doc

Valentina piscopo

August 2025

Contents

1 Introduzione

Quando ci affidiamo ai metodi di spiegazione per i modelli di *machine learning*, sorge una domanda cruciale: *quanto sono simili le spiegazioni prodotte da diversi algoritmi o modelli, e come possiamo misurare questa similarità?*

Nel contesto del *Rashomon effect* [11, 12], 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) [17], 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. Questa tesi adotta 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).

L'obiettivo finale è indagare se e come l'effetto Rashomon si manifesti non solo nelle predizioni, ma anche nell'interpretabilità, valutando la coerenza e l'affidabilità dei metodi XAI in scenari controllati.

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, 11, 12]:

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 [16].
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, 17].
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* [17].

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, 12]:

- **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.

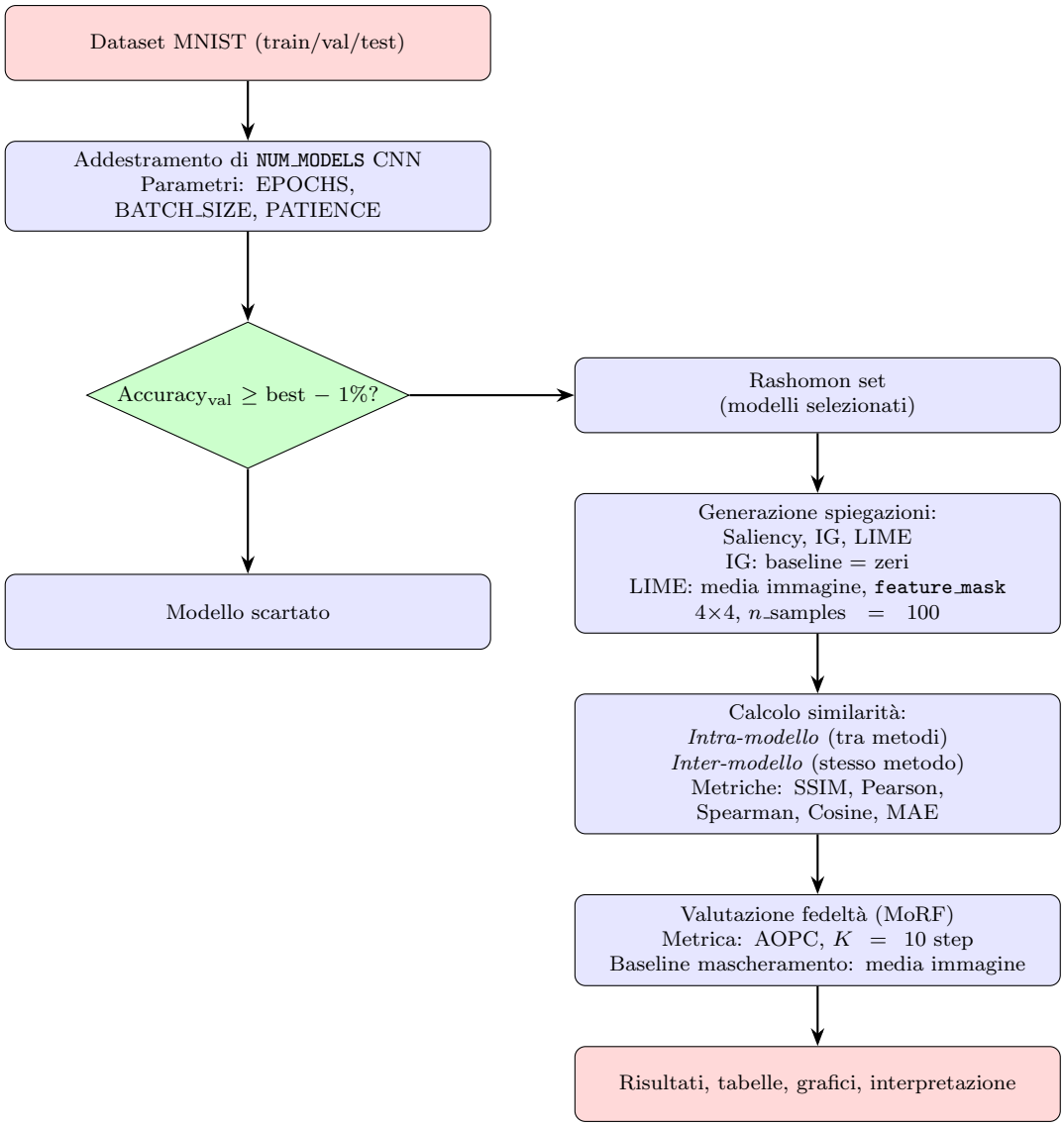

Figure 1: Pipeline dell'esperimento: addestramento e selezione del Rashomon set, generazione delle spiegazioni, confronto di similarità e valutazione della fedeltà (MoRF/AOPC).

