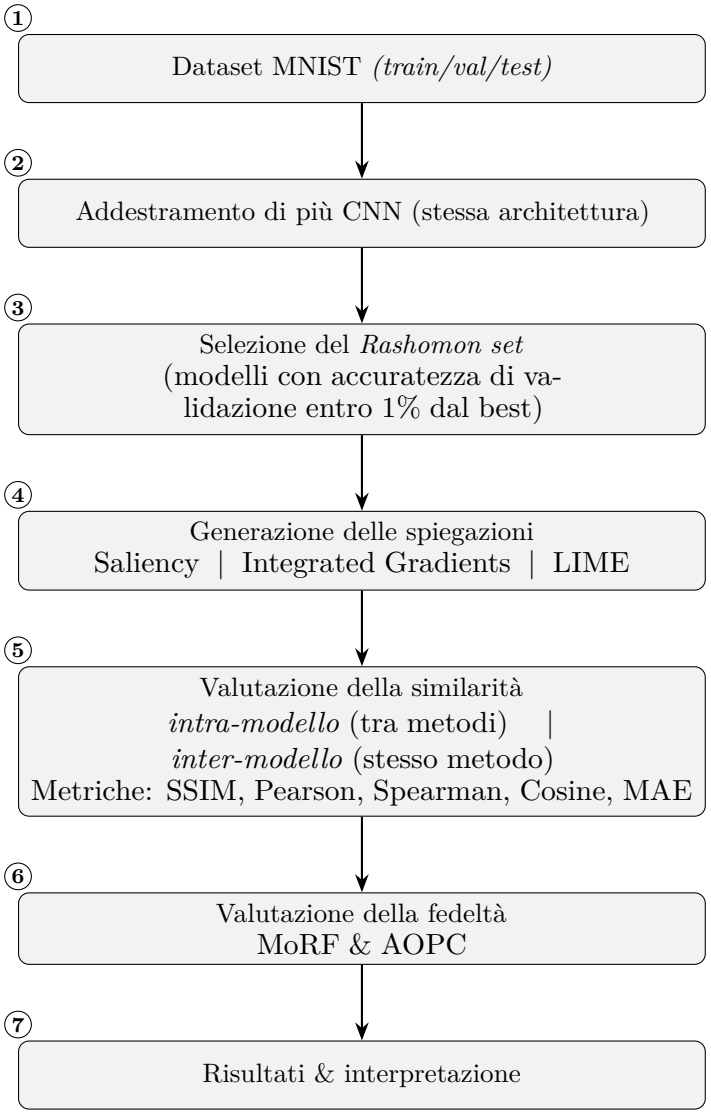


Figura 2.1: Pipeline sperimentale ad alto livello: costruzione del *Rashomon set*, generazione delle spiegazioni e valutazione di similarità e fedeltà.

2.1 Tecnologie

Per implementare il workflow sperimentale descritto, è stato utilizzato un ecosistema di strumenti largamente adottati in ambito *explainable AI*, in grado di garantire affidabilità, riproducibilità e scalabilità [1, 11]:

- **Python 3.x**: linguaggio di riferimento per la ricerca in XAI, scelto per la sua flessibilità e l'ampia disponibilità di librerie specializzate.
- **PyTorch**: libreria *open-source* per il *machine learning* e *deep learning*, dotata di supporto nativo per l'autograd e adatta alla prototipazione rapida di modelli. Utilizzata per la definizione, l'addestramento e la validazione delle reti neurali,

nonché per il calcolo dei gradienti richiesto dai metodi *Saliency* e *Integrated Gradients*.

- **Torchvision**: libreria complementare a PyTorch che offre dataset predefiniti (come MNIST), trasformazioni standard per immagini e modelli già pronti. Utilizzata per il download, la gestione e il preprocessing del dataset MNIST.
- **Captum** [8]: libreria open source specifica per l'interpretabilità di modelli PyTorch. Fornisce implementazioni ottimizzate di numerosi metodi XAI, con API coerenti e facilmente integrabili. Utilizzata per generare le spiegazioni su tutti i modelli del Rashomon set, includendo metodi *gradient-based* e *model-agnostic* (sezione 4.5).
- **NumPy**: libreria fondamentale per il calcolo scientifico e la manipolazione efficiente di array numerici, usata per l'elaborazione dei dati, la normalizzazione delle spiegazioni e il calcolo di metriche.
- **Scikit-learn**: utilizzata per il calcolo di metriche (correlazioni, MAE) e per funzioni di utilità scientifica.
- **SciPy**: impiegata per calcolare correlazioni avanzate come Spearman e Pearson.
- **Scikit-image**: usata per il calcolo della metrica *SSIM* (*Structural Similarity Index*).
- **Matplotlib**: libreria di riferimento per la visualizzazione scientifica, utilizzata per produrre grafici e visualizzazioni qualitative delle spiegazioni.
- **Tqdm** [23]: per fornire barre di avanzamento e monitorare l'esecuzione di processi lunghi.
- **Glob, os**: per la gestione di file, directory e per il caricamento/salvataggio dei modelli, a supporto della riproducibilità.

2.2 Riproducibilità e ambiente di sviluppo

Per garantire la riproducibilità degli esperimenti e la gestione efficiente delle dipendenze, sono state adottate le seguenti buone pratiche, in linea con le raccomandazioni di Pineau et al. [12]:

- **Gestione dell'ambiente con *conda***: tutte le dipendenze (librerie e relative versioni) sono state installate in un ambiente dedicato, così da poter replicare esattamente il contesto di esecuzione.

- **Controllo della casualità**: i *random seed* sono stati fissati per NumPy, PyTorch e il generatore casuale di Python, così da rendere i risultati ripetibili anche in presenza di componenti stocastiche come l'inizializzazione dei pesi o lo *shuffle* dei dati.
- **Salvataggio e caricamento dei modelli**: i modelli selezionati per il Rashomon set sono stati salvati su disco durante la fase di addestramento, evitando la ripetizione di processi costosi e permettendo il loro riutilizzo in fasi successive.

