



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali  
Corso di Laurea in Informatica

Tesi di Laurea

**EXPLAINABLE AI: STUDIO ED  
IMPLEMENTAZIONE DI TECNICHE DI  
MACHINE LEARNING INTERPRETABILI**

**EXPLAINABLE AI: STUDY AND  
IMPLEMENTATION OF INTERPRETABLE  
MACHINE LEARNING TECHNIQUES**

**AKINJOLAGBE SAMUEL FOLABI AKINOLA**

Relatore: *Andrea Ceccarelli*

Anno Accademico 2020-2021

Akinjolagbe Samuel Folabi Akinola: *Explainable AI: studio ed implementazione di tecniche di machine learning interpretabili*, Corso di Laurea in Informatica, © Anno Accademico 2020-2021

---

## INDICE

---

1	Introduzione	5
1.1	Struttura tesi	5
2	Quadro Teorico	7
2.1	Intelligenza Artificiale	7
2.1.1	Tipologie di Intelligenza Artificiale	8
2.2	Machine Learning	9
2.2.1	Funzionamento	9
2.2.2	Metodi di apprendimento	10
2.3	Reti Neurali	12
2.3.1	Tipologie di reti neurali	12
2.3.2	Activation Function	13
2.3.3	Misurare le prestazioni: Le metriche	14
2.3.4	Ottimizzatori	16
2.3.5	Frameworks	17
2.3.6	Reti Neurali Convoluzionali	17
2.4	Deep Learning	19
2.5	Importanza dell'Explainable AI	20
2.5.1	Perché se ne ha bisogno	21
2.5.2	AI Fairness	22
3	Explainable AI	23
3.1	Cosa è	23
3.2	Obiettivi	24
3.3	Contenuti	24
3.4	Tipologie	25
3.4.1	Interpretazioni post-hoc	25
4	Tecniche	27
4.1	Tecniche interpretabili	27
4.1.1	Regressione Lineare	27
4.1.2	Regressione Logistica	27
4.1.3	Foresta Casuale	28
4.1.4	XGBoost	28
4.1.5	Alberi Decisionali	29
4.1.6	K-Nearest Neighbor	29
4.2	Tecniche model-agnostic	29
4.2.1	LIME	30

4.2.2	Shapley values	30
4.2.3	SHAP	31
4.3	Precisione vs Interpretabilità	32
4.4	Cenni riguardo gli Adversarial Attacks	33
4.4.1	Classificazione	33
4.4.2	Strategie	34
5	Strumenti	35
5.1	Applicativi XAI	35
5.1.1	AIX360	35
5.1.2	ELI5	36
5.1.3	LIME	36
5.1.4	SHAP	38
5.1.5	Skater	38
5.1.6	tf-explain	39
5.1.7	What-If Tool	39
5.2	Applicativi Adversarial Attacks	40
5.2.1	Adversarial Robustness Toolbox	40
5.2.2	FoolBox	41
5.2.3	AdvBox	41
5.3	Librerie scelte	41
6	Tecniche utilizzate	43
6.1	Strumenti utilizzati	43
6.1.1	AkinolaVGG16	44
6.1.2	Struttura	45
6.1.3	Motivazioni	46
6.2	Esperimenti	47
6.2.1	LIME	47
6.2.2	SHAP Gradient Explainer	50
6.2.3	SHAP Kernel Explainer	54
6.2.4	tf-explain	57
6.2.5	Adversarial Robustness Toolbox	59
7	Conclusione	63
7.1	Possibili sviluppi / Lavori Futuri	64
A	Materiale aggiuntivo	65
A.1	LIME	65
A.2	SHAP Gradient Explainer	66
A.3	SHAP Kernel Explainer	70
A.4	tf-explain	71

---

## ELENCO DELLE FIGURE

---

Figura 1	Rappresentazione di Intelligenza Artificiale, Machine Learning, Reti Neurali e Deep Learning come bambole russe	8
Figura 2	Schema dei vari tipi di apprendimento	11
Figura 3	(a) Neurone biologico (b) Simulazione matematica del neurone [28]	12
Figura 4	Schema delle ANN più utilizzate [42]	13
Figura 5	Grafici delle funzioni di attivazione[28]	14
Figura 6	Schema base di una CNN[14]	18
Figura 7	Differenza tra Deep Learning e i vecchi algoritmi di Machine Learning	20
Figura 8	Confronto di User Experience con e senza XAI [18]	23
Figura 9	Confronto della varie interpretazioni post-hoc [26]	26
Figura 10	Confronto tra regressione lineare e regressione logistica	28
Figura 11	Confronto tra alberi decisionali, GBDT e Foresta Casuale	29
Figura 12	K-Nearest Neighbor	29
Figura 13	(a) Esempio di LIME con immagini (b) Esempio di LIME con testo [33]	31
Figura 14	Sintesi del funzionamento di SHAP[32]	31
Figura 15	Correlazione tra accuratezza e interpretabilità nei modelli di ML	32
Figura 16	Miglioramento delle tecniche tramite XAI	33
Figura 17	(a)immagine segmentata (b) immagine perturbata	37
Figura 18	Visualizzazione dei risultati di Grad-Cam[41]	40
Figura 19	Immagini utilizzate per gli esperimenti	44
Figura 20	Prestazioni del modello	46
Figura 21	Risultati predetti dai modelli	47
Figura 22	Spiegazione di LIME per i modelli VGG19 e MobileNet	49
Figura 23	Risultati predetti dai modelli	50
Figura 24	Spiegazione di SHAP Gradient Explainer per i layer di VGG19	52

4 Elenco delle figure

Figura 25	Spiegazione di SHAP Gradient Explainer per i layer di AkinolaVGG16	53
Figura 26	Risultati predetti dai modelli	54
Figura 27	Risultati di SHAP Kernel Explainer per i modelli InceptionV3 e AkinolaVGG16	56
Figura 28	Risultati predetti dai modelli	57
Figura 29	Spiegazione di tf-explain per AkinolaVGG16	59
Figura 30	Risultati ottenuti utilizzando l'immagine origina- le	60
Figura 31	Risultati ottenuti utilizzando l'immagine modifica- ta da DeepFool	61
Figura 32	Risultati ottenuti utilizzando l'immagine modifica- ta da Fast Gradient Method	61
Figura 33	Risultati ottenuti con AkinolaVGG16	65
Figura 34	Risultati ottenuti con InceptionV3	65
Figura 35	Risultati ottenuti con VGG16	66
Figura 36	Risultati ottenuti con InceptionV3	67
Figura 37	Risultati ottenuti con MobileNet	68
Figura 38	Risultati ottenuti con VGG16	69
Figura 39	Risultati ottenuti con Akinola	70
Figura 40	Risultati ottenuti con MobileNet	70
Figura 41	Risultati ottenuti con VGG19	70
Figura 42	Risultati ottenuti con InceptionV3	71
Figura 43	Risultati ottenuti con MobileNet	71
Figura 44	Risultati ottenuti con VGG16	72
Figura 45	Risultati ottenuti con VGG19	72

# 1

---

## INTRODUZIONE

---

Il veloce sviluppo delle tecnologie, ha causato un forte incremento nell'utilizzo di sistemi di Intelligenza Artificiale (IA). Questi sistemi stanno penetrando in una vasta gamma di settori, come l'istruzione, l'edilizia, la sanità, l'intrattenimento, la finanza e molte altri. Il loro ruolo sta diventando sempre più parte della vita quotidiana influenzando ciò che si compra, chi si assume, con chi si è amici, che notizie si ricevono e anche come si viene curati. Tuttavia, in genere si ha poca comprensione del motivo per cui, i sistemi di IA, prendono determinate decisioni. La crescente complessità di questi sistemi, sta rendendo sempre più difficile la loro comprensione, infatti, la loro architettura può essere così complessa che nemmeno gli esperti sono più in grado di spiegare queste decisioni. Nonostante ciò, capita spesso che a questi sistemi sia data una piena fiducia, ma bisognerebbe chiedersi "Che cosa dovrebbe essere la base di questa fiducia?". Per essere affidabili, oltre a fornire informazioni sull'accuratezza della previsione e su altri aspetti riguardanti le prestazioni, questi sistemi dovrebbero fornire agli utenti una efficace spiegazione del loro comportamento e le motivazioni che hanno portato a determinate scelte. L'obbiettivo di questa tesi, è mostrare in che modo sia possibile rendere queste scelte interpretabili.

### 1.1 STRUTTURA TESI

Per raggiungere l'obbiettivo, inizialmente, si tratta la parte teorica, presentando quindi, i concetti che si utilizzano nei capitoli successivi per analizzare e discutere gli utilizzi e l'importanza di una Intelligenza Artificiale interpretabile. Dopo ciò, si mostrano alcune tra le più importanti librerie per poi, infine, utilizzare una selezione di queste per effettuare degli esperimenti tramite i quali si spiegano le decisioni prese dai modelli di IA.

Il lavoro è suddiviso nel seguente modo:

- **Capitolo 2:** Quadro teorico riguardante i sistemi di Intelligenza Artificiale;
- **Capitolo 3:** Introduzione all'Explainable AI, discussione della sua importanza ed i suoi obiettivi;
- **Capitolo 4:** Panoramica delle tecniche di Machine Learning utilizzate al giorno d'oggi;
- **Capitolo 5:** Analisi e scelta dei strumenti più adatti per effettuare gli esperimenti;
- **Capitolo 6:** Esperimenti: motivazione degli strumenti utilizzati ed applicazione di tecniche di Explainable AI per l'interpretazione dei modelli di Machine Learning;
- **Capitolo 7:** Conclusioni e possibili sviluppi;
- **Appendice A:** Risultati degli esperimenti non analizzati nel capitolo 6.

# 2

---

## QUADRO TEORICO

---

La tecnologia si sta inserendo sempre di più nella vita di tutti i giorni e, per stare al passo con le aspettative dei consumatori, si fa sempre più affidamento sugli algoritmi di apprendimento automatico. Si possono osservare dei semplici esempi nei social media (attraverso il riconoscimento degli oggetti nelle foto) o quando si parla direttamente con dei dispositivi. Queste tecnologie sono comunemente associate all’Intelligenza Artificiale, al Machine Learning, alle Reti Neurali e al Deep Learning, e, sebbene tutte queste abbiano un ruolo specifico, questi termini vengono spesso usati in modo intercambiabile, cosa che può portare confusione in ciò che li differenzia [7]. Il modo più semplice di pensare ad Intelligenza Artificiale, Machine Learning, Reti Neurali e Deep Learning è come insieme e sottoinsiemi ovvero, ognuno è parte dell’insieme precedente (vedi Figura 1).

### 2.1 INTELLIGENZA ARTIFICIALE

Si può definire Intelligenza Artificiale il processo che permette, a computer e macchine, di simulare percezione, apprendimento, abilità di problem solving e processo decisionale dell’essere umano. Il primo esempio di questa idea, si ha quando nel 1950, Alan Turing pubblica “Computing Machinery and Intelligence”. Nell’articolo, Turing si propone di rispondere alla domanda “Può una macchina pensare ?” [1].

Nonostante le grandi possibilità, dalla sua prima definizione nel 1956 [1], questo argomento è stato relegato perlopiù ad immaginazione e fantascienza, fino a quando negli ultimi anni, soprattutto dal 2015 [8, 6], AI (termine inglese per IA) è diventata molto popolare grazie ai grandi volumi di dati, algoritmi avanzati e GPU che permettono di rendere il calcolo parallelo più economico, veloce e potente. AI è il completamento del testo mentre si scrive, indicazioni stradali fornite in tempo reale,

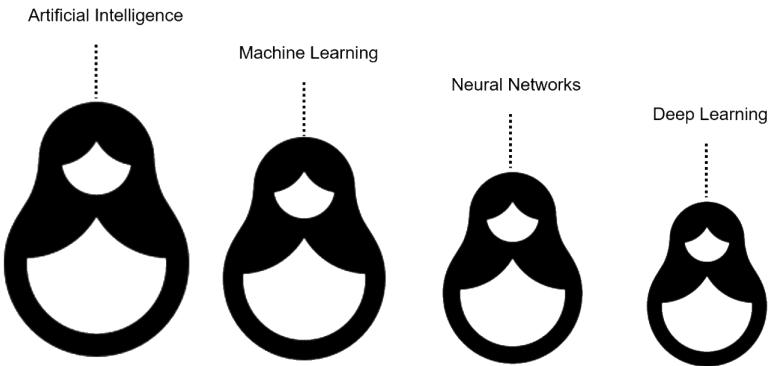


Figura 1.: Rappresentazione di Intelligenza Artificiale, Machine Learning, Reti Neurali e Deep Learning come bambole russe

raccomandazioni su cosa comprare o su che serie TV spendere notti insonni.

### 2.1.1 *Tipologie di Intelligenza Artificiale*

I sistemi di AI si possono categorizzare dall'abilità di replicare le caratteristiche umane, l'applicazione delle loro tecnologie e l'utilizzo nel mondo reale. Esistono principalmente tre categorie [1]:

- **Artificial Narrow Intelligence (ANI):** AI allenata e focalizzata nello eseguire specifici compiti. Rappresenta la maggior parte di AI presente al giorno d'oggi, ad esempio assistenti vocali come Cortana, Siri o Alexa, ecco perché si preferisce il termine Narrow Intelligence all'alternativa Weak AI in quanto non si possono definire queste tecnologie "deboli".
- **Artificial General Intelligence (AGI):** AI con capacità pari a quelle di un essere umano, ovvero, capace di risolvere molti tipi o classi di problemi e addirittura scegliere i problemi che vuole risolvere senza l'intervento umano. Questa tecnologia, detta anche Strong AI, è ancora completamente teorica senza nessun esempio pratico di uso ad ora.
- **Artificial Super Intelligence (ASI):** nonostante AGI sia ancora in una fase teorica, ciò non ha fermato i ricercatori dall'esplorare una

Intelligenza Artificiale con intelligenza e abilità superiori all'essere umano.

## 2.2 MACHINE LEARNING

Il Machine Learning (ML) è un sottoinsieme dell'Intelligenza Artificiale, il quale si concentra nel creare applicazioni che imparano dai dati migliorando la loro precisione nel tempo, senza il bisogno di essere programmate per farlo [2]. In questo contesto, gli algoritmi vengono addestrati per trovare particolari caratteristiche in un enorme numero di dati, così da poter predire in modo corretto, quelli che gli verranno sottoposti successivamente. Quindi, migliore è l'algoritmo e più accurate saranno le sue decisioni e predizioni. Degli esempi di Machine Learning possono essere:

- Assistenti vocali,
- Macchine con guida autonoma,
- Sistemi di raccomandazione,
- Spam detectors,
- Strumenti per l'analisi di immagini mediche.

Inoltre, essendo un tema in continuo sviluppo, grazie alla crescita dei big data, tecnologie che diventano più potenti ed economiche e con data scientist che sviluppano algoritmi migliori, il ML ha grandi possibilità di migliorare ulteriormente rendendo più efficienti le vite personali e lavorative.

### 2.2.1 *Funzionamento*

Ci sono essenzialmente quattro passi per creare un'applicazione di Machine Learning (o modello) [2]:

1. **Selezionare e preparare il training dataset:** ovvero, selezionare un insieme di dati rappresentativi che il modello utilizza per allenarsi a risolvere il problema per il quale è stato creato.
2. **Scegliere un algoritmo da utilizzare nel training dataset:** il tipo di algoritmo utilizzato dipende dal tipo e il numero di dati nel training dataset:

- Per i dati **Labeled** (con etichetta) spesso si utilizzano:
    - Algoritmi di regressione
    - Alberi decisionali
    - Classificatori basati su istanze
  - Per i dati **Unlabeled** (senza etichetta) spesso si utilizzano:
    - Algoritmi di Clustering
    - Regole di associazione
    - Reti neurali
3. **Allenare l'algoritmo per creare il modello:** processo iterativo che implica l'esecuzione di dati attraverso l'algoritmo, il confronto dell'output con un insieme di validazione (o validation set in inglese), la regolazione di pesi e bias<sup>1</sup> all'interno dell'algoritmo e l'esecuzione di nuovi dati fino a quando l'algoritmo restituisce i risultati corretti per la maggior parte del tempo.
4. **Utilizzo e miglioramento del modello:** utilizzo del modello con nuovi dati e, nel migliore dei casi, migliorare l'accuratezza e l'efficacia nel tempo.

### 2.2.2 *Metodi di apprendimento*

Sulla base del tipo di dati disponibili e gli scopi della ricerca, bisogna definire quali algoritmi di apprendimento da utilizzare, questi possono essere divisi in quattro principali categorie [29]:

**APPRENDIMENTO SUPERVISIONATO** Noto anche come supervised learning, utilizza un training set per insegnare ai modelli come produrre l'output desiderato. Questo set di dati include input e output corretti che consentono al modello di apprendere nel tempo. L'algoritmo misura la sua accuratezza attraverso la loss function, regolandosi fino a quando questa non è stata sufficientemente ridotta al minimo.

---

<sup>1</sup> Il termine bias riguarda la modalità in cui il modello dà importanza ad alcune delle funzionalità, il suo scopo è quello di aiutare a generalizzare meglio e a rendere i modelli meno sensibili a singoli dati.

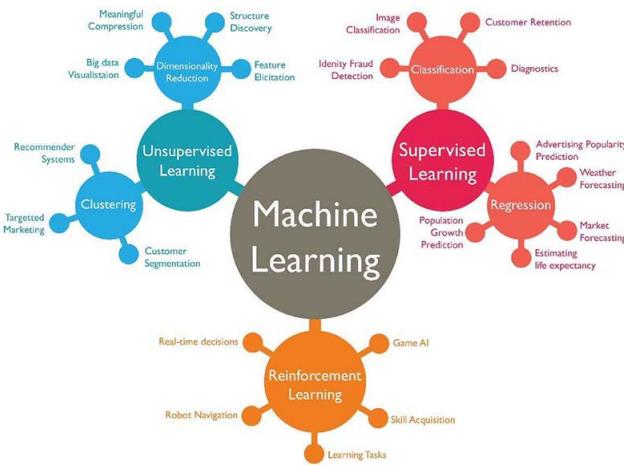


Figura 2.: Schema dei vari tipi di apprendimento

**APPRENDIMENTO NON SUPERVISIONATO** Noto anche come unsupervised learning, prende una grande quantità di dati e utilizza algoritmi per estrarre caratteristiche importanti, necessarie per etichettare, ordinare e classificare i dati senza il bisogno dell'intervento umano. Questo tipo di metodo riguarda meno l'automatizzazione di decisioni e più l'identificazione di pattern e relazioni non visibili all'occhio umano.