- **Captum** [9]: 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*.
- **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. [13]:

- **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 [11].

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. [16] 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].

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 [17].

Selezione del Rashomon set. La scelta dei modelli è stata guidata dalla *validation accuracy*, selezionando tutti quelli entro l'1% dal miglior modello, seguendo la definizione operativa di Mueller et al. [12]. Questo garantisce che le differenze osservate nelle spiegazioni non siano dovute a variazioni significative in termini di performance di classificazione. Ogni modello è stato salvato in formato `.pt` di PyTorch, assicurando la riproducibilità esatta 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 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	50
Patience (early stopping)	5 epoche
Soglia Rashomon	1% di differenza rispetto al miglior modello (val. acc.)
Batch size	128
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)

Table 1: Hyperparametri e setup utilizzati negli esperimenti.

3 Il Rashomon set

Il *Rashomon set* è una collezione di modelli diversi che ottengono prestazioni simili su un determinato compito. Il termine deriva dall'*effetto Rashomon*, che descrive la possibilità che più spiegazioni, tutte coerenti con i dati osservati, possano emergere per lo stesso fenomeno [12]. In ambito *machine learning*, ciò implica che, anche a parità di accuratezza, modelli con parametri differenti possono fornire spiegazioni distinte per le proprie decisioni [5, 18].

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* [14]: per ogni modello, l'allenamento 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

Al termine dell'addestramento, sono stati selezionati tutti i modelli che ottenevano un'accuratezza sul validation entro una certa soglia rispetto al modello con la miglior performance. In questo esperimento la soglia è pari all'1%:

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

dove:

- $\text{Accuracy}_{\text{val}}(m_{\text{best}})$ è la miglior accuratezza sul validation ottenuta tra tutti i modelli addestrati;
- ϵ è la tolleranza fissata (0.01 in questo caso), in linea con la definizione operativa di Rashomon set [5].

Questo criterio garantisce che i modelli selezionati siano equivalenti dal punto di vista delle prestazioni predittive, pur potendo avere rappresentazioni interne e logiche decisionali differenti [11, 12].

Listing 1: Addestramento modelli e selezione Rashomon set

```
for i in 1 to NUM_MODELS:
    imposta_seed(SEED + i)

    modello, acc_validazione = addestra_modello(dati_train, dati_val)
```

```
5
6     if acc_validazione > best_acc_validazione:
7         aggiorna_best_accuracy(acc_validazione)
8
9     if acc_validazione >= best_acc_validazione - SOGLIA_RASHOMON:
10        aggiungi_a_rashomon_set(modello)
11        salva_modello(modello)
```

3.3 Obiettivi

La costruzione del Rashomon set serve a due scopi principali:

1. **Studiare la variabilità delle spiegazioni**: se modelli ugualmente accurati forniscono spiegazioni simili, allora queste possono essere considerate robuste; se divergono, si manifesta l'effetto Rashomon anche nell'interpretabilità [12].
2. **Testare i metodi di spiegazione**: usando un insieme di modelli equivalenti, è possibile valutare la coerenza e la fedeltà delle spiegazioni prodotte da diversi algoritmi XAI [11].

3.4 Approfondimento: equivalenza tra modelli

Nel contesto di questo lavoro, due modelli sono considerati equivalenti se soddisfano la condizione di soglia sull'accuratezza definita sopra. 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 [12].
- Esplorare la diversità interna tra modelli che, in termini predittivi, sembrano "uguali", ma che possono differire nelle rappresentazioni interne e quindi nelle spiegazioni [11].

3.5 Perché utilizzare l'early stopping

L'*early stopping* [14] 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 [6].

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) [15], è possibile distinguere la variabilità dovuta al caso da quella legata a differenze strutturali nei modelli o nelle spiegazioni.

3.6 Scelta dell'ottimizzatore: Adam.

Per l'addestramento di tutte le reti neurali è stato utilizzato l'ottimizzatore *Adam* (*Adaptive Moment Estimation*) [8], ampiamente adottato in ambito *deep learning* per la sua capacità di combinare i vantaggi di AdaGrad e RMSProp. 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, 7].

4.1 Saliency

Il metodo della *saliency map* [17, 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*) [16] 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 2: Generazione delle spiegazioni

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

    spiegazioni["Saliency"] = calcola_saliency(modello, immagine,
        etichetta_vera)
    spiegazioni["IG"] = calcola_integrated_gradients(
        modello, immagine, etichetta_vera, baseline_zeri)
    spiegazioni["LIME"] = calcola_lime(
        modello, immagine, etichetta_vera,
        campioni=200, maschera_feature=FEATURE_MASK,
        baseline_media_pixel)

    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) [16, 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 [7, 17].