2.3 Scelte implementative

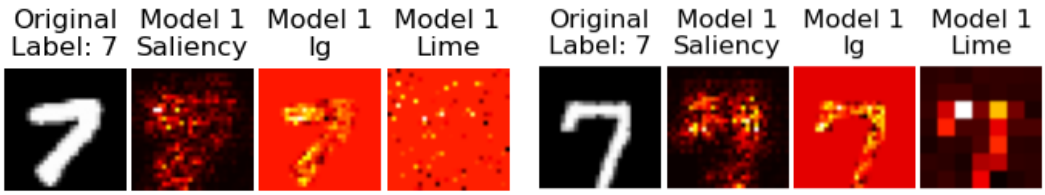
Gli esperimenti sono stati condotti interamente su **CPU**, scelta che ha avuto un impatto diretto sulle decisioni implementative. Lavorare senza GPU ha imposto di trovare un equilibrio tra fedeltà delle spiegazioni e tempi di calcolo ragionevoli, evitando configurazioni eccessivamente costose in termini computazionali [10].

LIME. Invece di operare a livello di singolo pixel (784 feature su MNIST), è stato adottato un approccio *patch-based*, suddividendo ogni immagine 28×28 in blocchi 4×4 (`feature_mask`), per un totale di 49 feature. Questa scelta è coerente con le raccomandazioni di Ribeiro et al. [15] per ridurre il rumore e migliorare la stabilità:

1. Riduce il rumore nelle spiegazioni: LIME soffre particolarmente quando le feature sono estremamente piccole, perché le perturbazioni casuali tendono a distruggere informazioni rilevanti.
2. Riduce drasticamente il numero di perturbazioni necessarie per ottenere una regressione lineare stabile.

Il numero di campioni *n_samples* è stato fissato a 200, per mitigare la variabilità intrinseca di LIME e migliorare la ripetibilità dei risultati. Poiché ogni spiegazione richiede *n_samples* forward pass, è stato utilizzato `perturbations_per_eval=50` per elaborare perturbazioni in batch, riducendo così l'overhead di chiamate ripetute al modello.

Integrated Gradients (IG). È stato mantenuto il valore predefinito di `n_steps` fornito da Captum. Questo riduce il numero di forward pass necessari rispetto a valori più alti (che migliorano la precisione dell'integrazione ma aumentano i tempi di calcolo). Come baseline è stata utilizzata un'immagine completamente nera (tutti zeri), scelta comune in letteratura per dataset con sfondo uniforme come MNIST [22].



(a) LIME pixel-wise (784 feature) (b) LIME a patch 4×4 (49 feature)

Figura 2.2: Confronto qualitativo delle mappe LIME prima/dopo il passaggio al masking a patch. L'approccio a patch riduce il rumore e mette a fuoco regioni più coerenti con la cifra. Immagini simili, stesso modello, $n_{\text{samples}} = 200$.

Saliency. Calcolata con il metodo base di Captum senza smoothing o normalizzazioni aggiuntive, per preservare la semplicità e l'interpretabilità diretta delle gradient map. Tecniche come SmoothGrad [21] avrebbero aumentato la stabilità visiva, ma anche i tempi di calcolo di ordini di grandezza.

MoRF/AOPC. La valutazione di fedeltà è stata implementata rimuovendo progressivamente le feature più importanti in 10 step. La baseline di rimpiazzo è stata scelta come *media dell'immagine* anziché zeri, per minimizzare il rischio di introdurre pattern artificiali fuori distribuzione che avrebbero potuto alterare il comportamento del modello [16].

Selezione del Rashomon set. Per ogni inizializzazione (seed) il modello viene addestrato sul **training set** e monitorato sul **validation set** per l'*early stopping*: si salva il *checkpoint* con la miglior prestazione in validazione (miglior *validation accuracy*, con pazienza `patience`=3) e l'addestramento si interrompe quando non si osservano ulteriori miglioramenti. Concluso il training, i modelli sono selezionati in base alla **validation accuracy**: tutti quelli con accuratezza $\geq$ (*best* -1%) entrano nel *Rashomon set* (11). Solo dopo aver concluso il set, i checkpoint selezionati vengono valutati sul **test set** per ottenere una stima imparziale della generalizzazione; il test non è usato per scegliere modelli o iperparametri (evitando data leakage). Tutti i pesi selezionati sono salvati in formato `.pt` di PyTorch, con semi fissati, per garantire la riproducibilità delle sperimentazioni.

Classe di riferimento per le attribuzioni. Le attribuzioni sono calcolate rispetto alla **classe vera** (*true label*) e non a quella predetta. Questa scelta, già discussa in studi precedenti [3], mira a mantenere coerenza con il *ground truth*, evitando che errori di classificazione contaminino il confronto tra spiegazioni. Tuttavia, si riconosce che questa impostazione non riflette scenari reali, dove la classe vera potrebbe non

essere disponibile — rappresentando quindi una possibile minaccia alla validità esterna.

2.4 Hyperparametri e setup sperimentale

Nella Tabella 2.1 sono riportati tutti i parametri e le impostazioni utilizzate negli esperimenti, in modo da permettere la riproducibilità dei risultati.

Parametro	**Valore**
Numero di modelli addestrati	10
Epoche massime	30
Patience (early stopping)	3 epoche
Soglia Rashomon	1% di differenza rispetto al miglior modello (val. acc.)
Batch size	64
Baseline IG	immagine di zeri
Baseline MoRF	media dell'immagine
n_samples LIME	200
Feature mask LIME	blocchi 4×4 (49 feature totali)
n_steps IG	valore predefinito Captum
Step MoRF	10
Classe di riferimento	Classe vera (*true label*)
Ottimizzatore	Adam
Tasso di apprendimento iniziale	0.001
Dataset	MNIST (grayscale, 28×28)

Tabella 2.1: Hyperparametri e setup utilizzati negli esperimenti.

