POLITECNICO DI TORINO

Corso di Laurea Magistrale

in Ingegneria Informatica (Computer Engineering)

Tesi di Laurea Magistrale

iNNvestigate-GUI: analisi e visualizzazione di reti neurali convoluzionali a portata di click

Relatori

Prof. Fabrizio Lamberti

Dr.ssa Lia Morra

Candidato

Davide Valentino

Ottobre, 2019

Al me di un futuro non troppo lontano,
che tu possa aver realizzato tutti i tuoi sogni,
e continuare a vivere per tutto ciò in cui credi.

Ringraziamenti

Ringrazio innanzitutto i miei relatori, il professor Fabrizio Lamberti per avermi permesso di intraprendere questo percorso, e la dottoressa Lia Morra per avermi accompagnato, mostrandomi la via e correggendo i miei errori, restando disponibile fino alla fine.

Ringrazio poi tutti gli amici, colleghi, famigliari, che in un modo o nell'altro mi hanno supportato, che mi sono rimasti vicino e mi hanno sopportato in questi ultimi, impegnativi mesi, e per tutti gli anni della mia carriera al Politecnico. Qualcuno è rimasto, qualcuno ha cambiato strada, ringrazio tutti portando con me qualche rimorso e qualche rimpianto.

Grazie a te, Eleonora, che sei entrata nella mia vita quando meno me l'aspettavo, e hai stravolto la mia quotidianità e mi hai permesso di vivere con te tanti dei momenti migliori della mia esistenza, in mezzo a libri, lavoro, esami, e chilometri. Mi hai sopportato, tanto, per tanto tempo, e non ti sei mai fatta scappare un momento per dirmi quanto fossi fiera di me, per supportarmi e tenermi la mano quando la strada si faceva più dura. Grazie per tutto, grazie di esserci sempre.

Grazie a mio fratello Riccardo, per tutte le volte che mi hai capito senza bisogno di dire altro, che mi hai lasciato la stanza per studiare e mi hai sopportato in tutti questi anni di università. Hai dovuto perfino andare via un anno intero per lasciarmi studiare, ed io non potrei essere più orgoglioso di quello che hai fatto in quest'anno, ed anche un po' invidioso. Mi auguro di poterti continuare a conoscere sempre di più, e di riuscire ad esserci sempre per te.

Ed infine grazie ai miei genitori, a mio papà, che nella distanza non ha mai smesso di sentirsi orgoglioso di me e di esserci se avevo bisogno, e a mia mamma, che ha dovuto sopportarmi più di tutti: scusa per tutte le volte che sono stato ingestibile e pesante, grazie di aver provato in ogni modo a capirmi e a darmi il tuo supporto. Grazie mamma, grazie papà, per avermi fatto

crescere con dei valori che porterò sempre con me, per avermi fatto diventare l'uomo che sono ora, con tutti i miei pregi e difetti. Grazie, per avermi permesso di seguire la strada che ho sempre voluto, fino alla fine, nonostante le difficoltà, nonostante le incertezze, grazie per aver messo sempre al primo posto me e Riccardo. Non sarei quello che sono ora se non fosse stato per tutto quello che avete fatto per me, non sarei arrivato dove mi trovo ora. Grazie, per esserci sempre stati, nel bene e nel male, e avermi sempre fatto sentire prezioso. Questa tesi è dedicata anche a voi, a tutto quello che avreste voluto fare e non avete potuto, a tutto quello che invece siete riusciti a raggiungere, e a tutto quello che avete fatto per me. Grazie papà, grazie mamma.

Abstract

Dalla fine degli anni ’80 del Novecento ai primi anni Duemila, una serie di passi avanti negli algoritmi utilizzati e il superamento di alcuni limiti tecnologici, ha permesso la grande diffusione delle reti neurali fino alla nascita del Deep Learning come è noto ai giorni nostri. Questo ha incrementato la necessità di comprendere il funzionamento interno delle reti neurali, sia per poter agire più consapevolmente nello sviluppo delle stesse, sia per poterle utilizzare anche negli ambiti più critici, dove la definizione di “black-box”, fino ad allora attribuita alle reti, non risultava sufficiente.

Se negli ultimi anni sono state perciò introdotte svariate tecniche di visualizzazione, in grado di fornire rappresentazioni delle reti neurali utili proprio per aumentarne la comprensibilità, pochi invece risultano gli strumenti disponibili per la loro integrazione nel processo di sviluppo o di utilizzo di una rete: la necessità di dover scrivere del codice ad-hoc e la mancanza di implementazioni di riferimento per tali tecniche ne scoraggia spesso l’utilizzo, soprattutto da parte di utenti meno esperti. Nella ricerca di strumenti utili per risolvere tale problema, e prendendo in considerazione le Convolutional Neural Network, sia per la loro estrema diffusione sia perché proprio per questo maggiormente oggetto di ricerca, si è rintracciato iNNvestigate, uno strumento a linea di comando che rappresenta una soluzione piuttosto completa, offrendo un’implementazione standard e unificata per molte delle tecniche di visualizzazione di attivazioni proposte in letteratura.

In questa tesi viene presentato iNNvestigate-GUI, uno strumento che sfrutta tutte le funzionalità offerte da iNNvestigate e ne aumenta le potenzialità in un’interfaccia grafica in ambiente web semplice ed intuitiva. Oltre ad ampliare il già vasto parco di tecniche di visualizzazione delle attivazioni incluse in iNNvestigate, e arricchendolo di funzionalità pensate appositamente per

enfatizzare i confronti tra i risultati ottenuti, iNNvestigate-GUI offre un'altra importante funzione. È infatti possibile ottenere diverse viste che forniscono informazioni e grafici interattivi utili ad evidenziare i casi più interessanti da analizzare, evitando all'utente di dover effettuare tale operazione manualmente. Ad esempio, fornendo un certo numero di modelli di reti in input, è possibile visualizzare graficamente come questi si comportino nella classificazione di un dataset di immagini, visualizzando la relazione tra confidenza e coerenza dei risultati ottenuti. Le immagini così suggerite possono dunque venire utilizzate come input per le tecniche di visualizzazione disponibili, per effettuare analisi più mirate.

Nonostante sia equipaggiato con funzionalità interessanti per tutti i progettisti e utilizzatori di reti neurali, si ritiene che il target principale di iNNvestigate-GUI siano gli utenti meno esperti. Per questo motivo, si è scelto di includere una valutazione dell'usabilità e delle potenzialità del tool riferendosi in particolar modo a questo tipo di utenza, raccogliendo i loro feedback riguardo alle principali modalità di utilizzo.

iNNvestigate-GUI è una web-application realizzata in Python utilizzando Flask e, sulla base di iNNvestigate, opera in Keras utilizzando TensorFlow come back-end. L'interfaccia web è stata creata utilizzando HTML, JavaScript, CSS e le librerie jQuery e D3, e tutte le interazioni tra client e server avvengono tramite AJAX.

Indice

1 Background [1-4]

1.1 Reti neurali

Una rete neurale è un modello computazionale in grado di implementare una qualsiasi funzione che, a partire da un certo numero di dati in input, produce un output attraverso un meccanismo interno ispirato al modello di riconoscimento del cervello umano, da cui il nome (in inglese: Artificial Neural Network - ANN; o Neural Network - NN).