**APPRENDIMENTO SEMI-SUPERVISIONATO** Noto anche come semi-supervised learning, è una via di mezzo tra l'apprendimento supervisionato e non supervisionato. Durante l'addestramento, utilizza un insieme di dati con etichetta più piccolo in modo da guidare la classificazione e l'estrazione di funzionalità di un insieme di dati senza etichetta più grande. L'apprendimento semi-supervisionato può risolvere il problema di non avere abbastanza dati con etichetta per addestrare un algoritmo di apprendimento supervisionato.

**APPRENDIMENTO PER RINFORZO** Noto anche come reinforcement learning, è un modello di apprendimento automatico comportamentale simile all'apprendimento supervisionato, ma l'algoritmo non viene addestrato utilizzando dati di esempio. Questo modello impara procedendo per tentativi.

In questa tesi ci si concentra sul Supervised Learning.

### 2.3 RETI NEURALI

Il cervello umano, interpreta le situazioni del mondo reale in un modo che i computer non possono replicare. Le reti neurali, sviluppate per la prima volta negli anni '50, sono utilizzate per affrontare questo problema [3]. Conosciute anche come Artificial Neural Networks (ANN) sono un tentativo di simulare la rete di neuroni che compone il cervello umano (vedi Figura 3), in modo che un computer sia in grado di simulare apprendimento e processo decisionale di una persona. Le ANN vengono create programmando i computer, in modo che si comportino come se esse fossero cellule cerebrali interconnesse [9]. Queste sono costituite da strati di nodi, contenenti un input layer (strato di input), uno o più hidden layers (strati nascosti) e un output layer (strato di output). La componente principale di una ANN sono i neuroni artificiali, ognuno di questi riceve degli input da più neuroni, li moltiplica per un peso assegnato, li somma, eventualmente aggiungendo il bias, applica una activation function e passa il risultato a uno o più neuroni. Se l'uscita di un singolo nodo è superiore al valore di soglia specificato, quel nodo viene attivato, inviando i dati al livello successivo della rete, in caso contrario, non viene passato nessun dato al livello successivo della rete.

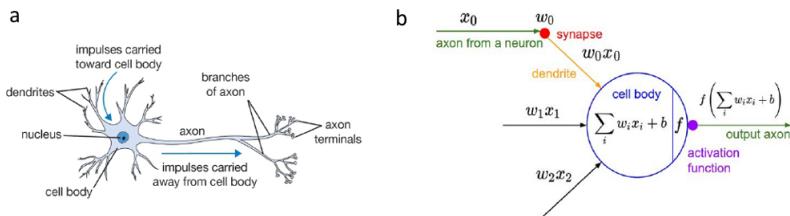


Figura 3.: (a) Neurone biologico (b) Simulazione matematica del neurone [28]

#### 2.3.1 Tipologie di reti neurali

Le reti neurali possono essere classificate in diversi tipi, i quali sono utilizzati per motivi differenti. Nonostante non sia un insieme completo, i seguenti rappresentano le reti neurali più utilizzate per risolvere i problemi più comuni (vedi Figura 4).

**FEEDFORWARD NEURAL NETWORK (FFNN)** Le FFNN sono una delle forme più semplici di ANN, in cui i dati o l'input viaggiano in una sola

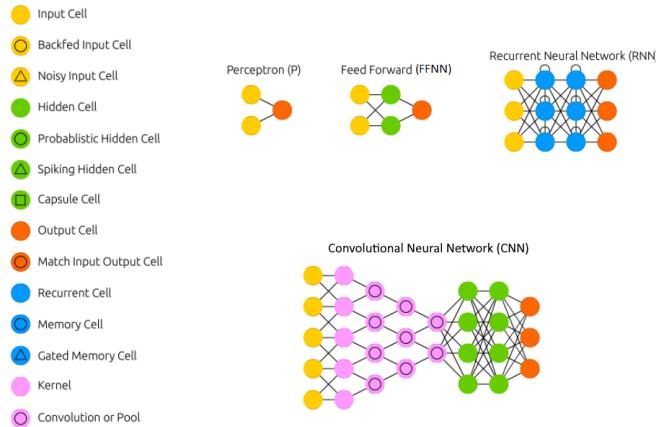


Figura 4.: Schema delle ANN più utilizzate [42]

direzione. I dati passano attraverso i nodi di input e escono sui nodi di output. Questa rete neurale può avere o meno dei livelli nascosti, nel caso non ne abbia viene indicata come perceptron (o perceptron)

**RECURRENT NEURAL NETWORK (RNN)** Le RNN sono un tipo di ANN che utilizzano dati sequenziali o dati di serie temporali. Questi algoritmi sono comunemente usati per problemi ordinali o temporali, come traduzione, auto correttori, riconoscimento vocale e altri. Questo significa che l'ordine dei dati ricevuti dal neurone è importante.

**CONVOLUTIONAL NEURAL NETWORK (CNN)** Le CNN sono tra le ANN più popolari. Questi modelli vengono utilizzati in diverse applicazioni e sono particolarmente diffusi nei progetti di elaborazione di immagini e video. Data la loro importanza si preferisce tornare sull'argomento successivamente.

### 2.3.2 Activation Function

Questa funzione viene utilizzata per introdurre la non linearità nell'output di un neurone. Questa decide se un neurone deve essere attivato oppure no, calcolando la somma pesata e aggiungendovi il bias. Se non applicassimo una funzione di attivazione il segnale di output sarebbe solo una funzione lineare. Le activation function più utilizzate sono (vedi Figura 5):

- **ReLU:** i suoi valori vanno da zero a infinito.  $[0, \infty)$ . Fornisce un

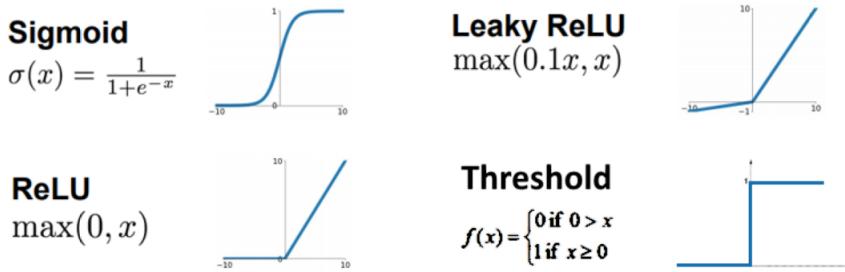


Figura 5.: Grafici delle funzioni di attivazione[28]

output "x" se  $x$  è positivo o altrimenti. Un problema è dato dal fatto che durante la fase di allenamento, ReLU, potrebbe provocare la morte di neuroni. Per risolvere questo problema è stato introdotto Leaky ReLU. Questo introduce una piccola pendenza per mantenere in vita gli aggiornamenti. Leaky ReLU varia da  $-\infty$  a  $+\infty$ .

- **Sigmoid**: è una funzione matematica avente una caratteristica curva a forma di "S" o curva sigmoide che varia tra 0 e 1, viene utilizzata per i modelli in cui è necessario ottenere come output una probabilità. Lo svantaggio della funzione di attivazione sigmoide è che può causare il blocco della rete neurale al momento dell'addestramento se viene fornito un forte input negativo.
- **Threshold**: è una funzione di attivazione basata su soglia. Se il valore di input è al di sopra o al di sotto di una certa soglia, il neurone viene attivato o meno e invia esattamente lo stesso segnale al livello successivo.

### 2.3.3 Misurare le prestazioni: Le metriche

Le metriche sono utilizzate per calcolare le performance del modello, in seguito vengono elencate le più importanti.

**CONFUSION MATRIX** Per precisione, la matrice di confusione non è una metrica, ma è uno strumento necessario per calcolare altre metriche, infatti è una rappresentazione tabellare delle performance di un classificatore.

		Truth					
		Asphalt	Concrete	Grass	Tree	Building	Total
Predicted	Asphalt	2385	4	0	1	4	2394
	Concrete	0	332	0	0	1	333
	Grass	0	1	908	8	0	917
	Tree	0	0	0	1084	9	1093
	Building	12	0	0	6	2053	2071
	Total	2397	337	908	1099	2067	6808

La matrice di confusione contiene:

- **TP (True Positives)**: Tutte le istanze di A classificate come A.
- **TN (True Negatives)**: Tutte le non istanze di A non classificate come A.
- **FP (False Positives)**: Tutte le non istanze di A classificate come A.
- **FN (False Negatives)**: Tutte le istanze di A non classificate come A.

**ACCURACY** L'accuracy rappresenta quanto accurato il modello è stato nel predire il risultato corretto.

$$\text{accuracy}(\hat{y}, y) = \begin{cases} 1 & \text{se } \hat{y} = y \\ 0 & \text{altrimenti} \end{cases}$$

**Precision** La precision è un numero in [0,1] che misura quanto è accurato il classificatore.

$$\frac{\text{TP}}{(\text{TP} + \text{FP})}$$

**Recall** La recall è un numero in [0,1] che misura la percentuale di elementi correttamente classificati, sul totale degli elementi di quella classe.

$$\frac{\text{TP}}{(\text{TP} + \text{FN})}$$

**LOSS FUNCTION** La loss function è una funzione per valutare le prestazioni di un algoritmo. Se le previsioni sono completamente sbagliate, questa produce un numero alto mentre se sono buone un numero basso. Fondamentalmente durante la modifica dell'algoritmo la loss function permette di capire se si sta migliorando oppure no. Quindi consente di capire di quanto i valori previsti sono diversi da quelli effettivi. Delle categorie di loss function sono:

- **Regression:** Si occupano di predire dei valori continui, ad esempio data l'area di un appartamento il numero di stanze e la grandezza delle stanze predire il prezzo di una stanza. Di queste si notano:

- **Errore medio assoluto:** è la media delle differenze assolute tra il valore (classe, o valore numerico) reale e quello predetto. È una metrica applicabile anche ai classificatori, ma usualmente viene usata per i regressori. Il suo valore è tra  $[0, \infty]$ .

$$\frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

- **Errore quadratico medio:** è la media delle differenze al quadrato tra il valore reale e quello predetto. Il suo valore è tra  $[0, \infty]$ .

$$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

- **Classificazione binaria:** algoritmo di previsione in cui l'output può essere uno dei due elementi indicati con 0 o 1.
- **Classificazione multi-classe :** si occupa dei problemi in cui il risultato può essere più di una classe. È l'estensione del problema di classificazione binaria.

#### 2.3.4 Ottimizzatori

Gli ottimizzatori sono algoritmi (o metodi) utilizzati per modificare gli attributi della rete neurale, come i pesi e la velocità di apprendimento, al fine di ridurre le perdite. Uno degli algoritmi più utilizzati per l'ottimizzazione delle reti neurali è Gradient Descent il quale ha come obiettivo ridurre al minimo la loss function della rete neurale. Si utilizzano principalmente 3 varianti di questo metodo:

**BATCH GRADIENT DESCENT** Tutti i dati del training set vengono presi in considerazione per eseguire il passo successivo. Si prende la media dei gradienti di tutti i dati del training set e si utilizza questo valore per aggiornare i parametri.

**STOCHASTIC GRADIENT DESCENT** Cerca di aggiornare i parametri del modello più frequentemente. In questo caso, i parametri del modello vengono modificati dopo il calcolo della loss su ciascun esempio di addestramento.

**MINI-BATCH GRADIENT DESCENT** È un miglioramento di SGD e BGD. Aggiorna i parametri del modello dopo ogni batch. Quindi, il set di dati è diviso in vari batch e dopo ogni batch, i parametri vengono aggiornati.

### 2.3.5 *Frameworks*

I framework più utilizzati sono:

**TENSORFLOW** Un framework open source end-to-end per il Machine Learning. Ha un ecosistema flessibile e completo che consente di creare e distribuire facilmente applicazioni basate sul ML [10].

**PYTORCH** Un framework open source per il Machine Learning che accelera il percorso dalla prototipazione della ricerca alla distribuzione in produzione [11].

**KERAS** È un'API progettata per gli esseri umani, non per le macchine. Keras segue le migliori pratiche per ridurre il carico cognitivo: offre API coerenti e semplici, riduce al minimo il numero di azioni dell'utente richieste per i casi d'uso comuni e fornisce messaggi di errore chiari e utilizzabili [12].

### 2.3.6 *Reti Neurali Convoluzionali*

Chiamate anche convolutional neural networks, si distinguono dalle altre reti neurali per le loro prestazioni superiori con la classificazione di immagini, video e il riconoscimento di testo e audio. I layer (strati) più importanti sono:

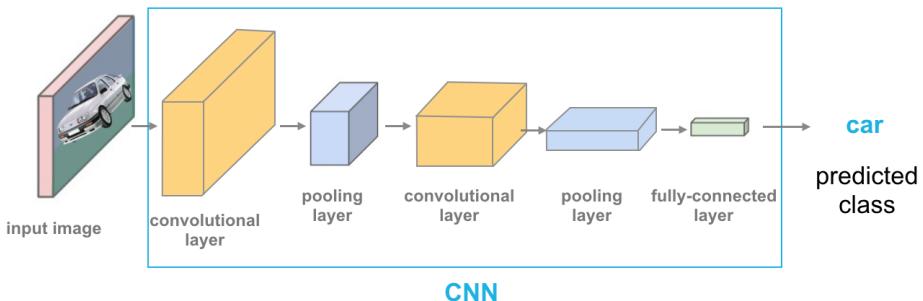


Figura 6.: Schema base di una CNN[14]

- **Convolutional layer**
- **Pooling layer**
- **Fully-connected layer**

Lo strato convoluzionale (o convolutional layer) è il primo strato di una CNN. In seguito, si ha che questi strati possono essere seguiti da strati convoluzionali aggiuntivi o strati di pooling, ma lo strato finale è il Fully-Connected layer (vedi Figura 6). Supponendo di analizzare un'immagine si ha che, con ogni strato la rete aumenta nella sua complessità, identificando porzioni maggiori dell'immagine. I livelli iniziali si concentrano su caratteristiche semplici, come colori e bordi e, man mano che i dati dell'immagine avanzano attraverso la rete, i livelli successivi riconoscono elementi e forme più grandi fino a quando, infine, sono in grado di riconoscere l'oggetto vero e proprio [4].

**CONVOLUTIONAL LAYER** È l'elemento fondamentale di una CNN ed è il luogo in cui avviene la maggior parte dei calcoli. Può essere considerato come l'estrattore di funzionalità di questa rete, impara a trovare caratteristiche spaziali di un dato in input. Questo livello viene prodotto applicando una serie di filtri diversi, noti anche come kernel convoluzionali. Questi filtri sono griglie di valori molto piccole che scorrono su un'immagine, pixel per pixel, e producono un'immagine di output filtrata con all'incirca le stesse dimensioni dell'immagine di input. Più kernel produrranno più immagini di output filtrate [4, 13].

**POOLING LAYER** Noto anche come downsampling, effettua una riduzione di dimensionalità, riducendo quindi il numero di parametri in input. Similmente al livello convoluzionale, l'operazione di polling applica un filtro sull'intero input, ma a differenza del precedente questo

filtro non ha pesi. In questo caso, il kernel applica una funzione di aggregazione ai valori all'interno del campo ricettivo, popolando l'array di output. Esistono due tipi principali di pooling:

- **Max pooling:** quando il filtro si sposta sull'input, seleziona il pixel con il valore massimo da inviare all'array di output. Questo approccio tende ad essere utilizzato più spesso rispetto al average polling.
- **Average polling:** quando il filtro si sposta sull'input, calcola il valore medio all'interno del campo ricettivo da inviare all'array di output.

Sebbene molte informazioni vengano perse nel livello di pooling, ha anche una serie di vantaggi per la CNN, ovvero aiuta a ridurre la complessità, migliorare l'efficienza e limitare il rischio di overfitting<sup>2</sup> [4, 13].

**FULLY-CONNECTED LAYER** È il layer presente alla fine di una rete neurale convoluzionale. Fully-Connected (completamente connesso) significa che ogni output prodotto alla fine del livello precedente è un input per ogni nodo in questo livello, quindi il ruolo di questo livello è quello di produrre un elenco di probabilità per ogni classe, di conseguenza ha tanti nodi quante sono le classi [14].

#### 2.4 DEEP LEARNING

Il Deep Learning (DL) è un sottoinsieme del Machine Learning, che si può definire essenzialmente come una rete neurale con tre o più livelli. Queste reti neurali tentano di simulare il comportamento del cervello umano e, sebbene lungi dall'essere all'altezza delle sue capacità, permettono di apprendere da grandi quantità di dati. Sebbene una rete neurale con un singolo livello possa ancora fare previsioni approssimative, gli hidden layers possono aiutare a ottimizzare e migliorare la precisione[5]. Questo utilizza reti neurali artificiali a più livelli per fornire un'accuratezza all'avanguardia in attività come il rilevamento di oggetti, il riconoscimento vocale, la traduzione e altre. Questo metodo differisce dalle tradizionali tecniche di Machine Learning in quanto può apprendere automaticamente da dati come immagini, video o testo, senza essere programmato per farlo e inoltre, grazie ad una architettura altamente flessibile, può apprendere direttamente dai dati grezzi aumentando la

---

<sup>2</sup> Caso in cui il modello si adatta troppo bene ai dati di training

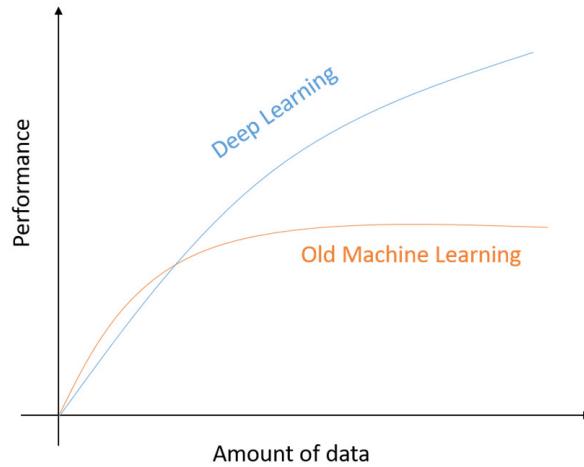


Figura 7.: Differenza tra Deep Learning e i vecchi algoritmi di Machine Learning

sua accuratezza predittiva al crescere della quantità di dati forniti (vedi Figura 7).