3 | Il Rashomon set

Il *Rashomon set* è l'insieme dei modelli che, per uno stesso compito e lo stesso split dei dati, ottengono prestazioni quasi ottimali (entro una soglia ε dal migliore). La coesistenza di molteplici soluzioni predittivamente equivalenti è nota come *effetto Rashomon* [5, 11]. In ambito XAI, questa molteplicità implica che differenze nelle spiegazioni possano riflettere *variazioni tra modelli* anche a parità di accuratezza, non necessariamente errori o rumore [18]. Per isolare l'effetto dovuto ai modelli è quindi cruciale **fissare il metodo di spiegazione** e confrontare le attribuzioni *tra modelli* (analisi inter–modello). Al contrario, il confronto *tra metodi* sullo stesso modello quantifica la variabilità indotta dallo strumento XAI e non l'effetto Rashomon in senso stretto. Sono stati adottati entrambi i piani: (i) inter–modello a metodo fissato, per misurare la stabilità entro il Rashomon set; (ii) intra–modello, per confrontare i metodi a parità di modello.

3.1 Costruzione

Il Rashomon set è stato costruito allenando più reti neurali della stessa architettura (una CNN semplice) sul dataset MNIST. Ogni rete parte da una diversa inizializzazione casuale dei pesi (*random seed* differente) ed è addestrata sugli stessi dati con un protocollo identico, che include la suddivisione in training, validation e test set.

Durante l'addestramento è stata applicata la tecnica di *early stopping* [13]: per ogni modello, l'addestramento viene interrotto se le prestazioni sul validation set non migliorano per un numero prestabilito di epoche consecutive. In questo modo si seleziona automaticamente la versione del modello che ha ottenuto la miglior accuratezza sul validation, riducendo il rischio di overfitting e garantendo un confronto equo tra i membri del set.

3.2 Selezione

Per ciascun seed il modello è addestrato sul **training set** e monitorato sul **validation set** con *early stopping*; si conserva il *checkpoint* con la miglior prestazione in validazione (best in validation). La costruzione del *Rashomon set* avviene sulla base della **validation accuracy** del checkpoint migliore: si includono tutti i modelli

entro l'1% dal migliore, ovvero

$$\text{Accuracy}_{\text{val}}(m) \;\geq\; \text{Accuracy}_{\text{val}}(m_{\text{best}}) - \epsilon$$

dove:

- $\text{Accuracy}_{\text{val}}(m_{\text{best}})$ è la massima accuratezza in validazione tra i modelli addestrati (valutata sul loro *best checkpoint*);
- $\epsilon = 0.01$ è la tolleranza (1%), in linea con la definizione operativa di *Rashomon set* [5].

Solo **dopo** aver chiuso il set secondo questa regola, i checkpoint selezionati sono valutati sul **test set** per ottenere una stima imparziale della generalizzazione: il test non è mai usato per la selezione, prevenendo *data leakage* [10, 11].

Listing 3.1: Training con early stopping, selezione su validation ed evaluation su test

```
best_val_acc = -inf
val_acc_per_model = []
checkpoints = []

for i in 1..NUM_MODELS:
    set_seed(SEED + i)

    # addestra su TRAIN, monitora VALIDATION con early stopping
    model = init_model()
    hist = train_with_early_stopping(
             model, train_data, val_data,
             monitor="validation", # loss o accuracy
             patience=PATIENCE)

    load_best_checkpoint(model, hist)

    acc_val = evaluate_accuracy(model, val_data)
    val_acc_per_model.append(acc_val)
    checkpoints.append(save_checkpoint(model))

    if acc_val > best_val_acc:
        best_val_acc = acc_val

```

```
# selezione Rashomon set su VALIDATION
RASHOMON = []
for i, acc_val in enumerate(val_acc_per_model):
    if acc_val >= best_val_acc - EPSILON: # EPSILON = 0.01
        RASHOMON.append(checkpoints[i])

# valutazione finale su TEST
test_scores = []
for ckpt in RASHOMON:
    model = load_checkpoint(ckpt)
    acc_test = evaluate_accuracy(model, test_data)
    test_scores.append(acc_test)
```

3.3 Validazione dell'equivalenza (accuracies)

Per verificare l'equivalenza operativa dei modelli selezionati (soglia 1% sulla validation), riportiamo le accuratezze di validazione e test per ciascun modello e le statistiche riassuntive sull'insieme.

Model	ValAcc	TestAcc
9	99.10%	99.08%
1	99.03%	99.16%
8	99.03%	99.02%
3	98.97%	98.91%
4	98.95%	99.04%
2	98.98%	98.99%
7	98.89%	98.86%
5	98.88%	98.96%
0	98.83%	99.01%
6	98.59%	98.77%

Tabella 3.1: Accuratezze per modello del Rashomon set.

	Media ± std	**Min..Max**
ValAcc	98.87% ± 0.17%	98.59% .. 99.10%
TestAcc	98.96% ± 0.11%	98.77% .. 99.16%

Tabella 3.2: Statistiche riassuntive sulle accuratezze del Rashomon set ($n = 10$ modelli).

Le distribuzioni sono strette (dev. std < 0.2 p.p. su entrambi i set) e i range rientrano nella soglia di selezione all'1%, confermando l'*equivalenza* dei modelli ai fini delle analisi di similarità e fedeltà.

3.4 Approfondimento

3.4.1 Equivalenza tra modelli

Nel contesto di questo lavoro, due modelli sono considerati equivalenti se soddisfano la condizione di soglia sull'accuratezza definita precedentemente. Questa scelta è in linea con la definizione originaria di *Rashomon set* [5], secondo cui l'insieme comprende modelli con prestazioni quasi ottimali rispetto al miglior modello ottenuto. L'adozione di tale criterio permette di:

- Rispettare la formulazione teorica riportata in letteratura, in cui il Rashomon set è visto come una regione nello spazio dei modelli che contiene tutte le soluzioni con accuratezza comparabile [11].
- Esplorare la diversità interna tra modelli che, in termini predittivi, sembrano "uguali", ma che possono differire nelle rappresentazioni interne e quindi nelle spiegazioni [10].