Il cervello umano, per quanto concerne le sue funzioni di riconoscimento visivo, può essere modellizzato come un sistema di layer, corrispondenti alle cortecce visive, formati da milioni di neuroni, ciascuno dei quali produce un output a partire dagli input ricevuti dagli occhi, grazie ad un "allenamento" di milioni di anni di evoluzione dell'umanità. A partire dalla corteccia visiva primaria (V1) i risultati dei neuroni che la compongono vengono dunque combinati in modo da effettuare il riconoscimento di informazioni più "semplici", come ad esempio le forme degli oggetti, e via via passando alle altre cortecce (V2, V3, V4, V5) il nostro cervello è in grado di riconoscere informazioni sempre più complesse che ci permettono di comprendere ciò che vediamo.

Le reti neurali sono quindi costituite da blocchi centrali chiamati **neuroni**, organizzati in **layer** che, a partire dal cosiddetto input layer, sono in grado di elaborare le informazioni prodotte dagli hidden layer intermedi fino a produrre l'output desiderato. Esse soddisfano il teorema dell'universalità, in quanto sono, almeno teoricamente, in grado di implementare qualsiasi funzione. Ciò che ne limita le possibilità si può ricondurre essenzialmente a tre fattori: è necessario disporre di una quantità sufficiente di **dati annotati**, come ad esempio immagini etichettate adeguatamente, sui quali allenarle in maniera opportuna; è necessario utilizzare algoritmi che

permettano di **impostare tutti i parametri della funzione da implementare**, in maniera efficace ed efficiente; infine, è necessario disporre di **tecnologia in grado di poter eseguire le operazioni richieste in un tempo sostenibile**.

Le reti neurali esistono fin dagli anni ‘50 del Novecento, nella loro concezione più semplice dove i neuroni venivano chiamati percettroni: classificatori binari che producono in output una classificazione 0/1, sulla base della rilevanza assegnata a ciascuno degli input attraverso numeri reali chiamati **pesi**. In particolare, se la somma delle rilevanze di tutti gli input del percettrone supera una certa soglia, chiamata **bias**, l’output è uguale a 1, altrimenti 0.

Si nota dunque che, per far sì che la rete neurale produca gli output desiderati, bisogna agire opportunamente su pesi e bias nella cosiddetta fase di allenamento. Il problema con i percettroni è che questa operazione risulta piuttosto complessa se non impossibile, a seconda delle dimensioni della rete, in quanto non c’è modo di far corrispondere un piccolo cambiamento di pesi o bias con un piccolo cambiamento nell’output prodotto, dato che i risultati sono sempre o 0 o 1, o un estremo o l’altro. In breve tempo si è dunque passati ad associare al neurone (o nodo, o semplicemente unità) una **funzione di attivazione**, inizialmente la funzione sigmoide (vedi figura 2), in modo tale che l’output potesse essere sì 0 o 1, ma anche un valore qualsiasi tra di essi. Per questo motivo l’output dei neuroni viene spesso chiamato “attivazione”, in quanto in effetti corrisponde all’output della funzione di attivazione applicata all’output stesso del neurone. L’introduzione di tale meccanismo ha reso possibile l’utilizzo di algoritmi per la scelta di pesi e bias in maniera automatica, come il **Gradient Descent** [5] (trattato poco sotto), rendendo dunque possibile l’allenamento automatico della rete.

La fase di training di una rete neurale è cruciale per il suo stesso funzionamento, in quanto è ciò che determina quali saranno gli output della rete. Per far ciò, è necessaria la presenza di un training set, un insieme di input con associato il relativo output desiderato, in modo da poter effettuare il confronto con gli effettivi output della rete. L’algoritmo di Gradient Descent infatti, permette l’allenamento della rete andando a minimizzare, per tutti gli input del training set, una funzione di costo il cui contributo maggiore è determinato dalla differenza tra l’output desiderato e l’output ottenuto. In particolare, il gradiente della funzione di costo rispetto a pesi e bias

attuali viene utilizzato per determinare la direzione del cambiamento degli stessi in modo tale che la funzione di costo diminuisca, e l'ammontare di tale cambiamento dipende da un cosiddetto iper-parametro della rete chiamato learning rate.

Con questo livello di complessità le reti neurali cominciavano dunque, già agli inizi della seconda metà del Novecento, a poter essere utilizzate per implementare con successo alcuni tipi di funzioni. Riferendoci nuovamente all'analogia con il cervello umano però, appare evidente che, per poter svolgere sempre più compiti, c'era bisogno di un'evoluzione che permettesse alle reti neurali di poter gestire dimensioni enormemente maggiori, per quanto riguarda sia il numero di layer sia il numero di neuroni. Il primo passo verso ciò che è oggi così frequentemente al centro dell'attenzione in cosi tanti ambiti, il Deep Learning, fu fatto nel 1986 quando, grazie a David Rumelhart, Geoffrey Hinton, e Ronald Williams, per la prima volta si capì l'importanza dell'algoritmo di **backpropagation** introdotto già negli anni '70, e ciò diede il via all'utilizzo delle reti neurali di tipo deep.

1.2 Il successo delle DNN

Le **Deep Neural Network (DNN)** sono reti neurali caratterizzate da un gran numero di hidden layer, a differenza delle NN standard dove può esserci anche un solo hidden layer. L'importanza delle DNN sta nel fatto che rispecchiano ancora più fedelmente il modello di riconoscimento visivo del cervello umano in quanto, essendo costituite da un gran numero di layer, esse sono in grado, livello dopo livello, di riconoscere via via informazioni sempre più complesse aggregando le informazioni più semplici riconosciute ai livelli precedenti. Questo permette alle DNN di raggiungere altissimi livelli di precisione che al giorno d'oggi, ormai in diversi ambiti, sono addirittura superiori a quelli umani.

Prima del 1986 però, era impossibile allenare DNN in quanto gli algoritmi di training, come Gradient Descent visto prima, risultavano troppo lenti rendendo le DNN inutilizzabili. Per questo motivo vennero introdotti algoritmi come **Stochastic Gradient Descent (SGD)** [5] dove

invece di calcolare i gradienti della funzione di costo da minimizzare per tutti gli input del training set, il calcolo viene effettuato solo per alcuni input, scelti in maniera stocastica. SGD, nelle sue varianti, risulta tutt'oggi molto utilizzato, ma ciò che ha contribuito maggiormente alla diffusione delle DNN è stata l'introduzione del suddetto algoritmo di backpropagation per la fase di training. Questo ha permesso infatti di sostituire la moltitudine di passaggi forward, che all'interno di una rete neurale dovevano essere eseguiti per calcolare i gradienti necessari per SGD, con due soli passaggi: uno forward per il calcolo degli output della rete ad ogni layer e uno backward (da cui il nome dell'algoritmo) che, a partire dagli output della rete e dunque layer dopo layer fino all'input, calcola in maniera simultanea tutte le derivate parziali utilizzate nel calcolo dei gradienti necessari per SGD.