## 2.5 IMPORTANZA DELL'EXPLAINABLE AI

Nonostante l'enorme utilizzo in vari settori, molti algoritmi di Machine Learning sono imperscrutabili, ecco perché vengono anche definiti black box, il che, soprattutto in contesti in cui le conseguenze sono significative, può portare a gravi problemi [16]. Questa "oscurità", per esempio, non ha permesso l'identificazione di bias algoritmici, derivanti da processi o dati incorretti, con pregiudizi verso determinati gruppi. Questi hanno portato a discriminazioni su larga scala in una serie di settori come assunzioni, promozioni, giustizia penale e assistenza sanitaria, il che ha reso, nel mondo delle industrie e della ricerca, questo argomento molto importante. Questi sono alcuni dei motivi per cui, nell'aprile 2017, DARPA ha fondato "Explainable AI (XAI) program" con l'obbiettivo di migliorare le spiegazioni dei modelli di AI [18], e sono state prese misure preventive come il "General Data Protection Regulation" (GDPR) nel quale l'unione europea garantisce ai cittadini il "diritto alla spiegazione" se questi sono affetti da una decisione presa da un algoritmo [19]

### 2.5.1 *Perché se ne ha bisogno*

Sono molte le casistiche in cui esiste la necessità di una intelligenza artificiale interpretabile, nel seguito ne descriviamo alcune.

**L'EXPLAINABLE AI È IMPORTANTE PER LE PERSONE CHE UTILIZZANO UNO STRUMENTO** Quando uno strumento di AI prende una decisione la persona che sta utilizzando questo strumento dovrebbe capire cosa c'è dietro questa decisione, un famoso esempio esplicativo è descritto in Pneumonia - Asthma [21], in cui, un sistema di Intelligenza Artificiale che era stato addestrato per prevedere il rischio di polmonite di una persona, è arrivato a conclusioni totalmente sbagliate. A partire dai dati reali il modello aveva appreso che i pazienti asmatici con problemi cardiaci, hanno un rischio inferiore di morire di polmonite rispetto alle persone sane. Questo però non può essere vero, poiché l'asma è un fattore che influisce negativamente sul recupero. I dati relativi all'addestramento erano sistematicamente distorti, perché, a differenza delle persone sane, la maggioranza di questi pazienti asmatici era sotto stretta supervisione medica. Quindi questo gruppo aveva un rischio significativamente inferiore di morire di polmonite.

**L'EXPLAINABLE AI È IMPORTANTE PER LE PERSONE CHE SONO GIUDICATE DA UNO STRUMENTO** Le persone affette devono essere in grado di capire il motivo di una data decisione. Un caso interessante è avvenuto nelle scuole di Houston in Texas [22]. Queste, per valutare le prestazioni degli insegnanti, utilizzavano un algoritmo di Intelligenza Artificiale, denominato Educational Value-Added Assessment System (EVAAS), questo sistema, in seguito, è stato contestato con successo dagli insegnanti in tribunale, in quanto non è stato possibile spiegare le motivazioni delle recensioni negative restituite dal sistema di AI.

**L'EXPLAINABLE AI PUÒ AIUTARE GLI SVILUPPATORI A MIGLIORARE GLI ALGORITMI DI AI** Rilevando bias dei dati, identificando errori nei modelli e rimediando alle debolezze. Un interessante esempio è stato osservato nel paragone tra una rete Deep Learning e il Fisher Vector [23]. Si è notato che durante la previsione di immagini contenenti un cavallo, nonostante il secondo abbia solitamente una precisione molto inferiore, le prestazioni erano simili. Analizzando i modelli con metodi interpretabili si è notato che il primo si basava sulla effettiva forma del cavallo, il secondo invece si basava su un watermark di copyright spesso presente

nelle immagini rappresentanti dei cavalli. Eliminando il watermark la precisione del Fisher Vector diminuì significativamente.

#### ALTRI ESEMPI POSSONO ESSERE

- Campo economico: sapere il motivo perché un finanziamento non è stato approvato, il che può permettere all'utente di capire cosa migliorare.
- Campo legale: decidere se una persona è sospetta oppure no, e sapere anche il perché della decisione nel caso l'algoritmo abbia dei pregiudizi verso specifiche classi di persone.
- Auto a guida autonoma: situazioni di incidenti inevitabili sapere perché una determinata scelta viene effettuata invece di un'altra.

#### 2.5.2 *AI Fairness*

Le tecniche XAI possono essere utilizzate per rivelare se attributi come etnia, sesso, localizzazione e variabili socioeconomiche, sono utilizzati, direttamente o indirettamente, nei modelli black box, in modo che questi abbiano pregiudizi contro determinati gruppi. La ricerca sull'equità nel marketing AI può generare approfondimenti su come XAI possa essere integrata con lo sviluppo e l'implementazione di sistemi di intelligenza artificiale per prevenire e rilevare i bias algoritmici nelle applicazioni [16]. Alcuni dei recenti regolamenti a cui questi modelli devono sottostare sono:

- GDPR, General Data Protection Regulation (EU);
- SR 11-7, Supervisory Guidance on Model Risk Management;
- CCPA (dal 2020), California Consumer Privacy Act (Stati Uniti);
- ECOA Equal Credit Opportunity Act (Stati Uniti);
- FCRA Fair Credit Reporting Act (Stati Uniti).

# 3

---

## EXPLAINABLE AI

---

La black box dell'intelligenza artificiale è stata a lungo argomento di discussione per un motivo: La necessità di fiducia. Perché la macchina prende questa decisione? Su che base viene presa questa decisione? È molto scomodo, se non pericoloso, non conoscere queste risposte, se, per esempio, si stanno facendo importanti scommesse aziendali o si stanno per pubblicare applicativi di uso pubblico che si basano su decisioni di una macchina che non si conosce. È qui che ha origine la necessità di una Intelligenza Artificiale interpretabile.

### 3.1 COSA È

XAI è la classe di sistemi che fornisce chiarezza su decisioni ed azioni, prese da un sistema di AI (vedi Figura 8). Ha come obiettivo la spiegazione della logica del processo decisionale, fornire un'idea di come il sistema si comporterà in futuro e far emergere i punti di forza e di debolezza del modello.

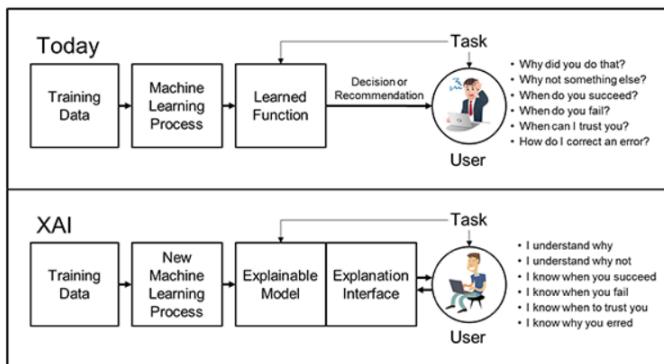


Figura 8.: Confronto di User Experience con e senza XAI [18]

Definire cos'è una spiegazione, è il punto di partenza per la creazione dei sistemi di XAI e consente di impostare i tre pilastri su cui si costruiscono le spiegazioni: obiettivi di una spiegazione, contenuti di una spiegazione e tipi di spiegazioni.

### 3.2 OBBIETTIVI

La necessità di sistemi spiegabili è definita principalmente in quattro punti[30]:

1. Verifica del sistema: capire le regole che governano il processo decisionale al fine di rilevare possibili errori.
2. Miglioramento del sistema: comprendere il modello e il set di dati per confrontare modelli diversi ed evitare guasti.
3. Imparare dal sistema: "Estrarre la conoscenza interna dal sistema AI", ovvero caratteristiche non visibili o non facilmente identificabili.
4. Conformità con la legislazione (in particolare con il "Diritto alla spiegazione" stabilito dall'Unione Europea): per trovare risposte a questioni legali e per informare le persone interessate dalle decisioni di questi sistemi.

### 3.3 CONTENUTI

Un modello di XAI dovrebbe rispondere a cinque domande:

1. Cosa ha fatto il modello?
2. Perché il modello ha eseguito P?
3. Perché il modello non ha eseguito X?
4. Cosa farebbe il modello se Y accadesse?
5. Come posso fare in modo che il modello esegua Z, dato il contesto attuale?

Alcuni contenuti di una spiegazione sono più interessanti o importanti per alcuni utenti rispetto ad altri. Ad esempio, ai ricercatori che sviluppano sistemi di Intelligenza Artificiale possono interessare spiegazioni

tecniche sul funzionamento al fine di migliorarli, mentre agli utenti finali, non avendo un background tecnico, questo tipo di informazioni possono risultare inutili. Si osserva che spesso si dà molta più importanza alle domande "perché"[24], e queste sono le domande su cui, in questa tesi, ci si concentra maggiormente.

### 3.4 TIPOLOGIE

Esistono fondamentalmente due approcci per ottenere l'interpretabilità dei modelli di Machine Learning:

- Utilizzare un modello trasparente (comprendibile senza ulteriori interventi).
- Utilizzare un modello black box e spiegare il suo comportamento tramite una tecnica post-hoc.

I modelli trasparenti sono argomento di grande ricerca, tuttavia, non sembrano essere pronti a sostituire i complessi modelli black box [17] ecco perché andremo ad analizzare solo le tecniche post-hoc.

#### 3.4.1 Interpretazioni post-hoc

In generale, i metodi per l'interpretabilità possono essere classificati come model-specific o model-agnostic (vedi Figura 9). In linea di principio, i metodi model-specific sono limitati a specifici tipi di modelli. Ad esempio, l'interpretazione dei pesi nella regressione lineare è un'interpretazione specifica del modello la quale non è applicabile ad altri. I metodi model-agnostic, invece, sono più generali e possono essere applicati a qualsiasi modello, il che facilita il confronto di diversi tipi di tecniche di interpretabilità ed elimina la necessità di sostituirle quando vengono cambiati i modelli, quindi questi metodi sono più flessibili ed utilizzabili [25].

**GLOBAL MODEL-SPECIFIC** Aumentano la comprensibilità dei modelli incorporando vincoli di interpretabilità nella struttura del modello. I vincoli strutturali possono includere: sparsity (dove vengono utilizzate meno caratteristiche come input) e monotonia (dove la relazione tra caratteristiche e previsioni è vincolata ad essere monotona). Alcuni esempi sono gli alberi decisionali o i modelli di regressione.

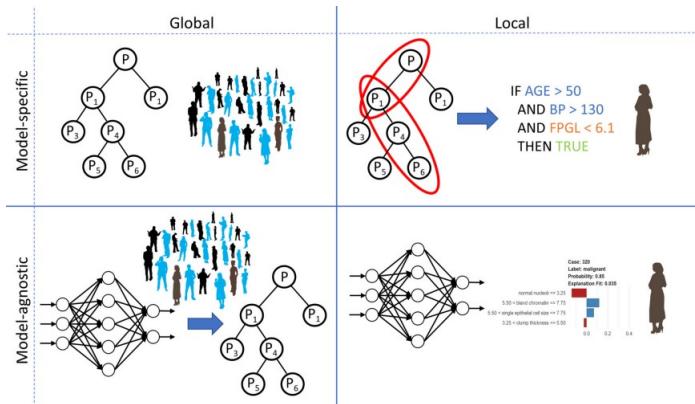


Figura 9.: Confronto delle varie interpretazioni post-hoc [26]

**LOCAL MODEL-SPECIFIC** Forniscono una spiegazione per un'istanza specifica in un modello. Un semplice esempio potrebbe essere quello di far comparire una pubblicità riguardante un brand quando il cliente è vicino ad un negozio del suddetto in un dato giorno o ad una certa ora. Alcuni esempi sono gli alberi decisionali (tramite decomposizione di alberi) e K-Nearest Neighbours.

**GLOBAL MODEL-AGNOSTIC** In questo caso il processo di spiegazione implica l'approssimazione di un modello black box con un modello interpretabile. Ad esempio, un modello di Deep Learning che calcola quanto la risposta dei clienti è influenzata da pubblicità ricevuta nel cellulare basata sulla posizione, che viene approssimato con un albero decisionale interpretabile.

**LOCAL MODEL-AGNOSTIC** L'obiettivo è generare spiegazioni indipendenti dal modello per un'istanza specifica. Alcuni esempi sono LIME e SHAP.

# 4

---

## TECNICHE

---

In questa sezione si mostrano alcune delle tecniche più utilizzate per l'interpretazione di modelli di Machine Learning.

### 4.1 TECNICHE INTERPRETABILI

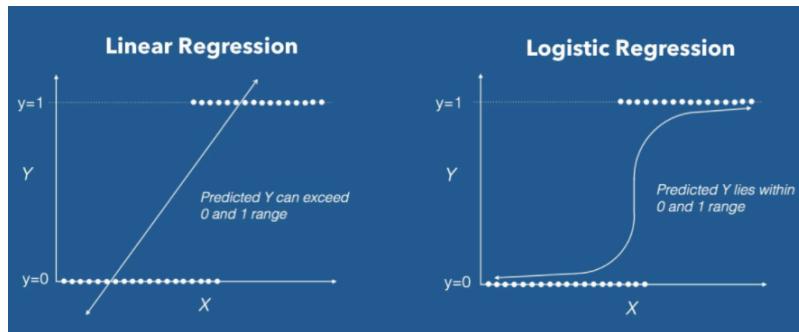
Non tutte le tecniche di Machine Learning sono delle black box impenerabili, alcune di queste sono molto più interpretabili delle reti neurali ed in seguito se ne descrivono alcune.

#### 4.1.1 *Regressione Lineare*

La regressione lineare viene utilizzata per prevedere il valore di una variabile in base al valore di un'altra variabile. La variabile che si desidera prevedere viene chiamata variabile dipendente. La variabile che si utilizza per prevedere il valore dell'altra variabile si chiama variabile indipendente. Un modello di regressione lineare cerca di adattare una linea di regressione ai punti dati che rappresentano al meglio le relazioni o le correlazioni.

#### 4.1.2 *Regressione Logistica*

La regressione logistica è principalmente preferita per attività di classificazione binaria come la previsione del tasso di abbandono, il rilevamento di e-mail di spam e le previsioni dei clic sugli annunci. La parte centrale della regressione logistica è la funzione logistica (cioè sigmoidale). Questa prende un'equazione lineare come input per la funzione sigmoidale e utilizza le probabilità per eseguire un'attività di classificazione binaria. La regressione logistica restituisce valori di probabilità.



#### 4.1.3 Foresta Casuale

Una foresta casuale è un insieme di alberi decisionali combinati con una tecnica chiamata bagging. Nel bagging, gli alberi decisionali vengono utilizzati come estimatori paralleli. La combinazione di molti alberi decisionali in parallelo riduce notevolmente il rischio di overfitting e si traduce in un modello molto più accurato. Il successo di una foresta casuale dipende in gran parte dall'utilizzo di alberi decisionali non correlati, in quanto, se usassimo alberi uguali o molto simili, il risultato complessivo non sarebbe molto diverso dal risultato di un singolo albero decisionale.

#### 4.1.4 XGBoost

XGBoost è una modifica di GBDT (Gradient Boosted Decisional Tree) il quale è costruito combinando alberi decisionali con una tecnica chiamata boosting. Boosting significa combinare un algoritmo di apprendimento in serie per ottenere uno modello (spesso alberi) forte da molti modelli deboli collegati in sequenza. Ogni albero tenta di ridurre al minimo gli errori dell'albero precedente. XGboost è una variante di GBDT ottimizzata per il potenziamento del gradiente distribuito progettata per essere altamente efficiente, flessibile e portatile [31]. Una caratteristica di XGBoost è che richiede un'attenta regolazione degli iperparametri, infatti, le prestazioni dei modelli dipendono fortemente dalla selezione dei valori ottimali.

#### 4.1.5 Alberi Decisionali

Gli alberi decisionali partizionano i punti dati (cioè le righe) ponendo domande in modo iterativo. Le suddivisioni sono eseguite in modo da produrre il maggior guadagno di informazioni. Le divisioni che aumentano la “purezza” dei nodi sono più informative.

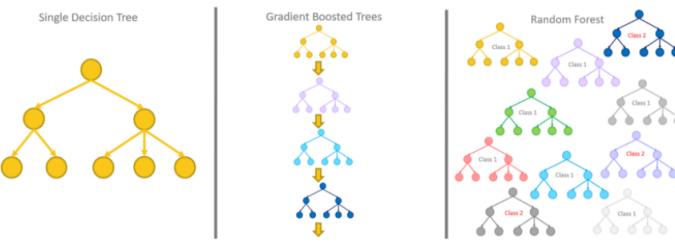


Figura 11.: Confronto tra alberi decisionali, GBDT e Foresta Casuale

#### 4.1.6 K-Nearest Neighbor

Utilizza la classificazione per stimare la probabilità che un punto (valore) sia membro di un gruppo o di un altro in base a quanti elementi di un gruppo siano vicini ad esso.

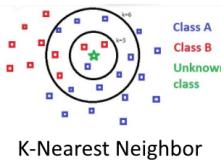


Figura 12.: K-Nearest Neighbor

## 4.2 TECNICHE MODEL-AGNOSTIC

Il grande vantaggio di queste tecniche, rispetto a quelle model-specific, è la loro flessibilità. Grazie a queste gli sviluppatori sono liberi di utilizzare il modello di Machine Learning che preferiscono, visto che l'interpretazione non dipende da quest'ultimo [39].

#### 4.2.1 LIME

LIME sta per Local Interpretable Model-agnostic Explanations [40] ed è un modello di tipo local model-agnostic. Il suo funzionamento è piuttosto intuitivo e si può descrivere in questo modo: si immagini di avere un modello black box in cui poter inserire dati, ottenere una previsione e di poter analizzare il modello quante volte se ne vuole con l'obbiettivo di capire il perché il modello ha preso una determinata scelta. LIME, verifica cosa accade alle previsioni quando si perturbano i dati in input, infatti, genera un nuovo set di dati costituito da dati perturbati e le rispettive previsioni. Su questo nuovo insieme, LIME addestra un modello interpretabile che viene ponderato in base alla vicinanza delle istanze campionate all'istanza di interesse [39].

##### *Immagini*

Non avrebbe molto senso perturbare singoli pixel, poiché sono molti i pixel che identificano un certo oggetto, di conseguenza, la modifica casuale di singoli pixel non cambierebbe di molto le previsioni. Pertanto, le variazioni delle immagini vengono create segmentando l'immagine in super-pixel e attivando o disattivando i super-pixel (vedi Figura 13 a). I super-pixel sono pixel interconnessi e possono essere disattivati sostituendo ogni pixel con un colore definito dall'utente [39].

##### *Testo*

In questo caso (vedi Figura 13 b) le variazioni dei dati vengono generate in modo diverso: a partire dal testo originale, vengono creati nuovi testi rimuovendo in modo casuale le parole dal testo originale. Il set di dati è rappresentato con caratteristiche binarie per ogni parola. Una caratteristica è 1 se la parola corrispondente è inclusa e 0 se è stata rimossa [39].