3.4.2 Perché utilizzare l'early stopping

L'*early stopping* [13] è stato adottato per due motivi principali:

- **Evitare l'overfitting**: interrompendo l'addestramento quando le prestazioni di validazione smettono di migliorare, si evita che il modello memorizzi il training set perdendo capacità di generalizzazione.
- **Equità nel confronto**: applicando la stessa regola a tutti i modelli, ciascun membro del Rashomon set viene selezionato nelle condizioni di miglior equilibrio tra bias e varianza, come raccomandato nelle buone pratiche di confronto tra modelli.

Questo approccio assicura che la variabilità osservata nelle spiegazioni non sia dovuta a un diverso grado di overfitting, ma a reali differenze nei percorsi di apprendimento. Inoltre, fissando i *random seed* per tutte le componenti stocastiche (inizializzazione dei pesi, shuffle dei dati, generatori casuali di librerie esterne) [14], è possibile distinguere la variabilità dovuta al caso da quella legata a differenze strutturali nei modelli o nelle spiegazioni.

3.4.3 Scelta dell'ottimizzatore: Adam.

Per l'addestramento di tutte le reti neurali è stato utilizzato l'ottimizzatore *Adam* (*Adaptive Moment Estimation*) [7], ampiamente adottato in ambito *deep learning*. Adam adatta dinamicamente il tasso di apprendimento per ciascun parametro in base alle stime dei momenti del gradiente, permettendo:

- Convergenza più rapida rispetto a metodi a tasso di apprendimento fisso.
- Maggiore stabilità anche in presenza di gradienti rumorosi, come avviene nei dati di immagini con variazioni locali.
- Ridotta necessità di un fine-tuning manuale del *learning rate*.

L'uso di Adam ha reso possibile mantenere tempi di addestramento contenuti pur garantendo una buona qualità della convergenza, aspetto particolarmente rilevante data l'esecuzione esclusivamente su CPU.

4 | Metodi di spiegazione adottati

Una volta selezionato il *Rashomon set* dei modelli, il passo successivo consiste nell'analizzare come ciascun modello giunge alle proprie decisioni. Per questo scopo sono stati utilizzati tre metodi di spiegazione, scelti per rappresentare sia approcci *gradient-based* che *model-agnostic* [1, 6].

4.1 Saliency

Il metodo della *saliency map* [16, 19] è uno dei più semplici e diffusi. Calcola la derivata della probabilità (o della logit) assegnata dal modello alla classe target rispetto a ciascun pixel dell'immagine di input. I pixel associati ai valori assoluti più elevati sono considerati più importanti per la decisione. Questo metodo è apprezzato per la sua immediatezza, ma è noto per essere sensibile al rumore e all'inizializzazione dei pesi del modello.

4.2 Integrated Gradients (IG)

Gli *Integrated Gradients* [22] migliorano l'approccio delle saliency map, correggendone alcune limitazioni. Calcolano il contributo di ogni feature effettuando una media dei gradienti lungo un percorso che va da una *baseline* (nel nostro caso, immagine nulla) fino all'immagine reale. Questo consente di ottenere spiegazioni più stabili e coerenti, meno influenzate da piccole variazioni nei dati o nei parametri del modello.

4.3 LIME

Il metodo *LIME* (*Local Interpretable Model-agnostic Explanations*) [15] genera nuove istanze di input perturbate (ad esempio, oscurando casualmente parti dell'immagine) e osserva come cambia la predizione del modello. Successivamente, addestra un modello interpretabile locale (ad esempio una regressione lineare) per stimare quali feature hanno avuto il maggiore impatto sulla decisione del modello. Nel caso di MNIST, le perturbazioni sono state effettuate non a livello di singolo pixel ma aggregando in blocchi 4×4 per ridurre il rumore e i tempi di calcolo (sezione 2.3).

Listing 4.1: Generazione delle spiegazioni

```
function genera_spiegazioni(modello, immagine, etichetta_vera):
    spiegazioni = {}
```

```
3
4       spiegazioni["Saliency"] = calcola_saliency(modello, immagine,
            etichetta_vera)
5       spiegazioni["IG"] = calcola_integrated_gradients(
6           modello, immagine, etichetta_vera, baseline_zeri)
7       spiegazioni["LIME"] = calcola_lime(
8           modello, immagine, etichetta_vera,
9           campioni=200, maschera_feature=FEATURE_MASK,
10          baseline_media_pixel)
11
12      return spiegazioni
```

4.4 Motivazioni della scelta

La combinazione di questi tre metodi consente di confrontare:

- Approcci *gradient-based* (Saliency, IG) e approcci *model-agnostic* (LIME) [15, 20, 22].
- Metodi semplici e veloci con tecniche più robuste e computazionalmente costose.
- Stabilità delle spiegazioni e capacità di cogliere diversi aspetti dell'importanza delle feature [6, 16].

4.5 Gradient-based vs Model-agnostic

I metodi di spiegazione *gradient-based* sfruttano direttamente la struttura interna del modello: calcolano come varia la predizione rispetto a piccole modifiche delle feature in input, utilizzando le derivate calcolate tramite *backpropagation* [20, 22]. Questi metodi sono veloci e, per modelli differenziabili come le reti neurali, forniscono indicazioni precise sulle feature che guidano la decisione.

I metodi *model-agnostic*, invece, trattano il modello come una "scatola nera": non richiedono accesso ai pesi o ai gradienti, ma solo la possibilità di effettuare predizioni su input modificati [6, 15]. Questo li rende molto flessibili (applicabili a qualsiasi modello), ma spesso più lenti e meno stabili, poiché si basano su approssimazioni locali.

5 | Valutare la similarità tra spiegazioni

Quando osserviamo due spiegazioni, possiamo essere tentati di giudicare istintivamente se siano simili o meno. Ma le apparenze ingannano: differenze visive possono non riflettere reali differenze nei pattern di importanza, e viceversa. Nel contesto di un *Rashomon set*, questa domanda diventa cruciale: modelli diversi, ma ugualmente accurati, arrivano alle stesse conclusioni per motivi simili, o per motivi profondamente diversi? Misurare la similarità tra spiegazioni è un passo fondamentale per capire la robustezza e la stabilità delle interpretazioni fornite dai metodi di *eXplainable AI* (XAI) [11, 16].