A questo punto, per dare vita al mondo del Deep Learning così come lo conosciamo oggi, rimanevano due barriere da superare: l'ottenimento dell'accesso a grandi quantità di dati annotati, e il limite tecnologico. Le implementazioni delle reti neurali si basano principalmente su operazioni matriciali di dimensioni enormi, e le CPU, anche le più moderne, non sono in grado di elaborare questo tipo di dati in maniera efficiente, poiché sono create per essere general purpose. Grazie all'interesse del mercato dei videogiochi nello sviluppo di **GPU sempre più potenti** e performanti, il cui scopo è proprio l'efficienza nell'elaborazione di operazioni matriciali, il limite tecnologico veniva superato verso la fine del Novecento. Dagli inizi degli anni Duemila poi, la creazione e la diffusione da parte di diverse entità come università, centri di ricerca e aziende private (come Google o Amazon), di **enormi dataset annotati** pensati sempre di più proprio per il mondo del Deep Learning, ha fatto sì che anche l'ultima barriera venisse superata. Da quel momento in poi le DNN hanno avuto sempre più successo, grazie a nuovi algoritmi di ottimizzazione per quanto riguarda il training della rete (come ad esempio Adagrad, RMSprop, Adam e molti altri [5]), grazie all'utilizzo di nuove funzioni di attivazione che si sono rivelate più performanti (es. ReLU [6]) e grazie all'introduzione di nuove operazioni di regolarizzazione, utili per migliorare l'efficacia della fase di apprendimento (come Dropout [7] e Batch-normalization [8]).

A questo punto, dunque, lo sviluppo delle DNN ha seguito strade diverse, a seconda dell'ambito nel quale venivano implementate ed utilizzate e a seconda delle caratteristiche che, per quell'ambito, era necessario aggiungere alle DNN "standard". Si assiste dunque all'introduzione di Convolutional Neural Network, Recurrent Neural Network, Long Short-Term Memory units, Deep Belief Network, Boltzman machines, e altre ancora [9].

Quello su cui si focalizzerà questo lavoro sono le reti neurali convoluzionali (**CNN**).

1.3 Convolutional Neural Network

Le **reti neurali convoluzionali** sono reti neurali di tipo deep che si caratterizzano tra le DNN in quanto comprendono layer cosiddetti convoluzionali, che si sono rivelati di fondamentale importanza per quanto riguarda il processo di apprendimento delle reti che trattano dati visivi. Le CNN infatti lavorano su immagini e video, e attraverso i **convolutional layer** sono in grado di comprendere la funzione di ogni pixel in input, anche tenendo conto di come sono organizzati spazialmente, in modo da produrre l'output finale, rappresentando in maniera ancora più verosimile il modello di riconoscimento visivo del cervello umano (figura 1).

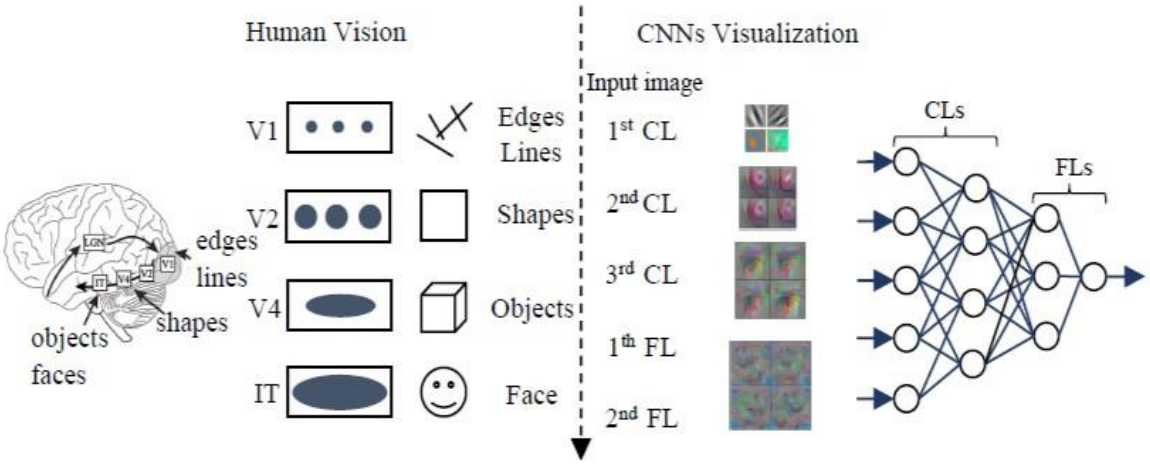

Figura 1 [47] – Come le CNN vedono il mondo

In particolare, i layer convoluzionali applicano dei **filtri** (**convoluzionali**) sui dati in input, tipicamente costituiti da matrici 2D o 3D, durante la cosiddetta operazione di convoluzione: in questo modo vengono messe in evidenza particolari caratteristiche dell'immagine in input, a seconda del filtro applicato. Tali caratteristiche sono quelle che vengono definite le **feature** riconosciute dalla rete, che possono corrispondere, a seconda della profondità alla quale si trova il layer convoluzionale, a pattern più o meno semplici, rappresentazioni più o meno significative e di complessità maggiore man mano che la profondità aumenta. L'output di un nodo di un layer convoluzionale è dunque una rappresentazione della feature riconosciuta dal filtro convoluzionale caratteristico di quel nodo, attraverso una matrice dimensionata opportunamente in base alle dimensioni del filtro e del tipo di immagine in input, ad esempio se in bianco e nero o a colori.

Per tenere in considerazione come sono organizzati i pixel nello spazio, ai fini del riconoscimento di feature più complesse e significative, oltre ai veri e propri layer convoluzionali vengono inseriti i layer di pooling, che hanno la funzione di aggregare, in vario modo, diverse porzioni dell'input in modo che la rete possa avere una visione dell'immagine più d'insieme e meno dettagliata. Le varie modalità con cui i **pooling layer** effettuano tali aggregazioni (downsampling) ne definiscono la tipologia: due tra i più famosi sono il Max Pooling layer e l'Average Pooling layer che consistono nell'estrarre i pixel significativi di una porzione dell'immagine in input scegliendo rispettivamente il pixel con valore maggiore o il valore medio dei pixel all'interno della parte di immagine analizzata.

In questo modo le CNN sono in grado di riconoscere le feature delle immagini in input, a partire da semplici pattern (come linee dritte, o angoli), fino ad arrivare a rappresentazioni significative (ad esempio la faccia di un uomo), che infine vengono combinate dai cosiddetti **dense layer** (o fully connected layer). Questi sono layer dal funzionamento analogo a quanto visto per le DNN in generale, costituiti dunque da neuroni che ricevono svariati input, ciascuno dei quali associato ad un peso, e producono un output sulla base del confronto con una soglia (bias). Gli input di questi layer nelle CNN corrispondono alle feature imparate dalla rete grazie ai layer

convoluzionali (e di pooling) precedenti, e l'output corrisponde dunque al riconoscimento di un insieme di feature nell'immagine in input, fino ad arrivare al risultato finale della rete.