#### 4.2.2 Shapley values

Una previsione può essere spiegata supponendo che ogni valore dell'istanza sia un "giocatore" in un gioco in cui la previsione è il premio. I valori di Shapley sono basati sulla teoria dei giochi cooperativi (coalizione) e sono un metodo per distribuire equamente questo profitto tra i giocatori in base al loro contributo. Quindi questo valore rappresenta

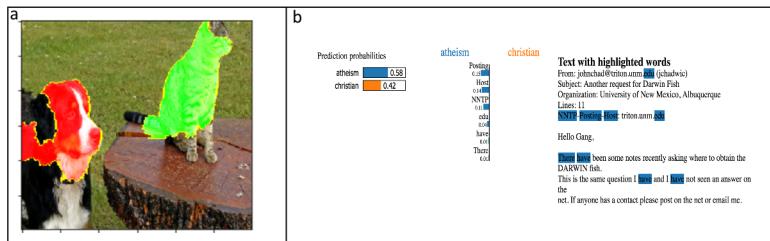


Figura 13.: (a) Esempio di LIME con immagini (b) Esempio di LIME con testo [33]

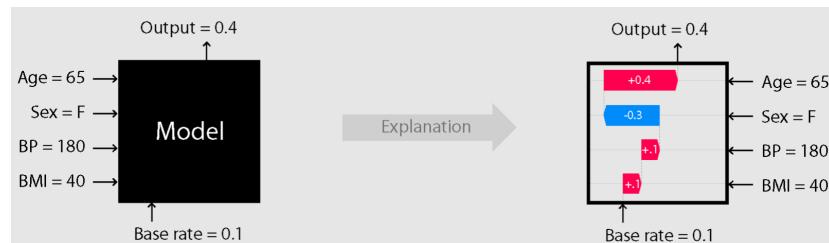


Figura 14.: Sintesi del funzionamento di SHAP[32]

il contributo marginale medio di una caratteristica in tutte le possibili coalizioni. Uno degli svantaggi è che i tempi di computazione crescono esponenzialmente con il numero delle caratteristiche.

#### 4.2.3 SHAP

SHAP sta per Shapley Additive exPlanations ed è un metodo di interpretazione che si basa sugli Shapley values di cui si è parlato in precedenza. L'obiettivo di SHAP è interpretare la previsione di qualsiasi istanza del modello di apprendimento automatico calcolando il contributo di ciascuna caratteristica per la previsione [32].

##### *Deep SHAP*

Deep SHAP combina i valori di Shapley e DeepLIFT. DeepLIFT linearizza i componenti non lineari di una rete neurale. Deep SHAP plasma DeepLIFT in modo da farlo diventare un'approssimazione dei valori Shapley.

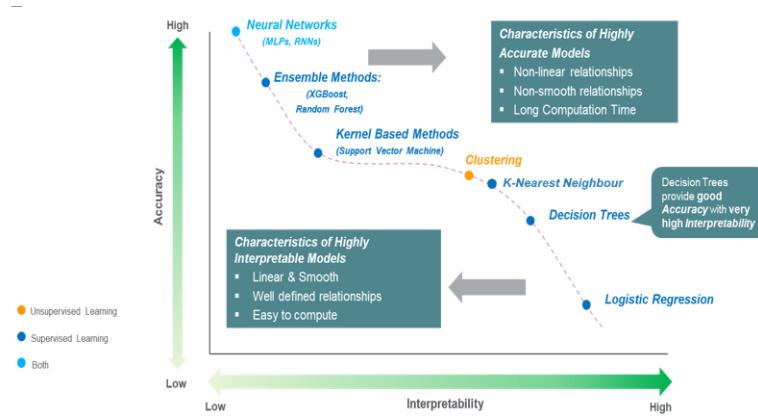


Figura 15.: Correlazione tra accuratezza e interpretabilità nei modelli di ML

### Kernel SHAP

Kernel SHAP combina due idee diverse: LIME e i valori Shapley. Kernel SHAP utilizza un modello di regressione lineare (modello LIME lineare) e un kernel di ponderazione appropriato per approssimare localmente la funzione  $f$ , quindi si ha che i coefficienti di regressione del modello LIME stimano i valori SHAP.

### Tree SHAP

Questa versione di SHAP mira a spiegare i modelli ML basati su alberi come foreste casuali, alberi decisionali e alberi gradient boost. TreeExplainer consente il calcolo esatto di spiegazioni locali ottimali per modelli basati su alberi, estende le spiegazioni locali per acquisire direttamente le caratteristiche e fornisce una nuova serie di strumenti per comprendere la struttura del modello globale sulla base di molte spiegazioni locali.

## 4.3 PRECISIONE VS INTERPRETABILITÀ

È più importante ottenere i migliori risultati o è meglio capire come sono stati prodotti questi risultati? Questa è una domanda a cui bisogna rispondere in ogni scenario di Deep Learning. Molte tecniche di Deep Learning sono di natura complessa e, sebbene risultino molto accurate in alcuni scenari, possono non sempre essere la scelta migliore.

Esiste una forte correlazione tra accuratezza e interpretabilità (vedi Figura 15). La Explainable AI mira alla creazione di un insieme di tecniche

capaci di produrre modelli più spiegabili pur mantenendo prestazioni di alto livello e consentendo, quindi, agli utenti di comprendere, fidarsi e collaborare efficientemente con gli emergenti sistemi di Intelligenza Artificiale.

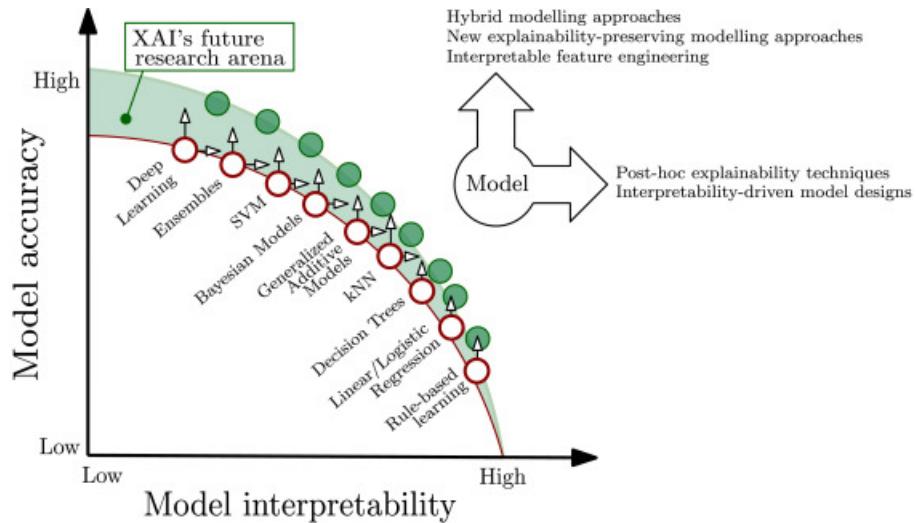


Figura 16.: Miglioramento delle tecniche tramite XAI

#### 4.4 CENNI RIGUARDO GLI ADVERSARIAL ATTACKS

Gli Adversarial Attacks sono dati manomessi sottoposti ad un modello di ML creati con l'obiettivo di fargli commettere un errore. Questi attacchi sono stati descritti formalmente per la prima volta nel 2004, quando i ricercatori hanno studiato le tecniche utilizzate dagli spammer per aggirare i filtri anti-spam [44]. In genere, questi, sono progettati prendendo dati reali e apportando modifiche mirate con l'obiettivo di ingannare l'algoritmo che li deve elaborare.

##### 4.4.1 *Classificazione*

In seguito una tassonomia che classifica gli attacchi contro i sistemi di apprendimento lungo tre assi:

###### INFLUENCE

- Causative attacks: influenzano l'apprendimento con il controllo sui dati di addestramento.

- Exploratory attacks: sfruttano le classificazioni errate ma non influiscono sull'addestramento.

#### SECURITY VIOLATION

- Integrity attacks: compromettono i dati tramite falsi negativi.
- Availability attacks: causano malfunzionamenti, di solito tramite falsi positivi.

#### SPECIFICITY

- Targeted attacks: si concentrano su un'istanza particolare.
- Indiscriminate attacks: agiscono su un'ampia classe di istanze.

#### 4.4.2 *Strategie*

In seguito le strategie più utilizzate:

**EVASION** Gli Evasion attacks sono il tipo di attacco più diffuso. In questo caso i campioni vengono modificati per evadere il rilevamento, così da venir classificati come legittimi anche se non lo sono. Ciò non comporta l'influenza sui dati di addestramento.

**POISONING** Il Poisoning ha come obiettivo la contaminazione del training set. I sistemi di apprendimento automatico possono essere riaddestrati utilizzando i dati raccolti durante le operazioni.

**MODEL EXTRACTION** Model extraction riguarda un attacco che ispeziona un modello black box con l'obiettivo di ricostruire il modello o estrarre i dati su cui è stato allenato.

# 5

---

## STRUMENTI

---

In questo capitolo si mostrano alcuni strumenti di cui si è valutato l'utilizzo, per poi definire quelli scelti per gli esperimenti del capitolo successivo.

### 5.1 APPLICATIVI XAI

Gli applicativi di Intelligenza Artificiale interpretabili sono strumenti che generano resoconti su come funziona il modello e cercano di spiegarne il funzionamento. In questa sezione, si analizzano alcuni dei più utilizzati.

#### 5.1.1 AIX360

AIX360 o AI Explainability 360 è un toolkit open source estensibile per aiutare a comprendere in che modo i modelli di Machine Learning predicono i risultati, è stato sviluppato dal gruppo di ricerca IBM. AIX360 contiene molti algoritmi con diversi scopi (vedi Tabella 1), alcuni si concentrano sulla spiegazione dei dati altri su quella del modello [36]. In seguito la lista completa degli algoritmi al suo interno.

- Data explanation
  - ProtoDash
  - Disentangled Inferred Prior VAE
- Local post-hoc explanation
  - ProtoDash
  - Contrastive Explanations Method
  - Contrastive Explanations Method with Monotonic Attribute Functions

Toolkit	Data Explanations	Directly Interpretable	Local Post-Hoc	Global Post-Hoc	Persona-Specific Explanations	Metrics
AIX360	✓	✓	✓	✓	✓	✓
Alibi			✓			
Skater		✓	✓	✓		
H2O		✓	✓	✓		
InterpretML		✓	✓	✓		
EthicalML-XAI				✓		
DALEX			✓	✓		
tf-explain			✓	✓		
iNNvestigate			✓			

Tabella 1.: Paragone di AIX360 e altre librerie[43]

- LIME
- SHAP
- Local direct explanation
  - Teaching AI to Explain its Decisions
- Global direct explanation
  - Boolean Decision Rules via Column Generation (Light Edition)
  - Generalized Linear Rule Models
- Global post-hoc explanation
  - ProfWeight
- Supported explainability metrics
  - Faithfulness
  - Monotonicity

### 5.1.2 ELI5

ELI5 è un pacchetto Python che aiuta a eseguire il debug dei classificatori di Machine Learning e a spiegare le loro previsioni. Supporta molti framework di ML come Scikit-learn, Keras, XGBoost, LightGBM e CatBoost [34]. Implementa anche diversi algoritmi per l’ispezione dei modelli black box tra cui anche LIME.

### 5.1.3 LIME

LIME utilizza le tecniche spiegate in 4.2.1. L’output di LIME è un elenco di spiegazioni, che riflette il contributo di ciascuna caratteristica alla previsione di una classe. LIME è in grado di spiegare qualsiasi classificatore

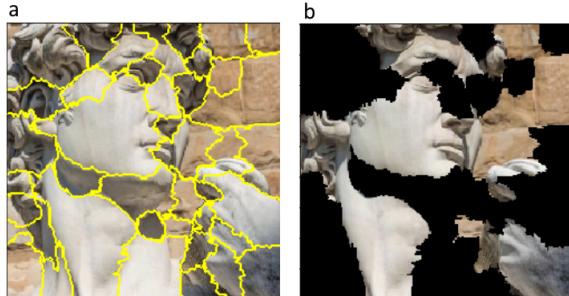


Figura 17.: (a)immagine segmentata (b) immagine perturbata

black box, con due o più classi. Tutto ciò che richiede è che il classificatore implementi una funzione che accetta testo grezzo o un array numpy e restituisca una probabilità per ogni classe.[33].

Analizzando il caso della classificazioni di immagini in generale i passaggi da effettuare sono:

1. Selezionare l'immagine in input;
2. Dividere l'immagine in super-pixels (vedi Figura 17 a);
3. Creare un nuovo dataset con immagini perturbate (vedi figura 17 b);
4. Valutare le immagini;
5. Allenare un modello interpretabile con i pesi esatti;
6. Selezionare le caratteristiche più significative, ovvero le immagini perturbate con le probabilità più alte;
7. Mostrare risultati.

Nonostante sia una delle migliori implementazioni e sia adatta a vari tipi di dati, due dei più grandi svantaggi sono: la sua inconsistenza, infatti se ripetute più volte le spiegazioni potrebbero essere diverse ed il fatto che durante ogni utilizzo bisogna provare diverse impostazioni per il kernel e vedere se queste producono una spiegazione “sensata”, il che può permettere agli utilizzatori di maneggiare i risultati per nascondere bias, rendendo LIME oggettivamente meno affidabile. Ecco perché è sempre consigliabile utilizzarlo con cautela [39].

### 5.1.4 *SHAP*

SHAP utilizza le tecniche spiegate in 4.2.3. Può essere utilizzato per spiegare vari tipi di modelli da semplici algoritmi di apprendimento automatico come regressione lineare e regressione logistica a modelli più complessi come i modelli di Deep Learning. È un metodo model-agnostic per spiegare i modelli basato sui valori Shapley della teoria dei giochi. Spiega come le diverse caratteristiche influenzano l'output e quale contributo hanno sul risultato finale [32]. All'interno del pacchetto si hanno le seguenti tecniche:

**DEEPEXPLAINER** È una implementazione di Deep SHAP (4.2.3) e viene utilizzato per calcolare i valori Shapley per i modelli di Deep Learning.

**GRADIENTEXPLAINER** Viene utilizzato per il calcolo approssimato dei valori Shapley nei diversi layers.

**KERNELEXPLAINER** È una implementazione di Kernel SHAP (4.2.3) ed è un metodo che utilizza una speciale regressione lineare ponderata per calcolare l'importanza di ogni caratteristica.

**LINEAREXPLAINER** Implementazione utilizzata per i modelli lineari.

**TREEEXPLAINER** È un'implementazione di Tree SHAP (4.2.3) e viene utilizzato per calcolare i valori Shapley per gli alberi.

Nonostante SHAP abbia una solida base concettuale nella teoria dei giochi e le previsioni siano distribuite uniformemente tra i valori caratteristici, uno dei suoi più grandi svantaggi è la lentezza, infatti richiede la computazione dei valori Shapley per numerose istanze [39].

### 5.1.5 *Skater*

Skater è un framework unificato per consentire l'interpretazione per tutti i tipi di modelli con l'obiettivo di aiutare a costruire un sistema di apprendimento automatico interpretabile. È una libreria Python open source progettata per interpretare le strutture apprese da un modello black box sia a livello globale che a livello locale [37].

### 5.1.6 *tf-explain*

*tf-explain* offre metodi di interpretabilità per Tensorflow 2.0 per facilitare il debug e la comprensione della rete neurale.[38]. Al suo interno ci sono i seguenti metodi:

- Activations Visualization
- Vanilla Gradients
- Gradients Input
- Occlusion Sensivity
- Grad-CAM
- SmoothGrad
- Integrated Gradients

Tutti i metodi elencati sono utilizzabili sia con un modello pre-trained (ovvero già addestrato) oppure, tramite callback per il l'addestramento di un nuovo modello. Grad-CAM viene spesso utilizzato per fini di debug in quanto permette di identificare esattamente le regioni in cui il modello sta identificando un oggetto[41], tramite questo è possibile identificare lo scorretto funzionamento di un modello. In un esperimento [41] tramite Grad-CAM si è identificato che un modello per identificare foto di dottori ed infermieri riportava risultati sbagliati (vedi Figura 18), infatti dando in input foto di dottori di sesso femminile il modello restituiva la classe degli infermieri, successivamente si è scoperto 78% delle immagini che contenevano dei dottori utilizzate per addestrare il modello erano di sesso maschile, un bilanciamento dei dati ha permesso il corretto funzionamento.

### 5.1.7 *What-If Tool*

What if Tool (WIT) è stato sviluppato da Google e fornisce un'interfaccia di facile utilizzo per espandere la comprensione dei modelli black box e degli algoritmi di regressione. Con il plugin, si possono eseguire inferenze su un ampio set di esempi e visualizzare immediatamente i risultati in vari modi. Inoltre, gli esempi possono essere modificati manualmente o in modo automatico e rieseguiti attraverso il modello per vedere i risultati delle modifiche. Contiene strumenti per analizzare le prestazioni

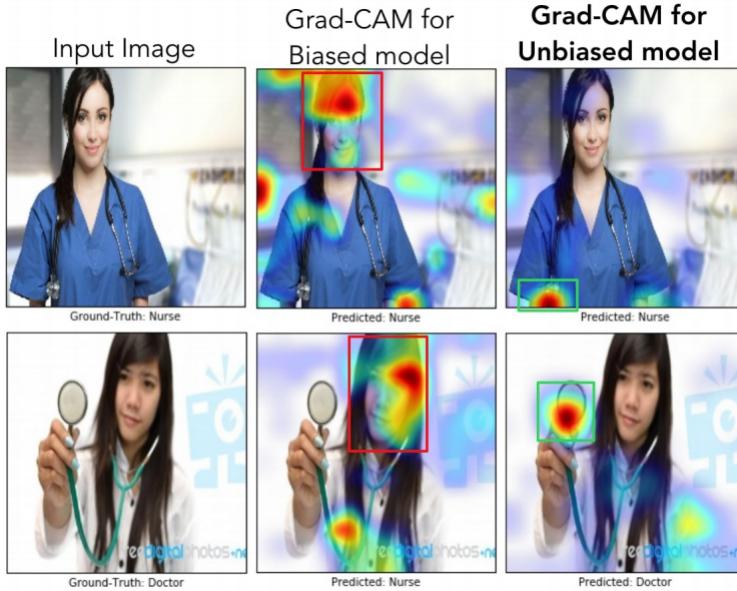


Figura 18.: Visualizzazione dei risultati di Grad-Cam[41]

e l'equità del modello su sottoinsiemi di un insieme di dati. Lo scopo dello strumento è fornire alle persone un modo semplice, intuitivo e potente per esplorare e analizzare i modelli di ML addestrati, attraverso un'interfaccia visiva e senza alcun codice richiesto[35].