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 [7, 16]. 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 "a occhio" 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) [12, 17].

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 [11, 17].

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, 10–12]:

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*). 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.

5.3 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	0.529	0.194	0.651	0.211	100
Saliency–LIME	0.178	0.468	0.392	0.663	0.177	100
IG–LIME	0.247	0.421	0.224	0.780	0.167	100

Table 2: Similarità *intra-modello*: confronto tra metodi diversi sullo stesso modello (medie; dallo script non sono disponibili le deviazioni standard).

Metodo	SSIM	Pearson	Spearman	Cosine	MAE	n
Saliency	0.439	0.612	0.688	0.727	0.069	450
IG	0.711	0.723	0.516	0.970	0.083	450
LIME	0.639	0.864	0.541	0.929	0.093	450

Table 3: Similarità *inter-modello*: stesso metodo applicato a modelli diversi del Rashomon set (medie; dallo script non sono disponibili le deviazioni standard).

Figure 2: 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).

5.4 Interpretazione

I valori mostrano una tendenza chiara:

- La similarità **inter-modello** è più alta per tutte le metriche rispetto all'intra-modello, indicando che un singolo metodo tende a produrre spiegazioni coerenti tra modelli diversi del Rashomon set.
- La similarità **intra-modello** è sensibilmente più bassa: metodi diversi, anche sullo stesso modello, producono spiegazioni tra loro divergenti.

Guardando ai risultati, emerge una tendenza coerente con quanto riportato in letteratura: la similarità **inter-modello** è sistematicamente più alta della **intra-modello**, indipendentemente dalla metrica utilizzata. Ad esempio, LIME raggiunge valori di Pearson pari a 0.864 tra modelli diversi, mentre la stessa metrica scende a 0.421–0.529 quando si confronta LIME con altri metodi sullo stesso modello.

La situazione si inverte quando cambiamo il metodo: qui le similarità calano drasticamente, con SSIM compresi tra 0.178 e 0.263 e correlazioni Spearman che, in certi confronti, non superano 0.392. Questi numeri confermano che la *scelta del metodo di spiegazione* è il fattore che più influenza forma, intensità e posizionamento delle regioni considerate rilevanti.

In altre parole, i metodi "raccontano storie diverse" anche quando osservano lo stesso modello. Questo suggerisce che, 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, almeno all'interno del Rashomon set considerato.

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 questo 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. 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*), 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 3: 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

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

Table 4: AOPC per metodo (più alto = spiegazione più efficace), $n = 100$.

I risultati indicano:

- **LIME** provoca il decadimento più rapido della confidenza media, segno che le feature che evidenzia sono spesso effettivamente rilevanti per il modello.
- **Integrated Gradients** ottiene un valore intermedio, combinando buona fedeltà con alta coerenza inter-modello.
- **Saliency** ha valori più bassi, suggerendo che le feature evidenziate non sempre sono decisive per la classificazione.

Questi valori vanno letti con attenzione. LIME, pur essendo il più rumoroso nelle visualizzazioni e meno coerente tra modelli, ottiene il punteggio AOPC più alto. Ciò indica che, quando individua una feature come rilevante, questa ha effettivamente un impatto forte sulla decisione del modello.
Integrated Gradients si colloca appena sotto, bilanciando fedeltà elevata e coerenza inter-modello: un compromesso interessante in scenari dove entrambe le proprietà sono desiderabili.
Saliency mostra valori più bassi, suggerendo che alcune feature evidenziate non sono sempre cruciali per la classificazione.

È importante sottolineare che un AOPC alto non implica necessariamente una spiegazione più interpretabile o comprensibile per l'utente umano: misura solo la capacità della spiegazione di identificare feature che, se rimosse, fanno crollare la confidenza del modello. Inoltre, i valori dipendono 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, fattori che influenzano sensibilmente il risultato.

6.4 Conclusioni

L'analisi combinata di similarità e fedeltà porta a diverse osservazioni:

- La variabilità intra-modello è maggiore di quella inter-modello: il metodo di spiegazione influenza più del modello stesso.
- LIME eccelle in termini di AOPC, ma soffre di maggiore variabilità e rumorosità, specialmente tra modelli diversi.
- Integrated Gradients offre un buon compromesso: alta coerenza tra modelli e fedeltà elevata.
- Saliency è meno performante in entrambi gli aspetti, suggerendo una minore utilità pratica nel contesto MNIST.

References

[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] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. URL `https://www.deeplearningbook.org/`.

[7] 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.

[8] 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`.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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`.

[16] 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.

[17] 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.

[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.