Durante la fase di apprendimento di una CNN dunque, oltre alla modifica di pesi e bias dei dense layer, vengono modificati i valori dei filtri convoluzionali, in modo che siano in grado di riconoscere le feature rappresentative delle immagini. Come detto in precedenza, l'allenamento necessita dell'utilizzo di funzioni di attivazione (figura 2) applicate all'output di ciascun nodo: per quanto riguarda in particolare i layer convoluzionali, la funzione di attivazione è applicata su ciascun valore contenuto nelle matrici in output ai vari nodi (corrispondenti alle feature apprese). Una delle funzioni di attivazione più utilizzata negli hidden layer delle CNN, e in generale nelle DNN, è la **ReLU** (Rectified Linear Unit), già citata in precedenza. In particolare, con ReLU, l'attivazione risulta uguale all'output generato dal nodo sul quale è applicata se tale valore è positivo, mentre è nulla per output negativi. Per via dei risultati ottenuti nell'allenamento delle reti utilizzando tale funzione di attivazione, negli anni l'utilizzo di ReLU è diventato standard de-facto per quanto riguarda i layer intermedi, mentre per l'output layer, dove le uscite ai vari neuroni devono corrispondere al risultato effettivo della rete, vengono usate altre funzioni di attivazione, a seconda del problema risolto dalla rete (ad esempio in un problema di

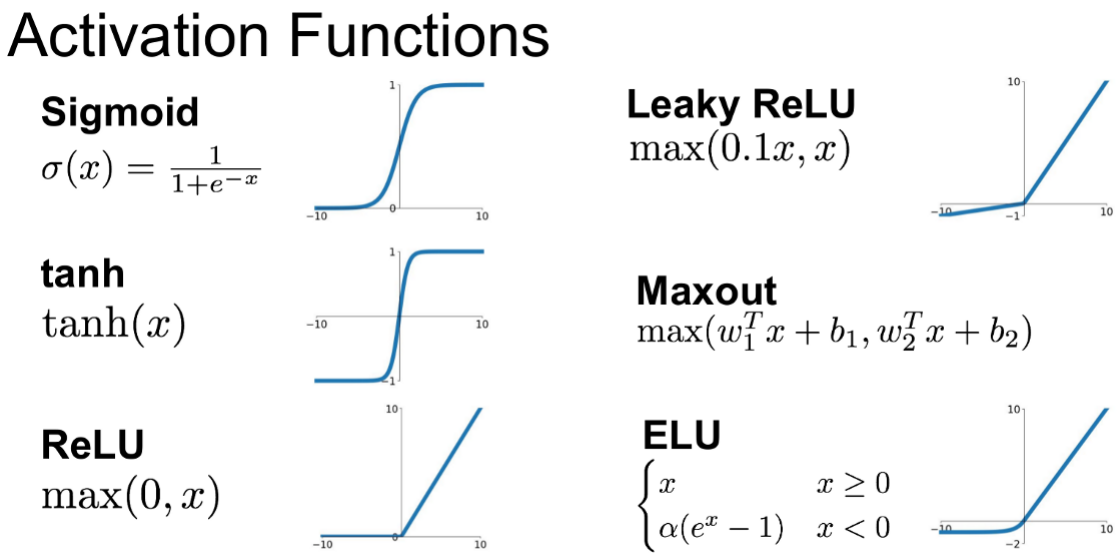

Figura 2 [48] – Principali funzioni di attivazione

classificazione, dove l'output dei neuroni dell'ultimo layer rappresenta la probabilità che la classe corrispondente a ciascun neurone sia la classe che la rete ha riconosciuto nell'immagine in input, una delle funzioni di attivazione più utilizzate per l'output layer è SoftMax, che in pratica rimodula i risultati dei nodi su cui è applicata in modo che siano compresi tra 0 e 1 e la somma tra di essi sia uguale a 1).

Per via della loro caratteristica di operare su dati visivi, le CNN sono probabilmente una delle tipologie di DNN più utilizzate, e si ritrovano oggi in molti dei processi di sviluppo delle più svariate applicazioni tecnologiche. Come il resto delle DNN però, le reti neurali convoluzionali risultano di difficile comprensione per via del loro funzionamento interno, che le rende così straordinariamente potenti allo stesso tempo. Per questo motivo, come verrà trattato nel paragrafo 1.4, è stata fin da subito evidente l'importanza dell'introduzione di tecniche utili alla loro comprensione, nello specifico di **tecniche di visualizzazione** delle reti neurali.

1.4 L'importanza della visualizzazione

Come si è detto, le reti neurali godono della proprietà di universalità, ovvero sono in grado di svolgere qualsiasi funzione, supponendo di avere un sistema che sia in grado di definire correttamente i parametri necessari all'ottenimento dell'output di tale funzione in tempi umanamente sostenibili. Ciò che permette la definizione dei parametri corretti per tale funzionamento è la backpropagation, e i tempi per tale processo sono diventati sempre più accettabili grazie all'avanzamento tecnologico delle GPU. Dunque, dalla loro nascita, le DNN sono state implementate nei più svariati ambiti portando sempre delle migliorie, spesso sorprendenti.

Quello che non si è detto fino a qui è che, se per un modello computazionale standard siamo in grado di definire il suo comportamento in termini di flusso di operazioni che da un input porta ad un output, questo non è possibile per le reti neurali. Queste infatti sono sì in grado di svolgere qualsiasi funzione, ma il flusso interno di operazioni che portano dall'ingresso ai risultati della rete non è esprimibile in maniera simbolica, ed è dunque impossibile per un essere umano

comprendere cosa avvenga al loro interno. Questa caratteristica è ciò che porta a definire le reti neurali, ed in particolare le DNN, trattandosi di strutture più complesse per definizione, come "**black-box**", scatole nere che, dato un input ed una configurazione ad hoc degli iper-parametri di funzionamento, sono in grado di generare un output più o meno accettabile rispetto a ciò che ci si aspetta, senza però poter comprendere come si è arrivati a tale risultato.

Questa particolarità delle reti neurali porta a due conseguenze immediate. Da un lato, risulta impossibile o estremamente difficile per un essere umano poter agire sulla struttura di una rete o sugli algoritmi che ne regolano il funzionamento, non riuscendo a comprenderne il funzionamento interno, e ciò ne limita le possibilità di evoluzione. Dall'altro lato, se in alcuni ambiti di applicazione tale inconsapevolezza può essere tollerata, esistono alcuni settori in cui non è possibile potersi "fidare" del comportamento della rete senza avere nessuna comprensione del funzionamento per poterne verificare i risultati.

Con la diffusione delle DNN dunque, si è sviluppata velocemente anche la necessità di poterle comprendere, sia per poterle migliorare consapevolmente, sia per poterle applicare in ambiti più restrittivi (ad esempio applicazioni medicali, o militari). Dalla fine del Novecento quindi, la ricerca ha iniziato a dedicarsi sempre di più, e sempre più efficacemente, alla comprensione del funzionamento delle reti, ed in particolare questo si traduce nella nascita di un settore di studio dedicato alla visualizzazione delle reti neurali.

La **visualizzazione** permette di poter comprendere meglio il funzionamento interno delle reti neurali, portando quindi al duplice obiettivo di dare la possibilità di poterle costruire e far evolvere in maniera migliore, e contemporaneamente di aumentarne la diffusione, grazie alla possibilità di riporre maggiore fiducia nei risultati delle stesse.

Visualizzare una rete neurale comunque, non corrisponde ad un solo tipo di concetto. La visualizzazione può avvenire infatti a diversi livelli, e la categorizzazione che si utilizzerà nel seguito nell'approfondire questo argomento riguarda **cosa visualizzare di una rete**. Come detto, ci si concentrerà in particolar modo sulle CNN, sia perché risultano ad oggi molto diffuse per via degli infiniti utilizzi, sia perché, per questo motivo, maggiori sono stati gli studi e le ricerche

effettuate per quanto riguarda la loro visualizzazione. Tuttavia, molti dei concetti descritti nel seguito sono da ritenersi ugualmente validi parlando di DNN e in generale di reti neurali nella maggior parte dei casi.