## 5.2 APPLICATIVI ADVERSARIAL ATTACKS

In questa sezione si analizzano le librerie prese in considerazione per l'esecuzione degli Adversarial Attacks.

### 5.2.1 *Adversarial Robustness Toolbox*

Adversarial Robustness Toolbox (ART) è una libreria Python per il Machine Learning Security. ART fornisce strumenti che consentono a sviluppatori e ricercatori di difendere e valutare modelli di Machine Learning contro gli Adversarial Attacks di tipo evasion, poisoning, extraction e inference. ART supporta i più diffusi framework di Machine Learning (TensorFlow, Keras, PyTorch, MXNet, scikit-learn, XGBoost, LightGBM, CatBoost, GPy, ecc.), tutti i tipi di dati (immagini, tabelle, audio, video, ecc.) e funzioni di Machine Learning (classificazione, rilevamento di oggetti, riconoscimento vocale, generazione, certificazione, ecc.) [45].

### 5.2.2 *FoolBox*

Foolbox è una libreria Python che consente di eseguire facilmente Adversarial Attacks contro modelli di ML come le reti neurali. È basato su EagerPy e funziona in modo nativo con i modelli in PyTorch, TensorFlow e JAX [47].

### 5.2.3 *AdvBox*

Fa parte della Advbox Family che è una serie di strumenti di sicurezza di ML parte di Baidu Open Source, tra cui la generazione, la difesa e il rilevamento di Adversarial Attacks. Adversarialbox (AdvBox) è un insieme di strumenti per generare Adversarial Attacks che ingannano le reti neurali compatibile con PaddlePaddle, PyTorch, Caffe2, MxNet, Keras, TensorFlow. Advbox può confrontare la robustezza dei modelli di apprendimento automatico, ed inoltre, fornisce uno strumento da terminale per generare esempi contraddittori. È ispirato e basato su FoolBoxv1 [46].

## 5.3 LIBRERIE SCELTE

Nel prossimo capitolo si effettuano vari esperimenti analizzando diverse immagini. Per fare ciò si è deciso di utilizzare le seguenti librerie:

- LIME
- SHAP Gradient Explainer
- SHAP Kernel Explainer
- tf-explain (Grad-CAM)
- Adversarial Robustness Toolbox

Questa scelta è data dal fatto che LIME e SHAP sono le più adatte, ed utilizzate, per la classificazione di immagini, inoltre, è possibile comparare i risultati in quanto utilizzano entrambe una tecnica di spiegazione di tipo locale, inoltre, nonostante siano contenute in altre librerie (vedi Tabella 2) si è deciso di utilizzare le versioni originali. La scelta di tf-explain è data dal fatto che si è voluto inserire l'utilizzo di uno strumento di debug utilizzando una tecnica differente dalle altre ovvero Grad-CAM. Infine adversarial-robustness-toolbox è la libreria che si è ritenuta più funzionale per gli scopi della tesi riguardo gli Adversarial Attacks.

	AIX360	Skater	ELI5	InterpretML	tf-explain
LIME	X	X	X	X	
SHAP	X			X	

Tabella 2.: Implementazioni di LIME e SHAP in altre librerie

# 6

---

## TECNICHE UTILIZZATE

---

In questo capitolo ci si concentra sulla classificazione di immagini e le motivazioni che hanno portato i modelli a prendere una determinata scelta.

### 6.1 STRUMENTI UTILIZZATI

Si prendono come esempi cinque immagini (vedi Figura 19) e si utilizzano i seguenti modelli per la loro classificazione:

- VGG16, Convolutional Neural Network con 21 layers (di cui 16 pesati) e input di 224x224.
- VGG19, Convolutional Neural Network con 26 layers (di cui 19 pesati) e input di 224x224.
- MobileNet, Convolutional Neural Network con 90 layers (di cui 53 pesati) e input di 224x224.
- InceptionV3, Convolutional Neural Network con 311 layers (di cui 48 pesati) di 299x299.
- AkinolaVGG16, Convolutional Neural Network con 21 layers (di cui 16 pesati) e input di 224x224.

I primi quattro modelli sono ben conosciuti e tra i più utilizzati per classificare le immagini, tutti questi permettono la classificazione delle 1000 classi di oggetti di ImageNet<sup>1</sup>, l'ultimo modello è stato creato per fini esplicativi in quanto rappresenta il modello ideale per gli esperimenti.

---

<sup>1</sup> ImageNet è un ampio database di immagini utilizzato per il riconoscimento di oggetti nell'ambito dell'AI

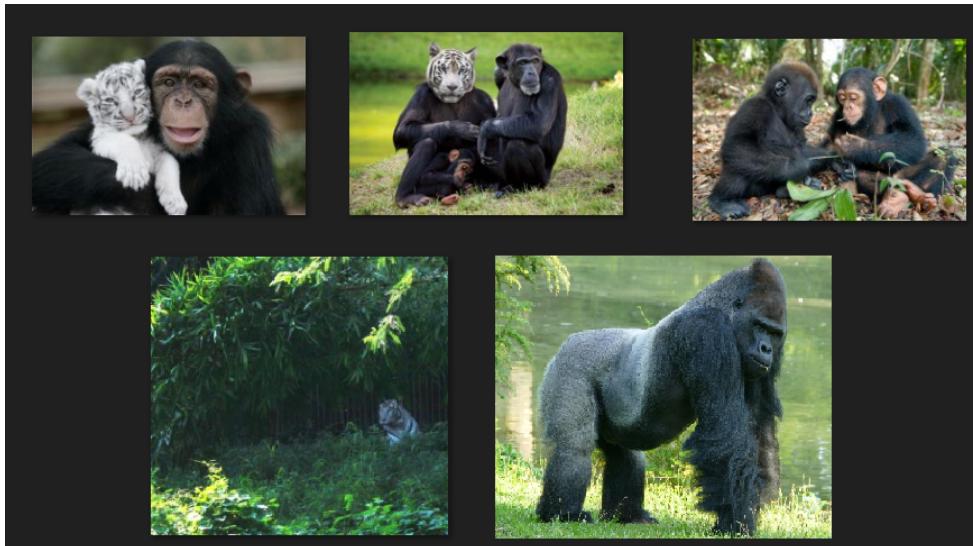


Figura 19.: Immagini utilizzate per gli esperimenti

Gli esperimenti si concentreranno più in specifico nel riconoscimento di 3 classi di immagini: Scimpanzé, Gorilla e Tigre bianca<sup>2</sup>. Per tutti gli esperimenti si utilizzano Tensorflow e Keras.

### 6.1.1 *AkinolaVGG16*

Questo modello si basa su VGG16, a differenza di questo però, può riconoscere solo 3 classi di oggetti. Per la sua costruzione si procede con l'importazione del modello VGG16 senza includere gli ultimi 3 layers in seguito si aggiungono 3 layers: un layer di Dropout per evitare overfitting, un layer Dense che definisce il numero di classi che il modello può riconoscere ed infine un layer Softmax.

---

```
#importazione del modello senza top
model = VGG16(
    input_shape=(224, 224, 3),
    include_top=False,
    pooling='avg' )

x = Dropout(rate=0.4)(model.output) #per evitare overfitting
x = Dense(3)(x)      #numero delle classi
x = Softmax()(x) model = Model(model.inputs, x)
```

---

<sup>2</sup> si specifica che ImageNet non ha una classe per la tigre bianca la quale viene spesso riconosciuta dalla classe snow\_leopard

Si sono inoltre resi tutti i livelli a parte gli ultimi 6 non addestrabili, il che ha consentito di addestrare un modello molto preciso in poco tempo e senza richiedere enormi risorse di calcolo. Dopo ciò, date 1078 immagini (circa 350 per ogni classe) prese in modo casuale da un famoso motore di ricerca, si applica un “augmentation” ai dati (in questo caso seleziona una porzione delle immagini e le capovolge orizzontalmente) per generare il training set e il validation set.

---

#### #generatore per il data augmentation

```
datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    preprocessing_function=_vgg16.preprocess_input,  
    horizontal_flip=True,  
    validation_split=0.2,  
)  
  
training = datagen.flow_from_directory(  
    'images',  
    target_size=(224, 224),  
    subset='training'  
)  
  
validation = datagen.flow_from_directory(  
    'images',  
    target_size=(224, 224),  
    subset='validation'  
)
```

---

Questi verranno utilizzati per allenare il modello. I risultati (vedi Figura 20) mostrano come 10 epoche siano più che sufficienti e permettano di ottenere una accuracy del 99%

### 6.1.2 Struttura

La struttura degli esperimenti è:

- Esperimenti Lime
- Esperimenti SHAP Gradient
- Esperimenti SHAP Kernel
- Esperimenti tf-explain

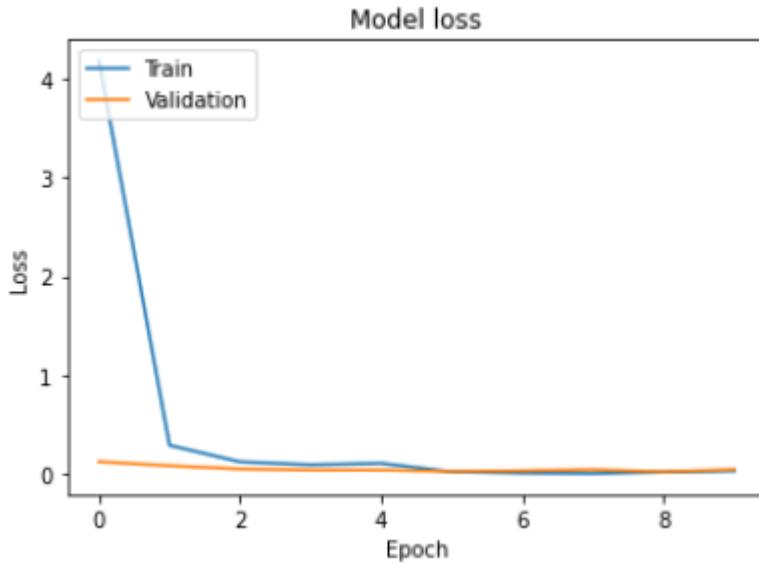


Figura 20.: Prestazioni del modello

- Esperimenti Adversarial Robustness Toolbox (ART)

Per i primi quattro esperimenti si analizza l'utilizzo con modelli diversi a seconda dei casi<sup>3</sup>, i risultati completi sono comunque visualizzabili in appendice A. Per l'ultimo esperimento si è preferito eseguire un solo esempio.

#### 6.1.3 Motivazioni

Nonostante esistano molti dataset pubblici con migliaia di immagini e perfezionati al dettaglio, per questi esperimenti si è scelto utilizzare normali immagini prese da internet per renderli più reali, per avere la possibilità di replicare ogni esperimento utilizzando una qualsiasi immagine ed infine si è preferito non fare qualcosa di "già visto". La scelta delle classi scimpanzé e tigre bianca è puramente casuale mentre la scelta della classe gorilla è data dal fatto che per i modelli analizzati scimpanzé e gorilla sono molto simili (si trovano, infatti, spesso nelle prime posizioni quando si cerca di riconoscere una delle due classi).

---

<sup>3</sup> le previsioni più precise, si ottengono sempre utilizzando il modello AkinolaVGG16

Tutti i codici presenti in questa tesi sono disponibili, in versione completa, online su GitHub<sup>4</sup> ed è possibile replicare gli esperimenti sia online tramite Google Colab<sup>5</sup> che in locale scaricando direttamente i Notebook presenti nella repository.

## 6.2 ESPERIMENTI

In questa sezione si analizzano i diversi esperimenti fatti. Per evitare ripetizioni si specifica che per tutti gli esperimenti fatti i passaggi iniziali sono:

- Importazione delle librerie richieste,
- Caricamento del modello utilizzato,
- Caricamento dell'immagine da analizzare,
- Utilizzo del modello per mostrare i risultati previsti.

Inoltre per le librerie bisogna provvedere all'installazione tramite il comando pip.

### 6.2.1 LIME

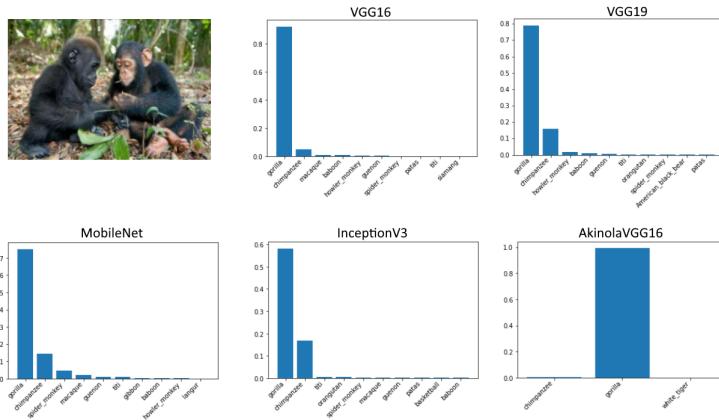


Figura 21.: Risultati predetti dai modelli

4 <https://github.com/SAAF-16/ExplainableAI-Bachelor-Thesis>

5 <https://colab.research.google.com/>

Per questo esperimento si utilizza LIME come spiegato in 5.1.3. Dopo aver predetto i risultati (vedi Figura 21) si prosegue con la segmentazione dell’immagine. Per fare ciò bisogna definire una funzione di segmentazione per segmentare appropriatamente l’immagine, questa fase non è uniforme, di conseguenza, per ogni immagine, si devono cambiare i valori dipendentemente dall’input analizzato. Successivamente si esegue la spiegazione dei risultati ottenuti, la cui durata dipende dalla quantità di risorse utilizzabili e il “num\_samples” (in questo caso 800 è un numero soddisfacente e non troppo lento da eseguire) ottenendo infine i risultati (vedi Listing 6.1).

---

Listing 6.1: Lime

---

```
# dati utilizzati
kernel_size=6
max_dist=10
ratio=0.0005

# genera i superpixel
superpixels = skimage.segmentation.quickshift(
    img,
    kernel_size=kernel_size,
    max_dist=max_dist,
    ratio=ratio
)

# genera funzione di segmentazione
seg_fn=SegmentationAlgorithm(
    'quickshift',
    kernel_size=kernel_size,
    max_dist=max_dist,
    ratio=ratio,
    random_seed=0
)

# imposta l'explainer
explainer = lime_image.LimeImageExplainer()

# esecuzione dell'explainer
explanation = explainer.explain_instance(
    img,
    class_fn,
    top_labels=4,
```

```

    hide_color=None,
    num_samples=800,
    segmentation_fn=seg_fn,
    num_features=num_superpixels
)

```

I risultati mostrano come tutti i modelli riescano a predire in maniera molto sicura la classe gorilla e come secondo risultato, con percentuali diverse, la classe scimpanzé. In seguito la spiegazione di LIME per i modelli VGG19 e MobileNet (vedi Figura 22).

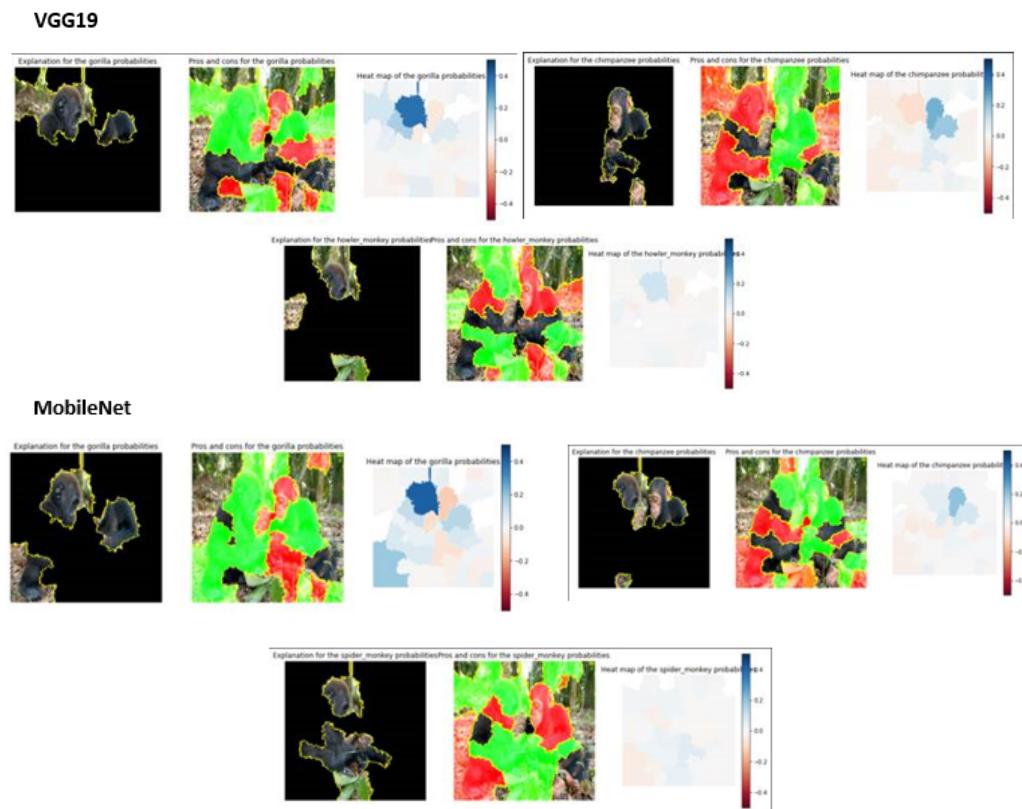


Figura 22.: Spiegazione di LIME per i modelli VGG19 e MobileNet

Come si può vedere LIME restituisce 3 tipologie di risultati ovvero i superpixel più importanti, una visualizzazione pro e contro e una heatmap che permette di vedere quanto è stata importante una determinata area dell'immagine analizzata. Ciò consente di capire dove i modelli identificano le loro previsioni.

- Per il primo risultato, ovvero il gorilla, si può vedere che l'output è molto simile, nonostante MobileNet sia leggermente più significativo, infatti, quando si esaminano anche heatmap e pro e contro si nota che identifica l'intero corpo del gorilla, anche se per essere corretti l'identificazione dell'intero corpo dice più riguardo all'affidabilità che alla precisione.
- Per il secondo risultato, ovvero lo scimpanzé si ha che VGG19 riconosce gran parte dello scimpanzé identificando la porzione di immagine che contiene il gorilla come parte negativa, Mobilenet invece, anche se (come si può vedere dalla heatmap) non pesantemente, identifica anche la parte superiore della testa del gorilla come punto di attivazione.
- Per il terzo risultato, ovvero spider\_monkey, i risultati sono più differenti anche se entrambi riconoscono la parte superiore della testa del gorilla come probabile zona.