5.1 Metriche adottate

Per trasformare un concetto qualitativo come "somiglianza visiva" in numeri, è necessario scegliere metriche che catturino aspetti diversi della relazione tra due mappe di importanza:

- **Structural Similarity Index (SSIM)** — valuta quanto due mappe siano simili in termini di struttura, considerando luminanza, contrasto e distribuzione spaziale. Un SSIM vicino a 1 indica che le due spiegazioni hanno pattern strutturali quasi identici [24].
- **Pearson correlation** — misura la correlazione lineare tra i valori di importanza, utile per capire se i valori crescono e decrescono insieme, indipendentemente dall'ordine dei pixel.
- **Spearman correlation** — analizza la correlazione tra i ranghi, quindi l'ordine relativo delle feature più importanti, anche se le scale numeriche sono diverse.
- **Cosine similarity** — confronta la direzione dei vettori di importanza, ignorando la loro lunghezza: due spiegazioni che mettono in evidenza le stesse zone avranno un valore vicino a 1, anche se una è "più intensa" dell'altra.
- **Mean Absolute Error (MAE)** — fornisce una misura diretta della differenza media assoluta nei valori di importanza; più è basso, più le mappe sono simili in valore assoluto.

La scelta di queste metriche consente di catturare diverse sfaccettature della similarità: dalla struttura globale all'ordine delle feature, fino alla corrispondenza numerica esatta [10, 16].

Range e interpretazione delle metriche. Per ogni metrica vengono specificati: il **range teorico**, il **range nel setup** (dopo min–max normalizzazione delle mappe in $[0, 1]$), la **direzione di miglioramento** (↑ (alto) meglio / ↓ (basso) meglio) e il significato pratico del valore.

Metrica	Range teorico	Range setup	Migliore	Interpretazione
SSIM	$[-1, 1]$	$[0, 1]$	↑	Somiglianza strutturale (luminanza/contrasto/-struttura). Con mappe in $[0, 1]$ e `data_range=1.0`, valori $\to 1$ indicano pattern quasi identici; ≈ 0 bassa somiglianza.
Pearson (r)	$[-1, 1]$	$[-1, 1]$	↑	Correlazione lineare: 1 = forte accordo, 0 = nessuna dipendenza, < 0 = inversione.
Spearman (ρ)	$[-1, 1]$	$[-1, 1]$	↑	Correlazione di rango: 1 stesso ordinamento, 0 indipendenza, < 0 ordinamenti opposti.
Cosine	$[-1, 1]$	$[0, 1]$	↑	Allineamento direzionale dei vettori: $\to 1$ molto allineati, ≈ 0 ortogonali. Con mappe non negative i valori sono ≥ 0.
MAE	$[0, \infty]$	$[0, 1]$	↓	Differenza media assoluta punto-a-punto: 0 = mappe identiche.

Tabella 5.1: Range, direzione e interpretazione delle metriche di similarità.

Nota: in tutto il lavoro le mappe di attribuzione sono normalizzate in $[0, 1]$ prima del calcolo delle metriche; per SSIM viene utilizzato `data_range=1.0`. Questo rende i valori tra metodi e modelli comparabili e "lega" MAE all'intervallo $[0, 1]$.

5.2 Procedura di confronto

Il confronto è stato condotto calcolando le metriche per ogni immagine del test set e per ogni coppia di spiegazioni, considerando due scenari distinti [4, 9–11]:

1. **Intra-modello** — confronto tra metodi diversi applicati allo stesso modello. In questo scenario si misura la coerenza tra approcci di spiegazione differenti: se producono mappe simili, significa che il modello è interpretato in modo coerente indipendentemente dal metodo XAI utilizzato.

2. **Inter-modello** — confronto dello stesso metodo applicato a modelli diversi appartenenti al *Rashomon set*. Qui si valuta la stabilità del metodo rispetto a variazioni nella struttura interna e nei parametri del modello, mantenendo invariato l'approccio di spiegazione.

Le attribuzioni sono calcolate sempre rispetto alla **classe vera** (*true label*)[1]. Questa scelta è stata adottata per garantire coerenza con il *ground truth* ed evitare che eventuali errori di classificazione alterino il confronto. Tuttavia, essa rappresenta una potenziale minaccia alla validità esterna: in scenari reali, la classe vera potrebbe non essere disponibile e il comportamento del metodo rispetto alla classe predetta potrebbe differire in modo significativo.

5.2.1 Definizione operativa di stabilità

Viene chiamata **stabilità** di un metodo di spiegazione la capacità di produrre spiegazioni simili al variare di fattori che non dovrebbero alterarne il contenuto informativo. In questo lavoro la nozione principale è la *stabilità inter-modello*: fissato un metodo a e una famiglia di modelli $\mathcal{M}$ con prestazioni equivalenti, le spiegazioni $E_a(x; m)$ restano coerenti al variare di $m \in \mathcal{M}$.

Operativamente, per ogni immagine x e per ogni coppia di modelli (m_i, m_j) del Rashomon set, misuriamo una similarità $\mathrm{sim}(\cdot, \cdot)$ (SSIM, Pearson, Spearman, Cosine, MAE) tra le mappe di attribuzione, e definiamo un indice di stabilità come media (e dispersione) delle similarità:

$$S_{\text{inter}}(a) = \mathbb{E}_x \, \mathbb{E}_{i<j} \Big[\, \mathrm{sim}\Big(E_a(x; m_i), E_a(x; m_j) \Big) \, \Big].$$

[1] La *true label* è l'etichetta corretta associata a un singolo campione (es. la cifra "5" in un'immagine MNIST). Più in generale, il termine *ground truth* indica il riferimento di verità utilizzato per valutare un modello, che nei problemi di classificazione coincide con le true labels.

Valori medi alti e deviazioni standard basse indicano alta stabilità. Notiamo che *stabilità* non implica *fedeltà*: un metodo può essere molto stabile ma poco fedele, e viceversa (sez. 6.1).

5.3 Risultati quantitativi

Le tabelle seguenti riassumono i valori medi delle metriche nei due scenari.