2 Visualizzazione di CNN

2.1 Cosa visualizzare

La trattazione che segue, che rappresenta lo stato dell'arte per quanto riguarda cosa visualizzare di una CNN, è suddivisa in base all'oggetto della visualizzazione e si basa principalmente sulla categorizzazione proposta da Hohman, F. *et al.* [10]. Nello specifico, come oggetto della visualizzazione si parlerà di **grafi computazionali** (2.1.1), riferendosi alla rappresentazione della struttura dei modelli, dunque dei **pesi** della rete (2.1.2), rappresentati graficamente come linee più o meno spesse, a seconda del loro valore, che collegano tra di loro input e output dei dense layer, per poi trattare di **attivazioni**, **feature** e **gradienti** (2.1.3), argomento di maggior interesse nella ricerca odierna e per questo anche al centro di questo lavoro, le cui tecniche di visualizzazione verranno ulteriormente approfondite in un paragrafo dedicato (2.2). Infine, si parlerà di visualizzazione di **embedding** (2.1.4), rappresentazioni bi- o tri-dimensionali dei vettori a molte dimensioni corrispondenti alle feature riconosciute dalla rete, che permettono di effettuare analisi d'insieme sul modello intero, e di visualizzazione di **metriche** (2.1.5), riferendosi a grafici riguardanti ad esempio l'errore in output o l'accuratezza delle reti.

In tutti i casi in cui le tecniche sotto analisi non siano state sviluppate ad hoc per le CNN, si parlerà in generale di reti neurali, riferendosi comunque principalmente a DNN (in quanto per reti neurali cosiddette "superficiali" tali visualizzazioni non sono così diffuse).

2.1.1 Grafo computazionale

Una delle possibili visualizzazioni di una rete neurale è la rappresentazione della struttura del modello in sé, il cosiddetto computational graph. Questo mostra l'architettura della rete, come avvengono le fasi di train e test, come i dati vengono salvati sul disco e così via. La tecnica più utilizzata per la visualizzazione di tali informazioni è il **Node-link diagram**, che nella maggior parte dei casi viene costruito rappresentando le operazioni di cui è composto il modello come nodi e le variabili come archi. Questo è diventato lo standard per via della grande diffusione di **TensorBoard** [29] (figura 24), un tool sviluppato per essere nativamente affiancato a **TensorFlow**[1] (uno dei più diffusi framework per la generazione di modelli di reti neurali), e che verrà trattato nel paragrafo 2.3.

2.1.2 Pesi

Come detto, le reti neurali sono composte da una serie di layer di diverso tipo, che definiscono le caratteristiche e il funzionamento della rete stessa, ciascuno dei quali è composto da un certo numero di neuroni, ovvero i punti della rete dove avviene l'apprendimento. Nelle CNN, i layer principali sui quali effettuare le visualizzazioni sono i convolutional layer e i fully connected layer. Per quanto riguarda questi ultimi, l'apprendimento avviene andando ad agire sul valore dei pesi, che rappresentano l'interconnessione tra i neuroni di un layer e quelli del layer successivo. I convolutional layer invece, applicano filtri sui dati in input durante l'operazione di convoluzione. I valori di questi filtri, insieme ai pesi dei fully connected layer, sono i parametri fondamentali che vengono modificati durante la fase di allenamento della rete, e può essere molto utile visualizzarli in modo da capire come funziona il processo di apprendimento.

Per quanto riguarda la visualizzazione dei filtri di convoluzione, la tecnica più utilizzata è quella di rappresentare l'effetto dell'applicazione dei filtri sui dati di input, visualizzando le feature

[1] https://www.tensorflow.org/

imparate dalla rete nei nodi dei convolutional layer attraverso le attivazioni provocate dall'input sui nodi in questione. Tali tecniche verranno trattate nella sezione 2.1.3.

Per la visualizzazione dei pesi dei fully connected layer invece, la tecnica più utilizzata è di nuovo il Node-link diagram, dove i neuroni sono rappresentati come nodi e i pesi come i link tra i nodi. Questi ultimi possono venire codificati utilizzando linee più o meno spesse a seconda del valore del peso, oppure utilizzando diversi colori. In alcuni casi vengono aggiunte informazioni all'interno della visualizzazione del singolo nodo, oltre al valore di attivazione del neurone che è solitamente presente, come ad esempio una lista di immagini che lo attivano maggiormente.

Un esempio di visualizzazione di questo tipo è quella implementata in **TensorFlow Playground** [11] (figura 3), un progetto open-source online che, sulla base di alcuni dataset d'esempio prefissati ed utilizzando un framework per il Deep Learning creato ad hoc, permette di allenare una rete neurale più o meno complessa, lasciando libero l'utente di definirne il numero di layer, il numero di neuroni in ciascun layer, il learning rate, la funzione di attivazione e così via. Per quanto riguarda i pesi invece, è possibile modificarne in real-time il valore in

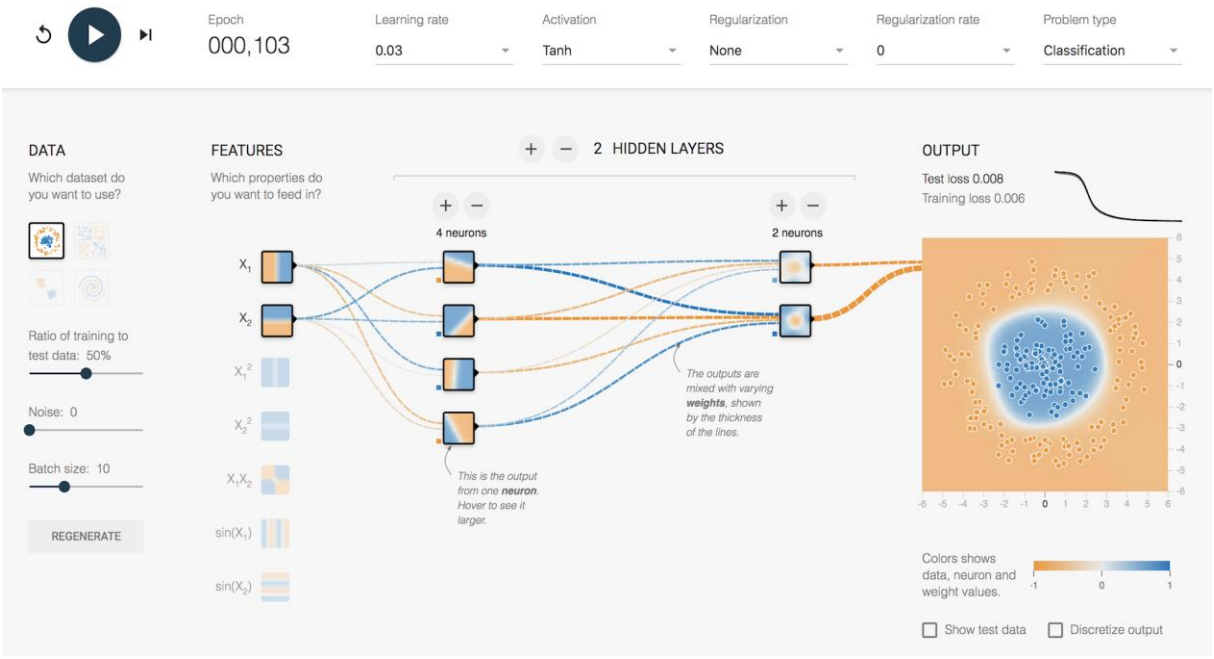

Figura 3 [10] – TensorFlow Playground

corrispondenza di qualsiasi connessione tra neuroni, lasciando all'utente molto spazio di sperimentazione. Tale strumento risulta molto utile per lo studente o per chi vuole approfondire la conoscenza dell'impatto che i vari iper-parametri, piuttosto che l'architettura del modello, hanno sulla rete stessa in termini di efficacia e prestazioni.