### 6.2.2 SHAP Gradient Explainer

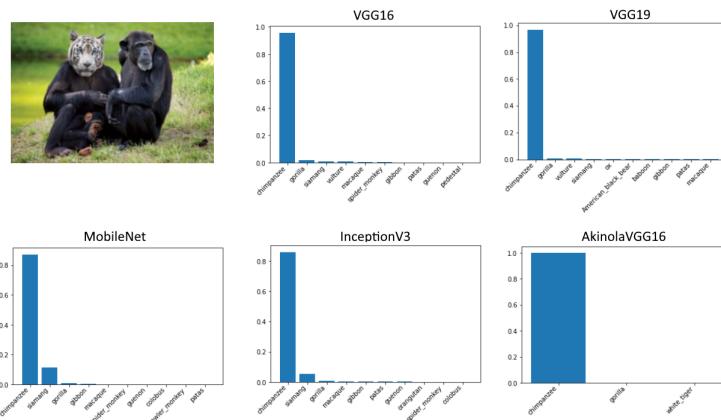


Figura 23.: Risultati predetti dai modelli

Per questo esperimento si utilizza SHAP Gradient Explainer come spiegato in 5.1.4. Dopo aver predetto i risultati (vedi Figura 23) si definisce il metodo per analizzare i diversi layer del modello utilizzato (vedi Listing 6.2), da notare come si è mantenuto il numero di samples uguale a 200 per non rendere l'esecuzione troppo lenta.

Listing 6.2: SHAP Gradient Explainer

---

```

# definisco il metodo shap_vals_layer che permette di visualizzare
# la spiegazione del layer n

def shap_vals_layer(n):
    # spiega come l'input al layer n del modello spiega le prime tre
    # classi
    def map2layer(x, layer):
        feed_dict = dict(zip([model.layers[0].input],
                            [preprocess_input(x.copy())]))
        return K.get_session().run(model.layers[layer].input, feed_dict)
    e = shap.GradientExplainer(
        (model.layers[n].input, model.layers[-1].output),
        map2layer(X, n), # X e' un dataset di 50 immagini che usiamo
        # per calcolare i valori Shapley
        local_smoothing=1
    )

    shap_values, indexes = e.shap_values(map2layer(to_explain,
                                                   n), nsamples=200, ranked_outputs=3)
    return (shap_values, indexes)

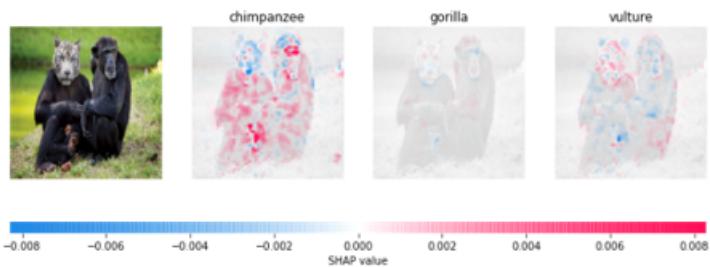
```

---

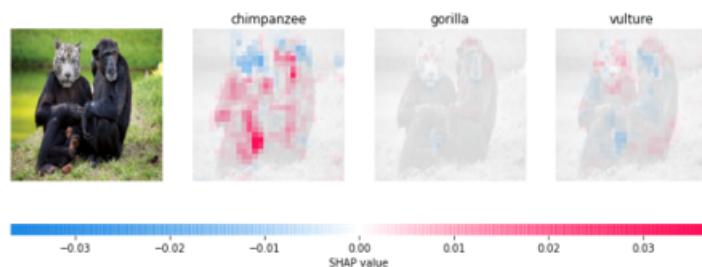
Anche in questo caso, nonostante questa sia un'immagine truccata, i modelli sono molto sicuri riguardo al primo risultato. Inoltre, a differenza di LIME, questa libreria consente di analizzare i singoli layer. In seguito la spiegazione di SHAP Gradient Explainer per i modelli VGG19 e AkinolaVGG16 (vedi Figure 24 e 25).

**VGG19**

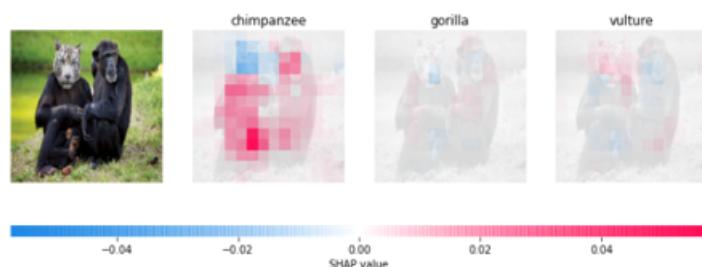
## Layer n.7



## Layer n.15



## Layer n.18



## Layer n.21

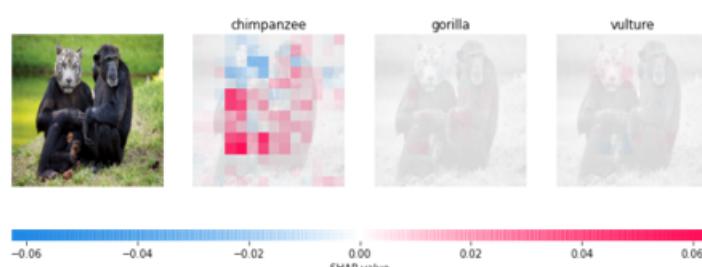


Figura 24.: Spiegazione di SHAP Gradient Explainer per i layer di VGG19

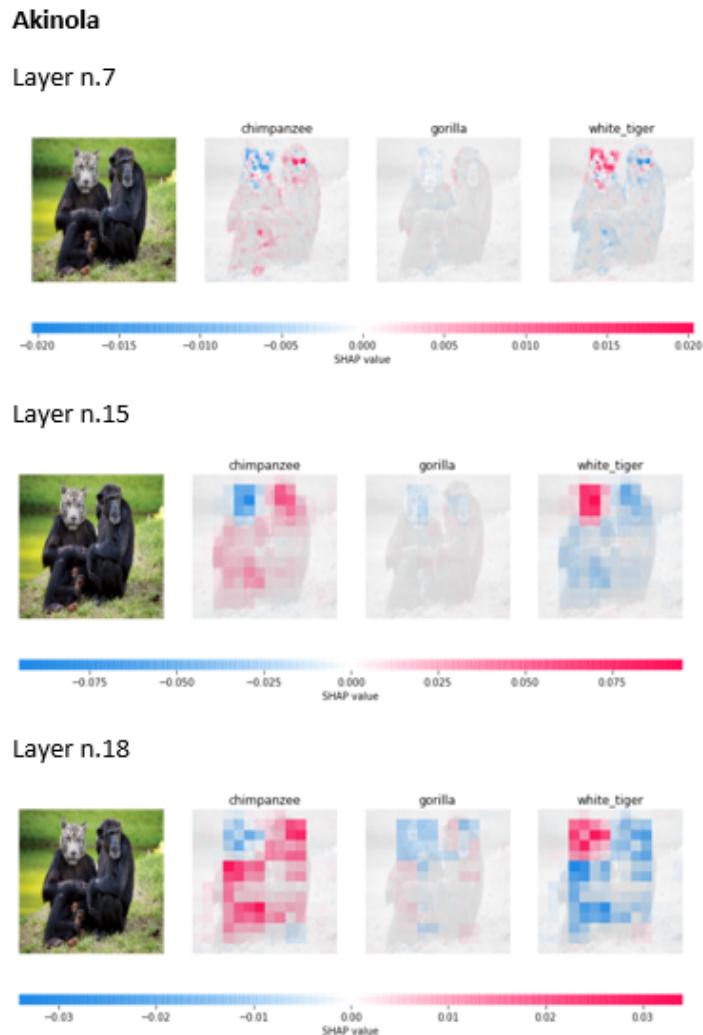


Figura 25.: Spiegazione di SHAP Gradient Explainer per i layer di Akinola-VGG16

Si può notare come questa libreria permetta di osservare come i layer convoluzionali procedono al riconoscimento, ovvero, riconoscendo nei primi livelli piccole porzioni delle immagini come bordi e linee fino a riconoscere nei livelli successivi porzioni più grandi dell'immagine, si osserva anche come, attraversando livelli di pooling, la dimensione del filtro cresca. Anche in questo caso i due modelli riescono ad identificare correttamente le zone appartenenti alla classe scimpanzé e le zone negative, inoltre si vede come nel modello AkinolaVGG16 è possibile

osservare la corretta delineazione delle zone positive e negative (con i valori opposti) anche per la classe white\_tiger.

### 6.2.3 SHAP Kernel Explainer

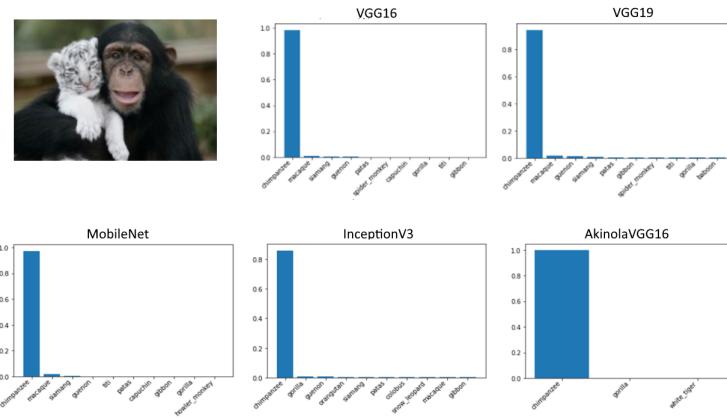


Figura 26.: Risultati predetti dai modelli

Per questo esperimento si utilizza SHAP Gradient Explainer come spiegato in 5.1.4. Dopo aver predetto i risultati (vedi Figura 26) si procede con la segmentazione dell’immagine (vedi Listing 6.3), anche in questo caso, come con LIME, a seconda dell’immagine bisogna modificare le variabili per definire la segmentazione, successivamente si definisce la funzione `model_fn` che viene utilizzata da SHAP kernel explainer, la quale dipende da una maschera binaria che rappresenta se una data regione dell’immagine è nascosta o meno, infine, si definisce ed esegue la spiegazione, in cui si è impostato il numero di samples uguale ad “auto” ovvero  $2 * \text{img.shape}[1] + 2048$ .

Listing 6.3: SHAP Kernel Explainer

---

```
#segmentazione

segments_slic = slic(img, n_segments=50, compactness=100, sigma=3)

def mask_image(zs, segmentation, image, background=None):
    if background is None:
```

```

background = image.mean((0,1))
out = np.zeros((zs.shape[0], image.shape[0], image.shape[1],
                image.shape[2]))
for i in range(zs.shape[0]):
    out[i,:,:,:] = image
    for j in range(zs.shape[1]):
        if zs[i,j] == 0:
            out[i][segmentation == j,:] = background
return out

def model_fn(z):

    return model.predict(preprocess_input((mask_image(z,
        segments_slic, img_arr))))


# utilizzo di Kernel SHAP per la spiegazione

explainer = shap.KernelExplainer(model_fn, np.zeros((1,100)))

shap_values = explainer.shap_values(np.ones((1,100)))

```

---

Anche in questo caso tutti i modelli hanno lo stesso primo risultato ovvero lo scimpanzé, in questa immagine c'è anche una tigre bianca, ovvero un oggetto facilmente riconoscibile e non simile allo scimpanzé, proseguiamo quindi con l'analisi dei risultati. In seguito l'interpretazione per i modelli InceptionV3 e Akinola VGG16 (vedi Figura 27).

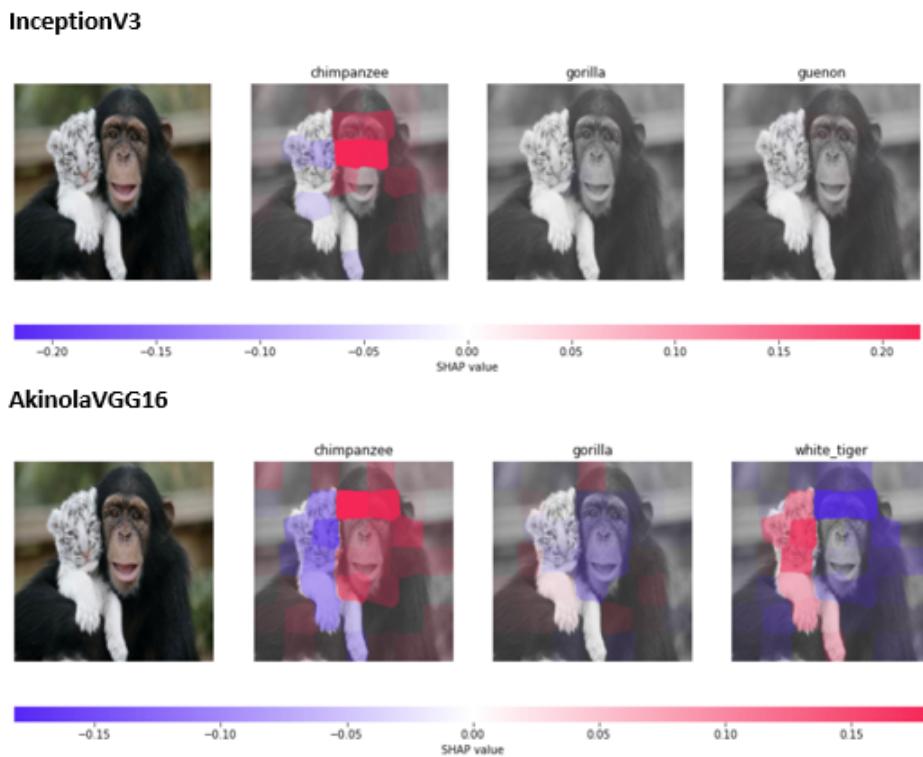


Figura 27.: Risultati di SHAP Kernel Explainer per i modelli InceptionV3 e AkinolaVGG16

I risultati mostrano come entrambi i modelli identifichino come area di maggiore importanza il viso dello scimpanzé anche se il primo modello dà più importanza alla parte centrale mentre il secondo si concentra di più nella parte superiore della testa, come spiegato precedentemente, la previsione viene spiegata con un punteggio in cui, in questo caso, le zone rosse danno un punteggio positivo mentre le zone blu un punteggio negativo. Ciò permette anche di analizzare con che sicurezza il modello analizzato sta funzionando, per esempio, si può vedere come InceptionV3 sia propenso ad identificare parte dell'area del viso e le zampe della tigre bianca come "non scimpanzé", per il modello AkinolaVGG16 si vede, invece, come la totale area della tigre bianca è identificata con sicurezza come "non scimpanzé", questa sicurezza spiega anche il posizionamento dei risultati per questo modello, dove infatti la tigre bianca risulta come terzo risultato dopo il gorilla in quanto, nonostante presen-

ti un'area di punteggi positivi, l'area negativa abbassa drasticamente il punteggio finale per questa previsione.

#### 6.2.4 *tf-explain*

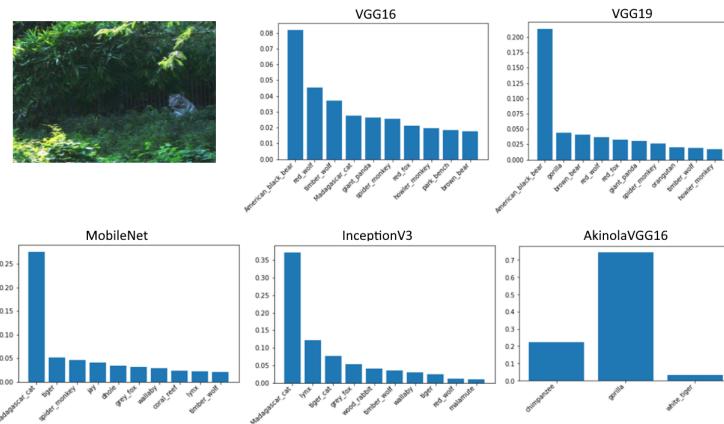


Figura 28.: Risultati predetti dai modelli

Per questo esperimento si utilizza tf-explain come spiegato in 5.1.6. Questo test è diverso dagli altri in quanto la motivazione del suo utilizzo è data dalla particolare immagine da analizzare, la quale ha messo in difficoltà tutti i modelli compreso il modello “ideale” AkinolaVGG16. Questo fatto ha portato a voler capire come mai neanche questo modello riuscisse a prevedere in modo corretto i risultati. Gran parte dei modelli restituisce risultati con percentuali molto basse e spiegazioni poco significative, il modello AkinolaVGG16, avendo meno scelta, ha una percentuale più alta, ecco perché si analizza solo questo modello. Questa libreria, a differenza delle altre, permette di effettuare un lavoro di debug. Inizialmente si potrebbe pensare che anche le precedenti librerie siano adatte a questo scopo, ma queste non lo sono per i seguenti motivi:

- LIME: come spiegato in precedenza non è consistente nelle sue interpretazioni, richiede la modifica della funzione di segmentazione per renderla utilizzabile con l'immagine analizzata e richiede un discreto tempo di esecuzione.
- SHAP Gradient Explainer: richiede grandi risorse di computazione e un elevata quantità di tempo per ogni esecuzione.

- SHAP Kernel Explainer: ha fondamentalmente i difetti dei due precedenti metodi.

A differenza delle precedenti, tf-explain consente di avere risultati immediati e consistenti ogni volta che lo si esegue permettendo inoltre di indagare i layer di interesse.

Dopo aver predetto i risultati (vedi Figura 28) si definisce ed esegue la spiegazione che, a differenza delle altre, non ha bisogno di tempi significativi per la computazione (vedi Listing 6.4).

---

Listing 6.4: tf-explain

---

```
#imposto GradCAM

explainer = GradCAM()

#inspezione il layer di interesse.
#"layer=None" prende il miglior layer a seconda del modello

layer=None

grid1 = explainer.explain(data,model,int(index[0]),
    layer_name=layer,colormap=2,use_guided_grads=1)
grid2 = explainer.explain(data,model,int(index[1]),
    layer_name=layer,colormap=2,use_guided_grads=1)
grid3 = explainer.explain(data,model,int(index[2]),
    layer_name=layer,colormap=2,use_guided_grads=1)
```

---

Osservando la spiegazione del modello (vedi Figura 29) si osserva che le aree di attivazione non combaciano con le previsioni del modello, infatti, per il gorilla e lo scimpanzé sono aree in cui non ce nulla, o meglio ci sono solo foglie e rami. Ricordando il modo in cui il training set di questo modello è stato generato, si è intuito che questo errore è stato causato da un dataset non corretto, di conseguenza il modello riconosce zone di foresta come se fossero scimpanzé o gorilla solo perché nel dataset la maggior parte delle immagini contenenti queste classi si trovava in questo genere di ambiente, e nonostante il modello riesca a vedere correttamente la tigre bianca, questa, risulta comunque solo come ultimo risultato.