Coppia	**SSIM**	**Pearson**	**Spearman**	**Cosine**	**MAE**	**n**
Saliency–IG	0.263 ± 0.064	0.529 ± 0.082	0.194 ± 0.052	0.651 ± 0.045	0.211 ± 0.059	100
Saliency–LIME	0.178 ± 0.043	0.468 ± 0.063	0.392 ± 0.089	0.663 ± 0.037	0.177 ± 0.045	100
IG–LIME	0.247 ± 0.054	0.421 ± 0.059	0.224 ± 0.059	0.780 ± 0.056	0.167 ± 0.046	100

Tabella 5.2: Similarità *intra-modello*: confronto tra metodi sullo stesso modello (media ± dev. std).

Metodo	**SSIM**	**Pearson**	**Spearman**	**Cosine**	**MAE**	**n**
Saliency	0.439 ± 0.067	0.612 ± 0.064	0.688 ± 0.062	0.727 ± 0.042	0.069 ± 0.012	450
IG	0.711 ± 0.077	0.723 ± 0.079	0.516 ± 0.085	0.970 ± 0.013	0.083 ± 0.049	450
LIME	0.639 ± 0.086	0.864 ± 0.051	0.541 ± 0.116	0.929 ± 0.028	0.093 ± 0.025	450

Tabella 5.3: Similarità *inter-modello*: stesso metodo su modelli diversi (media ± dev. std).

Rimandiamo alla Sez. 7.1 per il trade-off tra stabilità e fedeltà.

Figura 5.1: Esempi di mappe di attribuzione per 10 modelli del Rashomon set su alcune immagini di test (MNIST). Ogni riga corrisponde a una diversa immagine originale, seguita dalle spiegazioni generate con Saliency, Integrated Gradients (IG) e LIME. Si osserva come le spiegazioni siano generalmente più coerenti **tra modelli** usando lo stesso metodo (colonne verticali), mentre differenze più marcate emergono **tra metodi** sullo stesso modello (blocchi orizzontali).

5.3.1 Interpretazione

I risultati mostrano una tendenza chiara e robusta:

- La similarità **inter-modello** è sistematicamente più alta della **intra-modello** per tutte le metriche considerate. Questo indica che un singolo metodo tende a produrre spiegazioni coerenti anche al variare del modello del Rashomon set (es., SSIM: IG 0.711 ± 0.077, LIME 0.639 ± 0.086, Saliency 0.439 ± 0.067, $n = 450$).

- La similarità **intra-modello** (metodi diversi sullo stesso modello) è sensibilmente più bassa, confermando che la scelta del metodo di spiegazione introduce le maggiori divergenze (es., SSIM: Saliency–IG 0.263 ± 0.064, Saliency–LIME 0.178 ± 0.043, IG–LIME 0.247 ± 0.054, $n = 100$).

Guardando nel dettaglio, emerge un quadro coerente con la letteratura. In **inter-modello**, *Integrated Gradients (IG)* risulta il più stabile in termini di direzione del vettore di attribuzioni (Cosine 0.970 ± 0.013), mentre *LIME* mostra la correlazione lineare media più alta (Pearson 0.864 ± 0.051). *Saliency* mantiene un buon ordinamento relativo delle importanze al variare del modello (Spearman 0.688 ± 0.062).

Le deviazioni standard relativamente contenute in questo scenario confermano la coerenza tra modelli dello stesso metodo.

In **intra-modello**, le similarità calano marcatamente: ad esempio, *SSIM* scende a 0.178–0.263 con dispersioni non trascurabili ($\pm$0.043–$\pm$0.064), e le correlazioni di rango (*Spearman*) raramente superano 0.392 (Saliency–LIME 0.392 ± 0.089). Ciò suggerisce che metodi diversi "raccontano storie diverse" anche quando osservano lo stesso modello: differiscono nella *forma* (SSIM), nell'*ordinamento* (Spearman) e, in parte, nell''intensità/direzione (MAE/Cosine) delle regioni ritenute rilevanti.

In sintesi, nel contesto analizzato, la **scelta del metodo di spiegazione** ha un impatto più marcato sulla forma e sul contenuto della spiegazione rispetto alla **scelta del modello**. Le deviazioni standard riportate rendono esplicita questa variabilità: molto contenuta *tra modelli* per uno stesso metodo (es., Cosine di IG 0.970 ± 0.013), più ampia *tra metodi* sullo stesso modello (es., Spearman 0.194 ± 0.052 fino a 0.392 ± 0.089).

6 | Valutare la fedeltà delle spiegazioni: MoRF e AOPC

Misurare la similarità tra spiegazioni è utile per capire se due metodi "raccontano la stessa storia". Ma una spiegazione può anche essere coerente e stabile, eppure irrilevante per il modello. Per tale motivo serve valutare la *fedeltà*: le feature indicate come rilevanti sono davvero quelle che guidano la decisione del modello?

6.1 Procedura MoRF

Il metodo *Most Relevant First* (MoRF) è un approccio standard in letteratura per testare la fedeltà delle mappe di importanza [16, 17]. L'idea è semplice: se le feature indicate come importanti sono davvero decisive, rimuoverle dovrebbe ridurre rapidamente la confidenza del modello nella classe target.

Il procedimento seguito è stato il seguente:

1. Ordinare le feature o i pixel in base all'importanza decrescente indicata dalla mappa.
2. Mascherare progressivamente le più importanti, in $K = 10$ step uguali, partendo dalle più rilevanti.
3. Usare come *baseline* il valore medio dei pixel dell'immagine: questa scelta mantiene le immagini *in-distribution*, evitando artefatti dovuti a valori estremi come tutto nero o tutto bianco.
4. Dopo ogni mascheramento, registrare la probabilità che il modello assegna alla **classe vera** (*true label*).
5. Tracciare la curva MoRF, che mostra il decadimento della confidenza al crescere della porzione di immagine mascherata.