Un altro esempio è la visualizzazione sviluppata da **Harley** relativa ad una CNN per il riconoscimento di numeri scritti a mano sulla base del dataset MNIST [12] (figura 4). In questo ambiente, fruibile da browser web, è possibile inserire un numero scritto a mano in un apposito spazio e visualizzare in real-time il modello corrispondente al riconoscimento di tale numero. Questa visualizzazione è una sorta di Node-link diagram avanzato, in quanto il grafico è mostrato in uno spazio 3D, dove è possibile muoversi in ogni direzione. In particolare, vengono

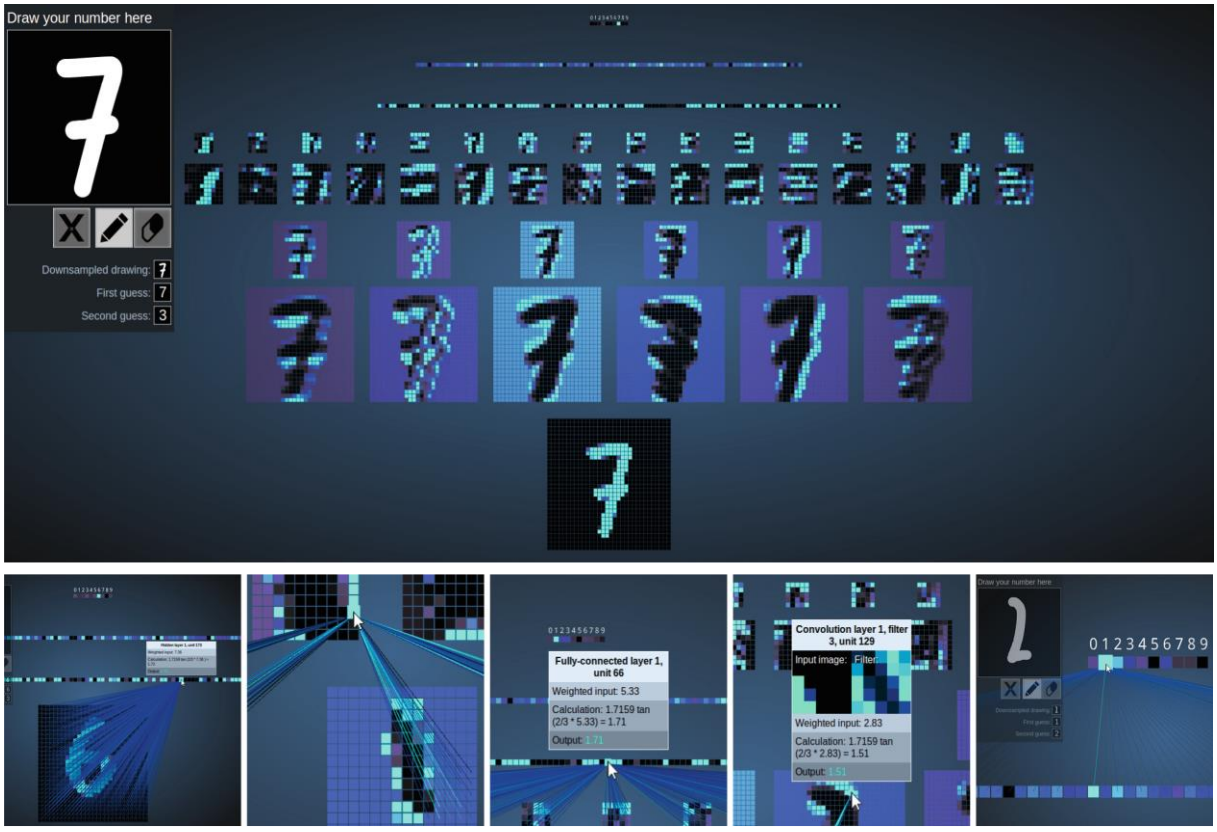

Figura 4 [12] – In alto è rappresentata la visualizzazione sviluppata da Harley sulla base del dataset MNIST della CNN che riconosce il numero 7 disegnato nel box in alto a sinistra. Al di sotto del box di input è anche possibile visualizzare la classificazione finale e la seconda scelta. In basso, da sinistra a destra sono rappresentati: gli archi mostrati passando con il mouse sopra un nodo di un fully connected layer, gli archi mostrati passando con il mouse sopra un nodo di un convolutional layer, i dettagli mostrati cliccando su un nodo di un fully connected layer, i dettagli mostrati cliccando su un nodo di un convolutional layer e infine le attivazioni dell'ultimo layer (quello che rappresenta la classificazione finale) per un input ambiguo.

mostrati di default solo i layer e i neuroni, dove i nodi dei layer convoluzionali sono rappresentati dalle feature corrispondenti, mentre i neuroni dei dense layer vengono visualizzati come punti più o meno colorati a seconda del valore di attivazione corrispondente. Per visualizzare i vari link tra i layer l'utente non deve fare altro che muoversi con il mouse sopra le rappresentazioni dei neuroni, selezionando il punto specifico da analizzare. Dunque, è possibile cliccare su tale punto per ottenere ulteriori dettagli, come la visualizzazione del filtro corrispondente in caso si stia analizzando un nodo di un layer convoluzionale, oppure il valore ottenuto dalla combinazione delle informazioni dal layer precedente oppure ancora l'output della funzione di attivazione.

2.1.3 Attivazioni, feature e gradienti

In generale, visualizzare le attivazioni può essere utile per comprendere come la rete risponde a particolari istanze di dati in ingresso (ovvero, ad esempio per una CNN, particolari immagini in input), e questo è ciò che viene chiamato **instance-level observation**. Una delle applicazioni di tale tipo di analisi riguarda l'utilizzo delle istanze come unità di test. Infatti, a seconda dei dati in ingresso, possono essere effettuate visualizzazioni ad hoc che risultano utili per capire perché un certo input produca un risultato corretto oppure sbagliato, e come si è arrivati a tale risultato.

Oltre ad effettuare analisi su singole istanze di ingresso, è possibile anche effettuare analisi su specifiche collezioni di istanze (**subset-level observation**), ad esempio considerando una serie di input tutti appartenenti alla stessa classe. Questo può essere utile per calcolare le attivazioni di tutte le istanze appartenenti alla collezione contemporaneamente, e dunque confrontarle per acquisire ulteriore conoscenza. Inoltre, è anche possibile considerare gruppi di collezioni di istanze ed effettuare un'analisi su di essi, ad esempio considerando contemporaneamente gruppi di collezioni appartenenti a classi differenti e dunque confrontare le attivazioni nei vari gruppi per provare a comprendere il processo decisionale della rete. La maggior parte degli studi su tecniche di questo tipo è stato svolto su dati testuali per le reti LSTM (Long Short-Term Me-

mory), ma alcuni tool, in grado di lavorare con diversi tipi di dati in input, possono essere utilizzati per effettuare tale tipo di analisi anche sulle immagini (ad esempio ActiVis [14], che verrà trattato nel paragrafo 2.3, che in particolare può lavorare con tutti i dati supportati da FBLearner, piattaforma per il Deep Learning sviluppata da Facebook).