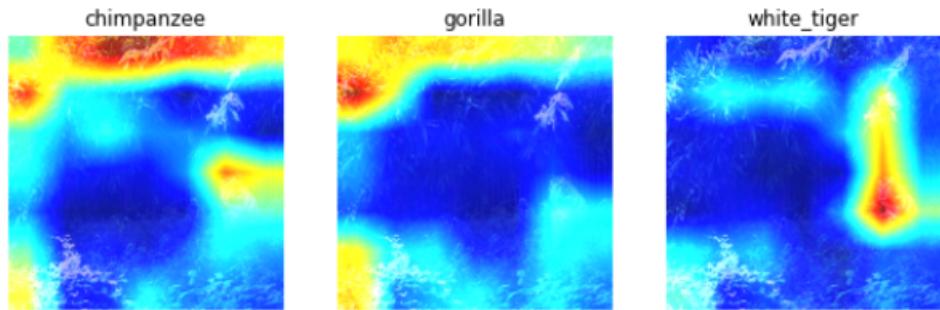


Figura 29.: Spiegazione di tf-explain per AkinolaVGG16

### 6.2.5 Adversarial Robustness Toolbox

In questo ultimo esperimento si utilizza adversarial robustness toolbox come spiegato in 5.2.1, eseguendo degli Adversarial Attacks di tipo evasion su una immagine per poi vedere come il modello reagisce, ovvero, cosa prevede e cosa lo ha portato a predire ciò. Visto che si ottengono risultati simili, per questo esperimento si utilizza solo il modello VGG16 e la libreria SHAP Gradient Explainer.

Si procede quindi con la definizione del classificatore, il quale viene definito con il modello utilizzato e clip\_values (che definisce il formato della immagine in input). Successivamente si definiscono gli attacchi che si vanno ad eseguire in cui eps/epsilon rappresenta la “forza” dell’attacco utilizzato ed infine si generano le immagini modificate in cui, nel secondo attacco, y=[289] rappresenta cosa si vuole che il modello identifichi, in questo caso 285 corrisponde alla classe ImageNet Egyptian\_cat (vedi Listing 6.5).

Listing 6.5: ART

---

```
#imposto il classificatore
classifier = KerasClassifier( model=model, clip_values=(0, 255))

#definisco gli attacchi
attack = DeepFool(classifier, epsilon=20)
attack2 = FastGradientMethod(classifier, eps=5, targeted=True)

#genero le immagini frutto degli attacchi
img_adv = attack.generate(img1)
img_adv2 = attack2.generate(img,y=[289])
```

---

Nel primo test (vedi Figura 30) si ha che la previsione per l'immagine originale è la classe gorilla la quale ha quasi il 100%, si può notare che la parte che ha influenzato di più la scelta è il viso del gorilla. Nel secondo test (vedi Figura 31) si utilizza l'immagine generata utilizzando DeepFool e si osserva come ora il primo risultato sia African\_elephant mentre la classe gorilla è scesa al quarto posto con meno del 0.005%, inoltre tramite si vede come le zone che il modello riconosce come elefante africano hanno poco o niente a che fare con esso e che parte delle zone riconosciute, nonostante siano riconosciute con sicurezza, non siano nemmeno parte del corpo del gorilla. Nel terzo test (vedi Figura 32) si utilizza l'immagine generata utilizzando Fast Gradient Method e si vede come ora il primo risultato sia Egyptian\_cat e che la classe gorilla non sia più nemmeno nei primi 10 risultati i quali sono tutti più comparabili ad un gatto egiziano che ad un gorilla, anche qui si osservano le aree che Fast Gradient Method ha manomesso, notando anche in questo caso come parte del riconoscimento sia nello sfondo. Si nota infine come in tutti e tre i casi l'immagine risulti visivamente identica.

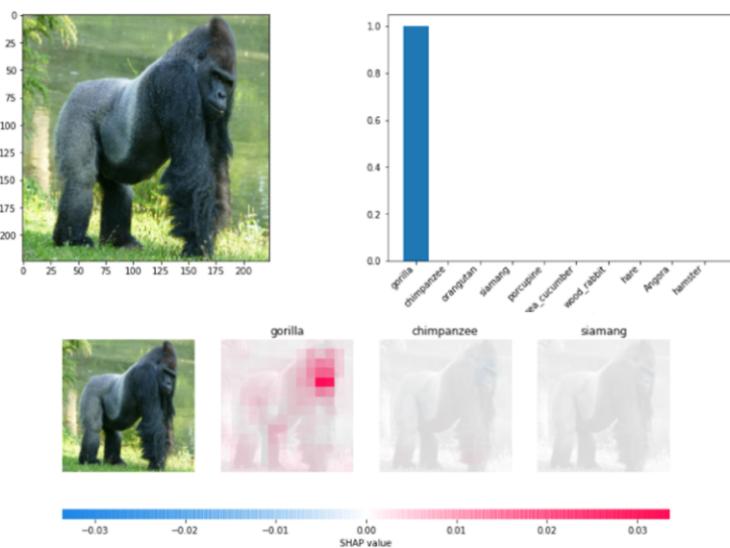


Figura 30.: Risultati ottenuti utilizzando l'immagine originale

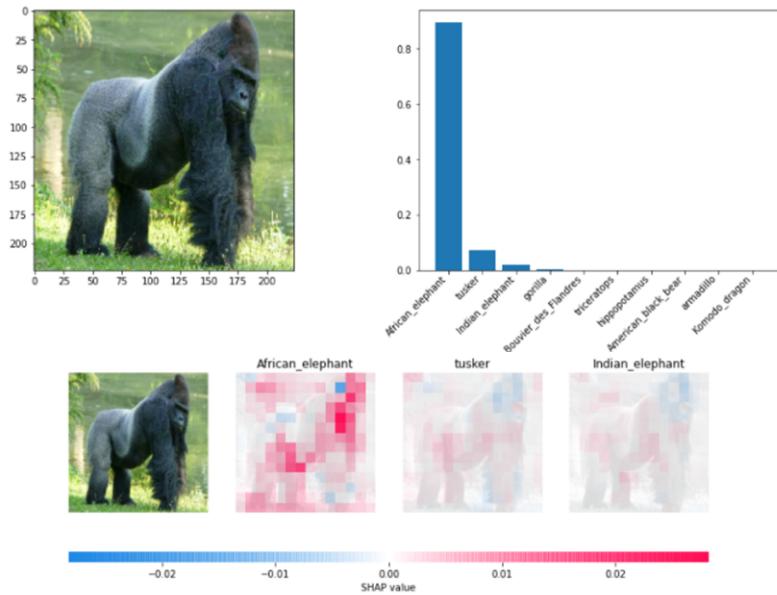


Figura 31.: Risultati ottenuti utilizzando l'immagine modificata da DeepFool

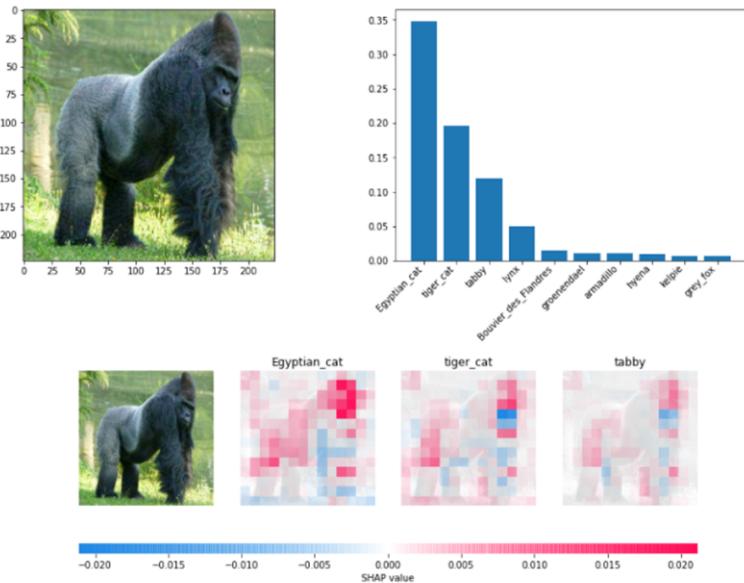


Figura 32.: Risultati ottenuti utilizzando l'immagine modificata da Fast Gradient Method



# 7

---

## CONCLUSIONE

---

In questa tesi si è parlato e analizzato l'argomento dell'Explainable AI e del perché sia importante avere la possibilità di interpretare le decisioni dei modelli black box.

Dopo aver trattato gli argomenti teorici, che sono il fulcro di interesse delle spiegazioni di XAI, si è descritto l'argomento di una Intelligenza Artificiale interpretabile e se n'è mostrata l'importanza ed i suoi principali obiettivi, in seguito si sono esposti diversi algoritmi interpretabili e si sono presentate alcune tecniche per l'interpretazione dei modelli di Deep Learning, successivamente si è proseguito mostrando alcuni strumenti per implementare le tecniche interpretabili negli esperimenti.

Si sono scelte LIME, SHAP Gradient Explainer, SHAP Kernel explainer tf-explain (GRAD-Cam) e ART le quali sono sembrate le più adatte per gli esperimenti. Tramite queste si è effettuata un'analisi dei modelli più utilizzati per la classificazione di immagini.

Per visualizzare risultati più significativi, si è provveduto alla creazione di un modello ideale per gli esperimenti, il quale ha permesso di vedere le librerie performare al meglio.

Le prime tre librerie hanno mostrato, con approcci diversi, come sia possibile giustificare le scelte prese dai modelli, tf-explain ha permesso di vedere come queste librerie non permettano solo di avere più fiducia nei modelli, ma siano anche validi alleati per rilevare errori ed ottenere modelli migliori ed infine tramite ART si è controllato come i modelli reagiscono ad attacchi esterni analizzando i punti di attivazione di queste immagini ingannevoli tramite SHAP.

Nonostante si siano ottenuti buoni risultati si sono anche notati i limiti di questi metodi che non sempre restituiscono risultati sensati, come si può vedere nei risultati in appendice A o, nel caso di LIME, pienamente affidabili. Questo fa capire come ci sia ancora molto lavoro da fare per renderli pienamente utilizzabili. Gli esperimenti hanno comunque mostrato come SHAP sia la libreria con le spiegazioni più affidabili.

Questi esperimenti hanno comunque mostrato come il domani in cui si può basare la fiducia su modelli più efficienti e soprattutto interpretabili sia sempre più vicino.

### 7.1 POSSIBILI SVILUPPI / LAVORI FUTURI

Il lavoro svolto si presta a sviluppi di vario genere. Due tra i più interessanti sono:

- L'utilizzo delle tecniche di Explainable AI per migliorare il riconoscimento di oggetti del "mondo reale", ovvero, il riconoscimento anche quando questi vengono ripresi da angoli insoliti o in posti non strettamente collegati ad essi. Degli studi [48] hanno mostrato come l'utilizzo del database ObjectNet, il quale comprende solo immagini di questo tipo, ha provocato una diminuzione dell'accuracy del riconoscimento dal 97% al 50-55% il che fa capire come in realtà il riconoscimento di oggetti sia ancora un problema difficile, di conseguenza c'è la necessità di algoritmi più intelligenti e ottimizzati. L'utilizzo di queste tecniche può rendere possibile la creazione di modelli che si focalizzano sull'oggetto da identificare isolandolo da tutto ciò che lo circonda, ed inoltre tramite il data augmentation si possono generare training set in grado di simulare angolature e prospettive insolite.
- L'utilizzo di queste tecniche non solo per capire ciò che la macchina sta facendo, ma anche per imparare dalla macchina stessa. I modelli di Intelligenza Artificiale hanno il vantaggio di potersi allenare innumerevoli volte su una grande quantità di dati il che gli permette di essere precisi nelle loro previsioni, un'interessante argomento di studio può essere l'utilizzo di queste capacità per analizzare argomenti da approfondire o non facilmente studiabili (un esempio potrebbe essere la classificazione di nei in dermatologia) per poi, tramite l'Explainable AI, ottenere una spiegazione e consentendo quindi di imparare da questi sistemi.

# A

---

## MATERIALE AGGIUNTIVO

---

### A.1 LIME

#### Risultati ottenuti utilizzando LIME

**Akinola**

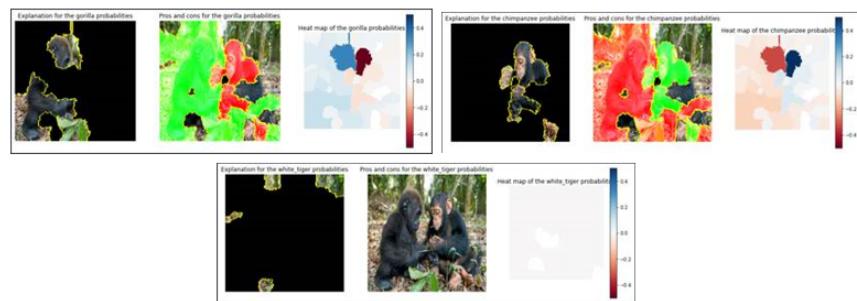


Figura 33.: Risultati ottenuti con AkinolaVGG16

**InceptionV3**

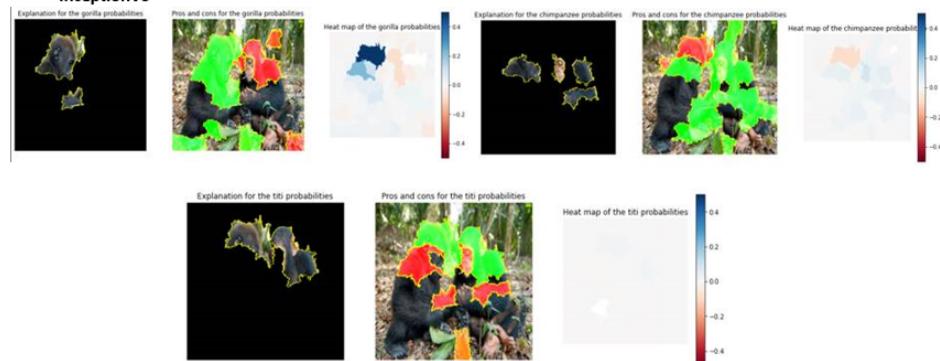


Figura 34.: Risultati ottenuti con InceptionV3

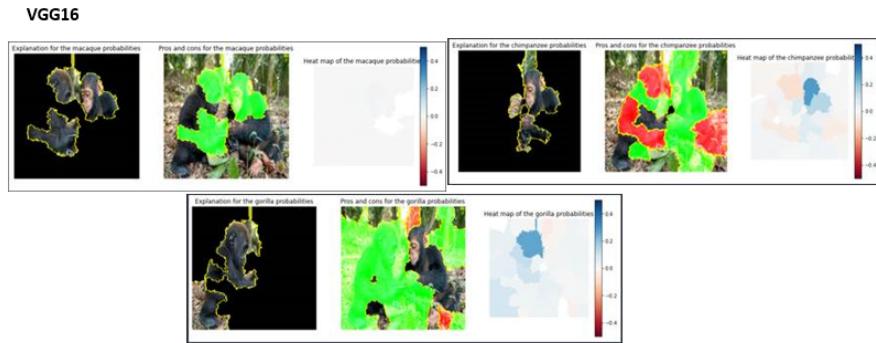


Figura 35.: Risultati ottenuti con VGG16

## A.2 SHAP GRADIENT EXPLAINER

Risultati ottenuti utilizzando SHAP Gradient Explainer

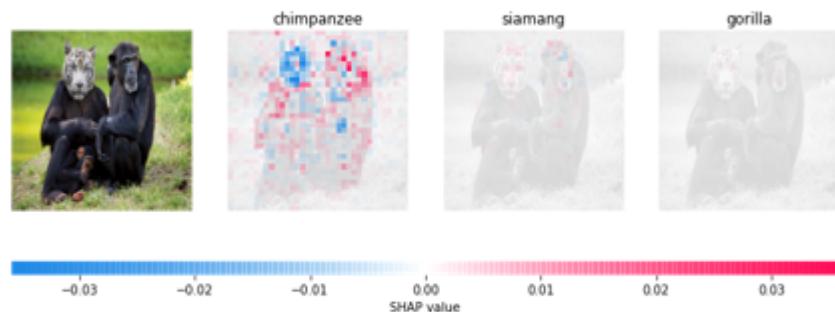
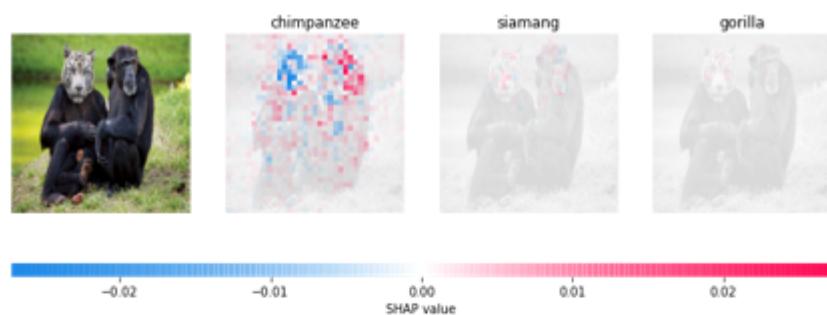
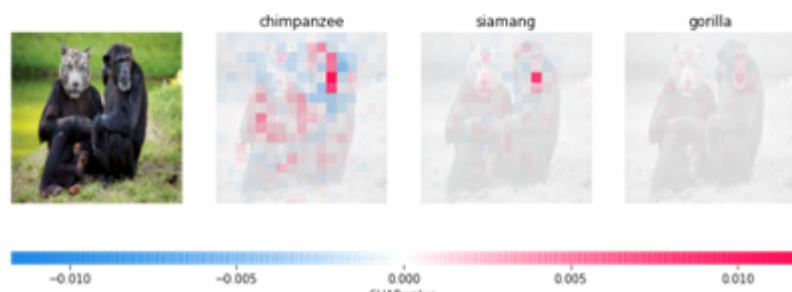
**Inception****Layer n.27****Layer n.50****Layer n.183**