6.2 AOPC: Area Over the Perturbation Curve

Per riassumere in un solo numero la qualità di una spiegazione, è stata utilizzata la metrica **AOPC** (*Area Over the Perturbation Curve*), introdotta da Samek et al. [16], calcolata come:

$$\text{AOPC} = \frac{1}{K} \sum_{k=1}^{K} \left[f(x) - f(x^{(k)}) \right]$$

dove:

- $f(x)$ è la predizione del modello sull'immagine originale.
- $f(x^{(k)})$ è la predizione dopo aver mascherato i primi k blocchi di feature più importanti.
- $K = 10$ è il numero di step di mascheramento.

Più il valore AOPC è alto, più la rimozione delle feature considerate importanti provoca un crollo rapido della confidenza: un segno di elevata fedeltà della spiegazione.

Listing 6.1: Procedura MoRF e calcolo AOPC

```
function calcola_aopc(modello, immagine, mappa_spiegazione, steps=10):
    probas = []

    indici_importanti = ordina_pixel_per_rilevanza(mappa_spiegazione)

    for step in 0 to steps:
        immagine_modificata = maschera_pixel(immagine, indici_importanti,
            step)
        confidenza = predici_confidenza(modello, immagine_modificata)
        aggiungi(probas, confidenza)

    conf_iniziale = probas[0]
    differenze = [conf_iniziale - p for p in probas]
    aopc = media(differenze)

    return aopc, probas
```

6.3 Risultati AOPC

I risultati mostrano che:

- **LIME** provoca il decadimento più rapido della confidenza media, segno che le feature evidenziate sono spesso effettivamente rilevanti per il modello.
- **Integrated Gradients** ottiene un valore intermedio, combinando buona fedeltà con alta coerenza inter-modello.

Metodo	AOPC (media ± std)
LIME	0.7993 ± 0.0948
Integrated Gradients	0.7511 ± 0.1873
Saliency	0.6568 ± 0.1495

Tabella 6.1: AOPC per metodo (più alto = spiegazione più efficace), $n = 100$.

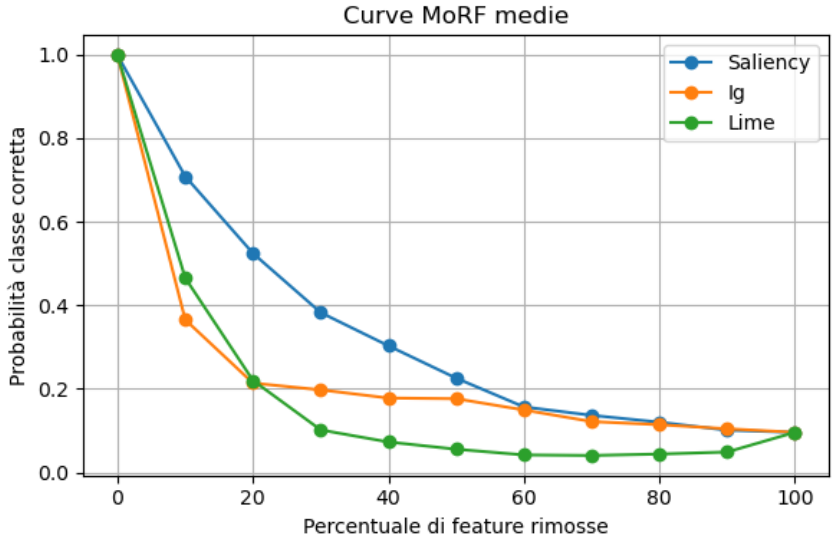


Figura 6.1: Curve MoRF medie: andamento della probabilità della classe corretta al crescere della percentuale di feature rimosse.

- **Saliency** registra valori più bassi, indicando che le feature evidenziate non sempre sono decisive per la classificazione.

La relazione tra questi risultati e la stabilità inter-modello è discussa in Sez. 7.1.

Tali risultati vanno interpretati con cautela. LIME, pur essendo più rumoroso nelle visualizzazioni e meno coerente tra modelli, ottiene il punteggio AOPC più alto. Questo suggerisce che, quando individua una feature come rilevante, essa ha un impatto marcato sulla decisione del modello.

Integrated Gradients si colloca leggermente sotto, offrendo un equilibrio tra fedeltà elevata e coerenza inter-modello: un compromesso interessante in scenari dove entrambe le proprietà sono desiderabili.

Saliency, infine, mostra valori più bassi, segnalando che alcune feature evidenziate potrebbero non essere cruciali per la classificazione.

È importante notare che un AOPC elevato non implica necessariamente maggiore interpretabilità per l'utente umano: misura soltanto la capacità della spiegazione di individuare feature la cui rimozione comporta un rapido calo di confidenza del modello. Inoltre, i valori sono influenzati da scelte implementative come la baseline

di mascheramento (media immagine) e, nel caso di LIME, dalla granularità delle perturbazioni (blocchi 4×4) e dal numero di campioni utilizzati.

7 | Discussione e implicazioni

L'analisi ha mostrato che, nel contesto di un *Rashomon set*, le differenze tra metodi di spiegazione possono risultare più marcate di quelle tra modelli ugualmente accurati. Questo rafforza l'idea che la scelta del metodo XAI sia cruciale almeno quanto quella dell'architettura del modello, quando l'obiettivo è comprendere le decisioni di un sistema di apprendimento automatico.

7.1 Trade-off tra fedeltà (AOPC) e stabilità (similarità)

Mettendo in relazione i risultati di *fedeltà* (AOPC; Tab. 6.1) con quelli di *stabilità* tra modelli (similarità inter-modello; Tab. 5.3), emerge un chiaro compromesso.

Evidenze quantitative. **LIME** massimizza in media la fedeltà ($AOPC$ 0.7993 ± 0.0948), indicando che le regioni identificate come rilevanti causano un rapido decadimento della confidenza quando rimosse. **Integrated Gradients (IG)** mostra invece una **stabilità inter-modello** molto elevata (es. Cosine 0.970 ± 0.013, SSIM 0.711 ± 0.077), producendo mappe più coerenti al variare dell'architettura. **Saliency** risulta intermedio/meno performante a seconda della metrica (es. Spearman inter-modello 0.688 ± 0.062), suggerendo un buon mantenimento del *ranking* pur con minore fedeltà rispetto a LIME.