Per quanto riguarda le CNN, le attivazioni più interessanti da visualizzare sono quelle dei nodi dei layer convoluzionali e dei dense layer. Nello specifico, quando si parla di visualizzazione delle attivazioni di un nodo di un convolutional layer, ci si riferisce soprattutto alla visualizzazione del risultato dell'applicazione del filtro di convoluzione relativo a quel nodo sulla rappresentazione in input derivante dal layer precedente, che insieme con la funzione di attivazione determina la feature appresa da quel neurone, ovvero ci si riferisce sostanzialmente alla visualizzazione della feature più che delle singole attivazioni. Visualizzare le feature è sicuramente tra gli argomenti di maggior interesse per quanto riguarda la comprensibilità delle CNN, ed è quello sul quale gli studi si sono maggiormente concentrati, producendo un certo numero di tecniche di visualizzazione. Inoltre, può essere utile considerare non un solo nodo alla volta ma un intero layer, in modo da visualizzare il tipo di feature che vengono apprese complessivamente: ad esempio un layer potrebbe risultare più sensibile agli oggetti piuttosto che ai colori o alle forme, e così via.

Parlando invece di visualizzazione di attivazioni di un dense layer, può avere senso rappresentare graficamente il semplice output della funzione di attivazione, attraverso un punto più o meno colorato, una barra più o meno piena, a seconda del suo valore in relazione a quelli degli altri nodi dello stesso layer. Tale valore però rappresenta come le feature provenienti dai convolutional layer siano state combinate in un certo nodo di un dense layer, quindi anche in questo caso la visualizzazione più interessante riguarda proprio le feature, ed in particolare come le feature rilevate ai layer convoluzionali precedenti vengano combinate ai nodi dei dense layer ai fini del risultato finale della rete.

Oltre alla funzione di attivazione, un'altra operazione che svolge un ruolo fondamentale per quanto riguarda l'allenamento di una rete neurale è il calcolo dei gradienti durante la fase di backpropagation. Come visto, questo algoritmo, attraverso tecniche di ottimizzazione, permette

di ripercuotere l'errore nell'output della rete indietro verso l'input in maniera efficiente, in modo da poter modificare opportunamente i parametri ad ogni layer (ovvero, principalmente, i valori che costituiscono i filtri convoluzionali e i pesi dei dense layer, per quanto riguarda le CNN) per produrre risultati finali sempre più precisi. Questo costituisce il cuore del processo di apprendimento di una rete neurale, ed in particolare in una CNN è grazie a questo meccanismo, ed ai gradienti, che i layer convoluzionali riescono a riconoscere le varie feature all'interno delle immagini di input, e che i dense layer sono in grado di assemblare correttamente tali feature per generare l'output finale.

Per tutti questi motivi, nel paragrafo 2.2 si approfondiranno le principali **tecniche di visualizzazione delle attivazioni**, alcune delle quali focalizzate sulla rappresentazione delle feature, altre basate principalmente sull'utilizzo dei gradienti, alcune incentrate sulle feature imparate da uno specifico nodo ed altre ancora che hanno come obiettivo quello di fornire una visione d'insieme delle feature apprese da un certo layer. Tutte insieme appartengono ad uno degli ambiti di maggior interesse riguardante la visualizzazione di CNN, e sono dunque anche centrali per il lavoro svolto in questo studio.

2.1.4 Embedding

Un altro aspetto della visualizzazione delle feature riguarda la rappresentazione di una visuale d'insieme relativa a tutta la rete, in particolare visualizzando la conoscenza appresa in ciascun layer. Poiché tipicamente le feature vengono rappresentate come vettori a molte dimensioni chiamati embedding, questo tipo di visualizzazione consiste principalmente nel proiettare tali embedding in uno spazio bi- o tri-dimensionale, **utilizzando tecniche di riduzione di dimensionalità**: due tra le più utilizzate sono Principal Component Analysis (PCA) e t-distributed Stochastic Neighbor Embedding (t-SNE) [13]. In questo modo è possibile evidenziare visivamente eventuali relazioni tra le feature proiettate, utili per capire come la rete stia assimilando le informazioni.

È possibile utilizzare le tecniche di riduzione di dimensionalità anche in altri modi, ad esempio per concentrarsi maggiormente su come le istanze di ingresso vengano trasformate durante il passaggio attraverso i vari layer della rete neurale [14], oppure per visualizzare come gli embedding cambino nel tempo, per valutarne la qualità durante la fase di allenamento [15]. Ad ogni modo, in questo ambito molto dipende dalla tecnica di riduzione di dimensionalità utilizzata, e la ricerca oggi è dunque maggiormente focalizzata sullo sviluppo di nuove tecniche di questo tipo, piuttosto che sul tipo di visualizzazione da adottare per rappresentare gli embedding, che risulta simile in tutti i tool che la rendono disponibile (ad esempio, vedi la figura 29 relativa a **Embedding Projector** [34], nel paragrafo 2.3).

2.1.5 Metriche

Poiché tipicamente i modelli sono piuttosto grandi e complessi, il metodo più semplice per avere una visione d'insieme sull'efficacia e le prestazioni di una rete neurale è la visualizzazione di grafici e statistiche basati su una serie di metriche. Alcune di queste possono essere la perdita (l'errore nell'output), l'accuratezza e svariate misure di errore, che ad esempio vengono calcolate e mostrate automaticamente alla fine di ogni **epoca** (l'unità temporale durante la fase di apprendimento di una rete neurale) per mostrare l'andamento della fase di training. La visualizzazione di tali metriche è fondamentale anche per quanto riguarda il confronto tra diversi modelli contemporaneamente, e ciò avviene tipicamente rappresentando i dati in grafici a linee o istogrammi.

2.2 Tecniche di visualizzazione delle attivazioni

In questo paragrafo verranno approfondite le più importanti tecniche di visualizzazione delle attivazioni, oggi al centro dell'attenzione per quanto riguarda la comprensione delle reti neurali, e per questo motivo al centro di questo lavoro. In particolare, si farà riferimento al problema di

classificazione, uno dei problemi tipici risolti dalle reti neurali, e alle CNN. La trattazione è suddivisa in sezioni, ciascuna delle quali comprende tecniche affini in base al significato dell'output prodotto dalla tecnica stessa ai fini della comprensione della CNN. In particolare si parlerà di tecniche di visualizzazione che mostrano quali sono i pattern nelle immagini in input che causano le attivazioni maggiori ad un certo nodo della rete (2.2.1), quindi si analizzeranno le tecniche in grado di evidenziare le regioni che nelle immagini in input risultano discriminanti per una certa classe (2.2.2), dunque i metodi che permettono di visualizzare la rilevanza di ogni pixel delle immagini in input relativamente ad una certa classe (2.2.3). Infine, si tratteranno le tecniche in grado di mostrare le aree di attenzione delle immagini in input sulle quali le CNN si focalizzano per la classificazione finale (2.2.4), e per ultimi i metodi che mostrano le feature imparate ad un determinato layer di una rete, a prescindere dalle immagini in input (2.2.5).