Figura 36.: Risultati ottenuti con InceptionV3

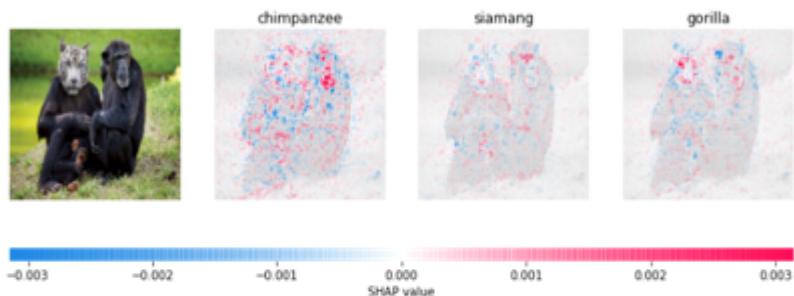
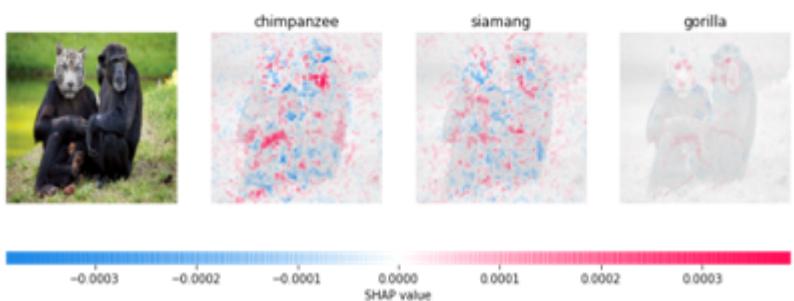
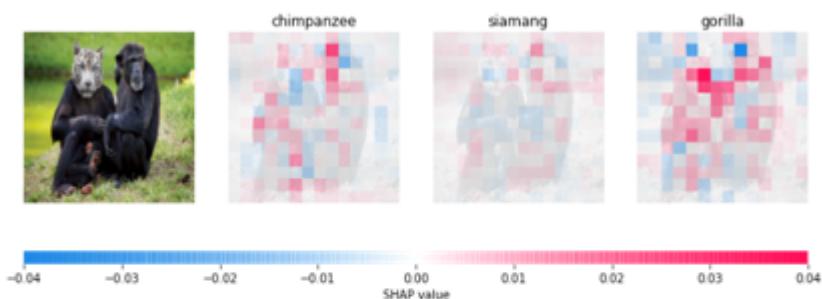
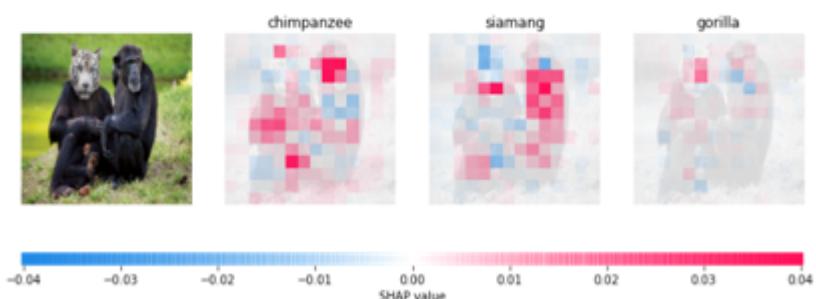
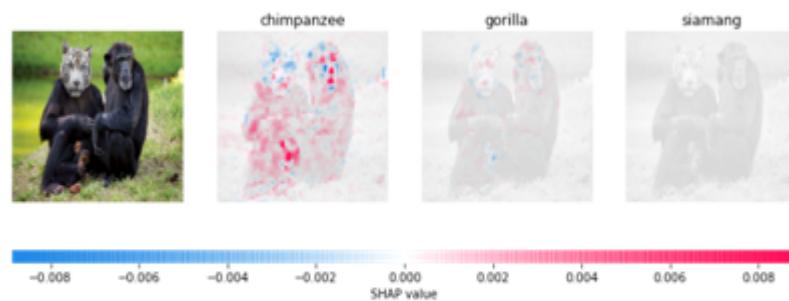
**Mobilenet****Layer n.7****Layer n.15****Layer n.40****Layer n.69**

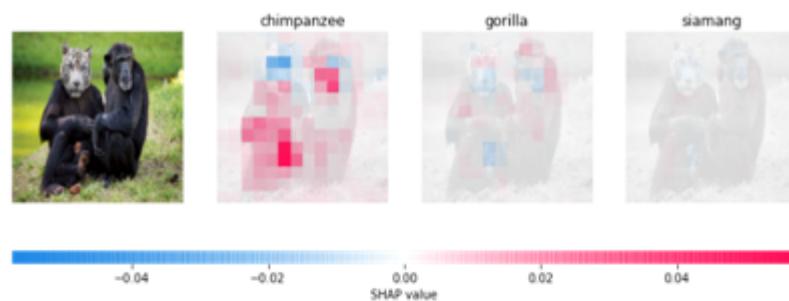
Figura 37.: Risultati ottenuti con MobileNet

**VGG16**

## Layer n.7



## Layer n.15



## Layer n.18

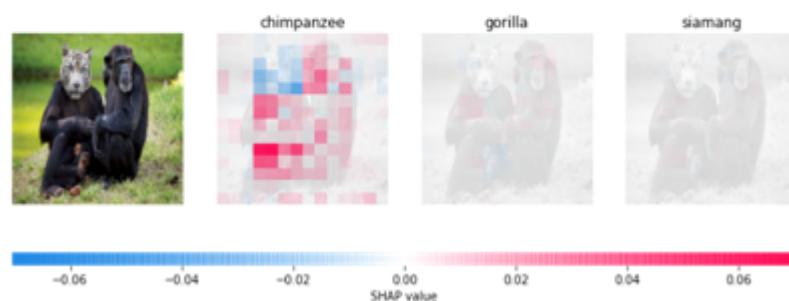


Figura 38.: Risultati ottenuti con VGG16

### A.3 SHAP KERNEL EXPLAINER

Risultati ottenuti utilizzando SHAP Kernel Explainer

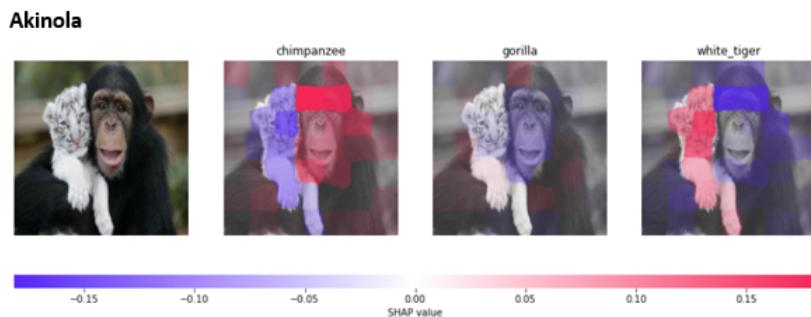


Figura 39.: Risultati ottenuti con Akinola

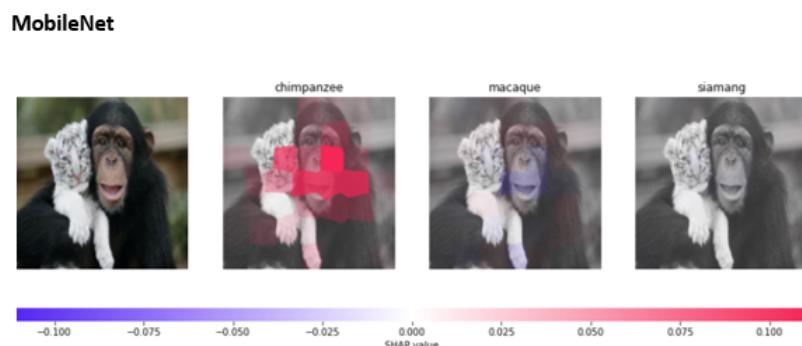


Figura 40.: Risultati ottenuti con MobileNet

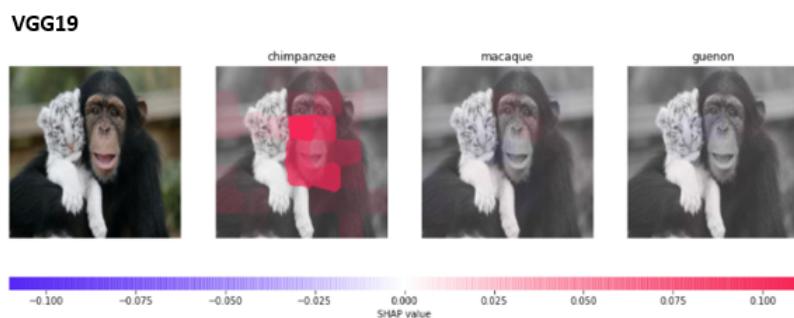


Figura 41.: Risultati ottenuti con VGG19

#### A.4 TF-EXPLAIN

Risultati ottenuti utilizzando tf-explain

InceptionV3



Figura 42.: Risultati ottenuti con InceptionV3

MobileNet

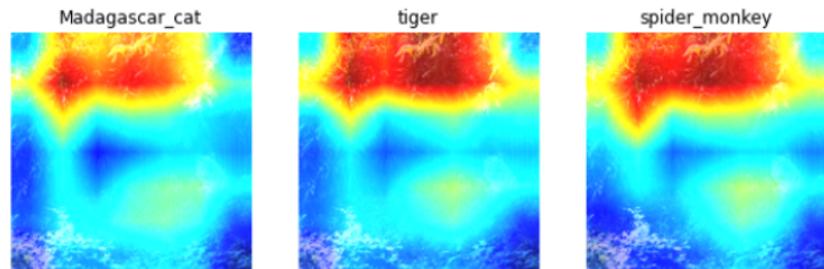


Figura 43.: Risultati ottenuti con MobileNet

VGG16

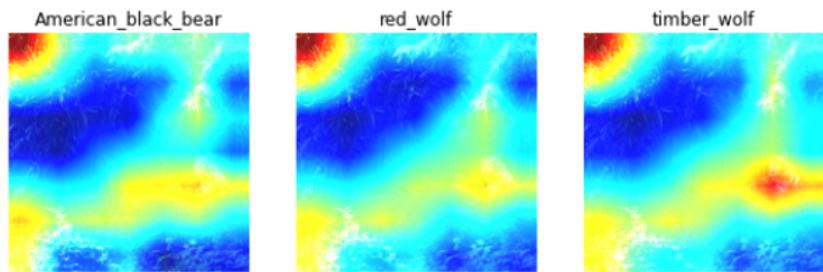


Figura 44.: Risultati ottenuti con VGG16

VGG19

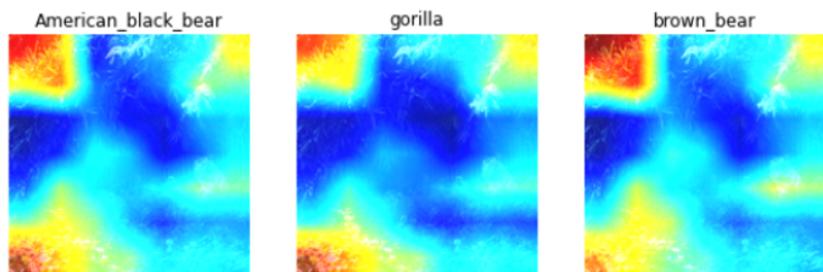


Figura 45.: Risultati ottenuti con VGG19

---

## BIBLIOGRAFIA

---

- [1] IBM Cloud Education - *What is Artificial Intelligence (AI)?* - <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence> (Cited on pages 7 and 8.)
- [2] IBM Cloud Education - *What is Machine Learning?* - <https://www.ibm.com/cloud/learn/machine-learning> (Cited on page 9.)
- [3] IBM Cloud Education - *What are Neural Networks?* - <https://www.ibm.com/cloud/learn/neural-networks> (Cited on page 12.)
- [4] IBM Cloud Education - *What are Convolutional Neural Networks?* - <https://www.ibm.com/cloud/learn/convolutional-neural-networks> (Cited on pages 18 and 19.)
- [5] IBM Cloud Education - *What is Deep Learning?* - <https://www.ibm.com/cloud/learn/deep-learning> (Cited on page 19.)
- [6] Michael Copeland - *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?* - <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/> (Cited on page 7.)
- [7] Eda Kavlakoglu - *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?* - <https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks> (Cited on page 7.)
- [8] Nvidia - *Deep Learning* - <https://developer.nvidia.com/deep-learning> (Cited on page 7.)
- [9] Bernard Marr - *What Are Artificial Neural Networks - A Simple Explanation For Absolutely Anyone* -

- <https://www.forbes.com/sites/bernardmarr/2018/09/24/what-are-artificial-neural-networks-a-simple-explanation-for-absolutely-anyone/?sh=58bdb62d1245> (Cited on page 12.)
- [10] Tensorflow - <https://www.tensorflow.org/> (Cited on page 17.)
- [11] Pytorch - <https://pytorch.org/> (Cited on page 17.)
- [12] Keras - <https://keras.io/> (Cited on page 17.)
- [13] Keiron O'Shea and Ryan Nash - *An Introduction to Convolutional Neural Networks* - <https://arxiv.org/pdf/1511.08458.pdf> (Cited on pages 18 and 19.)
- [14] Cezanne Camacho - *Convolutional Neural Networks* - [https://cezannec.github.io/Convolutional\\_Neural\\_Networks/](https://cezannec.github.io/Convolutional_Neural_Networks/) (Cited on pages 3, 18, and 19.)
- [15] Jason Brownlee - *Different Types of Learning in Machine Learning* - <https://machinelearningmastery.com/types-of-learning-in-machine-learning/>
- [16] Arun Rai - *Explainable AI: from black box to glass box* - <https://link.springer.com/content/pdf/10.1007/s11747-019-00710-5.pdf> (Cited on pages 20 and 22.)
- [17] Giorgio Visani - *Explainable Machine Learning* - <https://towardsdatascience.com/explainable-machine-learning-9d1ca0547aeo> (Cited on page 25.)
- [18] Matt Turek - *Explainable Artificial Intelligence (XAI)* - <https://www.darpa.mil/program/explainable-artificial-intelligence> (Cited on pages 3, 20, and 23.)
- [19] Bryce Goodman, Seth Flaxman - *European Union regulations on algorithmic decision-making and a "right to explanation"* - <https://arxiv.org/abs/1606.08813> (Cited on page 20.)
- [20] Feiyu Xu,Wei Fan - *Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges* - paper

- [21] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, Noémie Elhadad - *Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission* - Paper (Cited on page 21.)
- [22] Cameron Langford - *Houston Schools Must Face Teacher Evaluation Lawsuit* - Paper (Cited on page 21.)
- [23] Sebastian Lapuschkin , Alexander Binder , Gregoire Montavon , Klaus-Robert Muller and Wojciech Samek - *Analyzing Classifiers: Fisher Vectors and Deep Neural Networks* - <http://iphome.hhi.de/samek/pdf/LapCVPR16.pdf> (Cited on page 21.)
- [24] Mireia Ribera, Agata Lapedriza - *Can we do better explanations? A proposal of User-Centered Explainable AI* - Paper (Cited on page 25.)
- [25] Tameru Hailesilassie - *Rule Extraction Algorithm for Deep Neural Networks: A Review* - <https://arxiv.org/ftp/arxiv/papers/1610/1610.05267.pdf> (Cited on page 25.)
- [26] Gregor Stiglic , Primoz Kocbek , Nino Fijacko , Marinka Zitnik , Katrien Verbert , Leona Cilar - *Interpretability of Machine Learning based prediction models in healthcare* - <https://arxiv.org/ftp/arxiv/papers/2002/2002.08596.pdf> (Cited on pages 3 and 26.)
- [27] Samo Plibersek - *Explainable and privacy-preserving Artificial Intelligence* - <https://dotscience.com/blog/2019-11-28-explainable-ai-part-1/>
- [28] Paolo Galeone - *Introduzione al Machine Learning* - <https://pgaleone.eu/italian-machine-learning-course/> (Cited on pages 3, 12, and 14.)
- [29] Isha Salian - *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* - <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/> (Cited on page 10.)
- [30] Wojciech Samek, Thomas Wiegand, Klaus-Robert Muller - *Explainable Artificial Intelligence: understanding, visualizing and interpreting*

- Deep Learning models* - <https://arxiv.org/pdf/1708.08296.pdf> (Cited on page 24.)
- [31] XGBoost - *XGBoost Documentation* - <https://xgboost.readthedocs.io/en/latest/> (Cited on page 28.)
- [32] Scott Lundberg - *SHAP* - <https://github.com/slundberg/SHAP> (Cited on pages 3, 31, and 38.)
- [33] Marco Tulio Correia Ribeiro - *LIME* - <https://github.com/marcotcr/LIME> (Cited on pages 3, 31, and 37.)
- [34] TeamHG-Memex - *ELI5* - <https://github.com/TeamHG-Memex/eli5> (Cited on page 36.)
- [35] PAIR code - *What-If Tool* - <https://pair-code.github.io/what-if-tool/> (Cited on page 40.)
- [36] Trusted-AI - *AIX360* - <https://github.com/Trusted-AI/AIX360> (Cited on page 35.)
- [37] ORACLE - *Skater* - <https://github.com/oracle/Skater> (Cited on page 38.)
- [38] sicara - *tf-explain* - <https://github.com/sicara/tf-explain> (Cited on page 39.)
- [39] Christoph Molnar - *Interpretable Machine Learning* - <https://christophm.github.io/interpretable-ml-book/> (Cited on pages 29, 30, 37, and 38.)
- [40] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin - "Why Should I Trust You?": *Explaining the Predictions of Any Classifier* - <https://arxiv.org/pdf/1602.04938.pdf> (Cited on page 30.)
- [41] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra - *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization* - <https://arxiv.org/pdf/1610.02391v4.pdf> (Cited on pages 3, 39, and 40.)
- [42] Fjodor Van Veen - *The Neural Network Zoo* - <https://www.asimovinstitute.org/neural-network-zoo/> (Cited on pages 3 and 13.)

- [43] IBM Research - *One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques* - <https://arxiv.org/pdf/1909.03012.pdf> (Cited on page 36.)
- [44] Anita JainSamiran, NundySamiran, Kamran Abbasi - *Corruption: Medicine's dirty open secret* - Paper (Cited on page 33.)
- [45] Trusted-AI - *Adversarial Robustness Toolbox* - <https://github.com/Trusted-AI/adversarial-robustness-toolbox> (Cited on page 40.)
- [46] AdvFamily - *AdvBox* - <https://github.com/advboxes/AdvBox> (Cited on page 41.)
- [47] Bethge Lab - *FoolBox* - <https://github.com/bethgelab/foolbox> (Cited on page 41.)
- [48] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, Boris Katz - *ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models* - <https://objectnet.dev/objectnet-a-large-scale-bias-controlled-dataset-for-pushing-the-limits-of-object-recognition-models.pdf> (Cited on page 64.)