Quando privilegiare cosa. In contesti di *debug/diagnosi*, dove è cruciale scoprire feature realmente decisive, può essere preferibile **LIME** (maggiore AOPC), accettando una variabilità superiore. In scenari di *comunicazione verso utenti finali* o documentazione, dove la consistenza tra modelli è prioritaria, risulta più adatto **Integrated Gradients** (maggiore stabilità). **Saliency** può essere utile quando interessa soprattutto preservare l'*ordinamento relativo* delle importanze (correlazioni di rango), più che la struttura spaziale o la magnitudine assoluta.

Limiti di interpretazione. Un AOPC elevato non implica automaticamente una migliore interpretabilità per l'utente umano: misura l'impatto della rimozione di regioni ritenute importanti, non la *chiarezza* della spiegazione. Inoltre, le grandezze osservate dipendono da scelte implementative come la baseline di mascheramento per MoRF e la granularità/numero di campioni in LIME.

7.2 Sintesi dei risultati principali

- La **variabilità intra-modello** è risultata più elevata della variabilità inter-modello, segnalando che metodi diversi possono produrre interpretazioni significativamente differenti anche sullo stesso modello.
- **LIME** ha mostrato la fedeltà più alta (AOPC), ma a costo di minore coerenza e maggiore rumorosità.
- **Integrated Gradients** ha offerto un buon compromesso tra stabilità e fedeltà, rendendolo adatto a scenari in cui entrambe le proprietà siano importanti.
- **Saliency** ha avuto prestazioni più basse in entrambe le metriche, suggerendo affidabilità limitata nel contesto MNIST e con configurazione base.

7.3 Implicazioni pratiche

I risultati suggeriscono che:

1. La scelta del metodo XAI deve considerare sia la stabilità rispetto a variazioni del modello sia la fedeltà rispetto alla decisione effettiva, non solo la leggibilità visiva.
2. In applicazioni critiche (es. medicina, finanza) può essere opportuno combinare più metodi per verificare la consistenza delle spiegazioni.
3. Il *Rashomon set* si rivela uno strumento efficace per validare la robustezza delle spiegazioni in scenari realistici, dove più modelli equivalenti possono coesistere.

7.4 Limiti e sviluppi futuri

Le principali limitazioni riguardano:

- l'uso di un dataset semplice (MNIST), che non riflette la complessità di domini reali;
- l'impiego di una sola architettura, che riduce la generalizzabilità dei risultati;
- l'analisi di un numero limitato di metodi XAI.

Lavori futuri potrebbero includere l'estensione a dataset più complessi (es. CIFAR-10, ImageNet), il confronto tra architetture differenti, l'inclusione di metodi più recenti (Grad-CAM, SHAP) e l'uso di metriche *user-centered* per valutare anche l'efficacia comunicativa delle spiegazioni.

Conclusioni

Questo studio ha proposto un approccio sperimentale basato sul *Rashomon set* per valutare la robustezza e la fedeltà delle spiegazioni fornite da diversi metodi XAI. Il workflow ha previsto:

- la costruzione di un insieme di modelli equivalenti in accuratezza;
- la generazione di spiegazioni tramite Saliency, Integrated Gradients e LIME;
- la valutazione della similarità (SSIM, Pearson, Spearman, Cosine, MAE) e della fedeltà (MoRF, AOPC).

L'analisi ha evidenziato che:

1. la variabilità dovuta al metodo è maggiore di quella dovuta al modello;
2. LIME eccelle in fedeltà ma presenta minore coerenza;
3. Integrated Gradients bilancia stabilità e fedeltà;
4. Saliency risulta meno performante in entrambi gli aspetti.

In sintesi, la robustezza delle spiegazioni non può essere assunta a priori e deve essere verificata sperimentalmente. L'approccio basato sul *Rashomon set* fornisce un quadro metodologico utile e riproducibile per indagare tali aspetti, ponendo le basi per futuri studi su dataset più complessi, architetture eterogenee e metodi di spiegazione avanzati.

Bibliografia

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[3] Leila Arras, Ahmed Osman, and Wojciech Samek. Evaluating recurrent neural network explanations. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2923–2931, 2019.

[4] Umang Bhatt, Adrian Weller, and Jose M. F. Moura. Evaluating and aggregating feature-based model explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3016–3022, 2021. doi: 10.24963/ijcai.2021/416.

[5] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. In *Journal of Machine Learning Research*, volume 20, pages 1–81, 2019.

[6] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, 2018. doi: 10.1145/3236009.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL https://arxiv.org/abs/1412.6980.

[8] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Brandon Reynolds, Alexander Melnikov, Nadezhda Kliushkina, Carlos Araya, Siqi Yan, and Serge Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch. In *Proceedings of the PyTorch Developer Conference*, 2020.

[9] Satyapriya Krishna, Tien Han, Jyun-Yu Gu, Rahul Puri, Himabindu Lakkaraju, and Sameer Singh. The disagreement problem in explainable machine learning:

A practitioner's perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 26688–26701, 2022.

[10] E. Leventi, A. Papadopoulos, and D. Tzovaras. Consistency and reliability of explainable ai methods. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[11] S. T. Mueller et al. The rashomon effect in explainable artificial intelligence. *Frontiers in Artificial Intelligence*, 6:1–14, 2023. doi: 10.3389/frai.2023.123456.

[12] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research: a report from the neurips 2019 reproducibility program. *Journal of Machine Learning Research*, 22(164):1–20, 2021.

[13] Lutz Prechelt. Early stopping – but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[14] Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*, 2017. URL https://arxiv.org/abs/1707.09861.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[16] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2016.

[17] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITW 2017 - Information Theory and Applications Workshop*, pages 1–10, 2017. doi: 10.1109/ITA.2017.8003443.

[18] Lesia Semenova, Cynthia Rudin, and Ron Parr. Existence of rashomon sets for classification. *arXiv preprint arXiv:1908.01755*, 2019.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[20] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR Workshop)*, 2014. URL https://arxiv.org/abs/1312.6034.

[21] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[22] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[23] tqdm contributors. tqdm: A fast, extensible progress bar for python and cli, 2016. Available at: https://github.com/tqdm/tqdm.

[24] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.