2.2.1 Pattern in input che causano attivazioni maggiori ad un certo nodo

I gradienti possono essere visualizzati attraverso quelle che vengono chiamate **sensitivity/saliency maps**, ovvero rappresentazioni costituite da heatmap sovrapponibili all'immagine di input che si sta analizzando, in modo che i pixel più importanti in essa ai fini della classificazione finale risultino evidenziati. In questo caso, l'"importanza di un pixel" sarà tanto maggiore quanto è minore il cambiamento necessario in quel particolare pixel dell'immagine di input per far variare maggiormente il risultato finale della classificazione.

Uno dei primi approcci in questo senso è stato il metodo di deconvoluzione [16] (figura 5) dove, al posto del calcolo dei gradienti, viene realizzata una **DeconvNet** per propagare all'indietro le attivazioni ad un certo nodo di un layer convoluzionale, in modo da rilevare i pattern che nell'immagine in input causano le attivazioni maggiori a quel nodo. DeconvNet risulta una sorta di rete inversa, dove i pesi sono presi dalla CNN principale e trasposti (quindi non necessita di nessun allenamento), ed ogni layer è inverso rispetto a quello della rete principale: in particolare, sono fondamentali per questo approccio i Max Pooling layer. Durante il passaggio forward

nella rete principale, oltre ad effettuare l'operazione di down-sampling della rappresentazione dell'immagine in input, i Max Pooling layer vengono utilizzati anche per popolare i cosiddetti switch. Tali switch non sono altro che strutture dati che contengono le posizioni delle attivazioni maggiori ai vari nodi convoluzionali: questi vengono utilizzati dalla DeconvNet nell'operazione di up-sampling per poter ricostruire l'immagine in input evidenziando appunto i pattern che in essa corrispondono alle attivazioni maggiori. In assenza delle informazioni contenute negli switch, ovvero in CNN senza Max Pooling layer, questa tecnica diventa meno affidabile

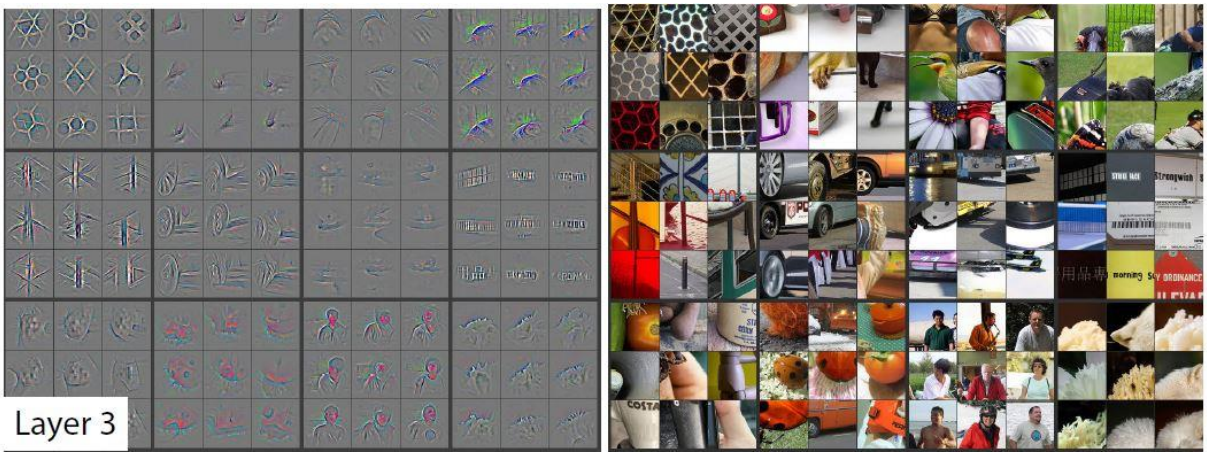

Figura 5 [16] – Esempio di visualizzazione generata da DeconvNet: un sottoinsieme casuale delle feature estratte al layer 3 del modello utilizzato in [16] dove, per ciascuna di esse, vengono mostrate le rappresentazioni generate dal metodo di convoluzione corrispondenti alle 9 attivazioni maggiori. A destra vengono inoltre mostrate le 9 immagini in input corrispondenti alle attivazioni maggiori delle feature visualizzate.

Figura 6 [17] – Esempio di visualizzazione prodotta dal metodo Gradient: saliency map relative alle top-1 prediction per alcune immagini di test selezionate casualmente in [17] dal dataset ILSVRC-2013.

in quanto la visualizzazione risultante non sarà più sufficientemente legata all'immagine in input, e la sensitivity map in output alla DeconvNet mostrerà solamente una sorta di pattern medio che causa le attivazioni maggiori al nodo in esame.

Rientra in questa categoria il metodo illustrato in [17] (figura 6), solitamente chiamato semplicemente **Gradient** o **Saliency** poiché, utilizzando i gradienti, genera appunto una gradient-based saliency map. Questa tecnica è considerata una generalizzazione rispetto a DeconvNet, in quanto basandosi solamente sul gradiente, non risulta più vincolata alla presenza di Max Pooling layer, e può così essere applicata a molti più tipi di CNN. DeconvNet però, con l'utilizzo di una vera e propria struttura inversa per la generazione della saliency map riesce a produrre mappe più definite (in particolare grazie all'utilizzo di ReLU all'interno della DeconvNet, che azzera i contributi di attivazione negativi provenienti dai top layer della CNN di partenza). Inoltre, le feature visualizzate in DeconvNet risultano più globali, essendo calcolate utilizzando tutta la rete inversa, rispetto a quelle gradient-based, in quanto queste ultime sono in effetti basate esclusivamente sul gradiente delle attivazioni in un certo nodo rispetto all'immagine di input (dunque locali a quel nodo).

Per migliorare la visualizzazione delle sensitivity map (quelle prodotte dai due metodi fino ad ora descritti, ma anche qualsiasi altro tipo di sensitivity map) è stata sviluppata una tecnica chiamata **SmoothGrad**[2] [18] (figura 7). In particolare, questa prevede l'introduzione di piccole perturbazioni nell'immagine di input in modo da generare un set di campioni da utilizzare per la visualizzazione finale, che corrisponderà ad una media delle saliency map relative ad ogni campione. Inoltre, ulteriori miglioramenti vengono ottenuti inserendo rumore casuale durante la fase di allenamento della rete.

Guided Backpropagation [19] (figura 8), estende il metodo di deconvoluzione, combinandolo con la tecnica basata sui gradienti descritta in [17], risultando in visualizzazioni ben più definite. In particolare, con questa tecnica vengono isolati al nodo in esame sia i contributi di attivazione negativi per una certa classe provenienti dai top layer della CNN (grazie all'applicazione di

[2] https://pair-code.github.io/saliency/

ReLU come in DeconvNet), sia i contributivi negativi provenienti dai layer precedenti (attraverso la fase di backpropagation basata sui gradienti). Come in [17] dunque, il metodo può essere applicato su un gran numero di CNN non essendo vincolato alla presenza dei Max Pooling layer. Inoltre, come in [16], vengono considerate maggiormente le feature globali rispetto a quelle locali ai singoli nodi.

Figura 7 (https://pair-code.github.io/saliency/) – Esempio di visualizzazioni generate da SmoothGrad. In particolare, a destra vengono mostrate, in alto, le heatmap generate rispettivamente dal metodo Gradient [17], Integrated Gradients [39] e Guided Backpropagation [19] utilizzando il modello InceptionV3 sull'immagine a sinistra, proveniente dal dataset di ImageNet. In basso a destra invece, sono presenti le relative rappresentazioni ottimizzate utilizzando SmoothGrad

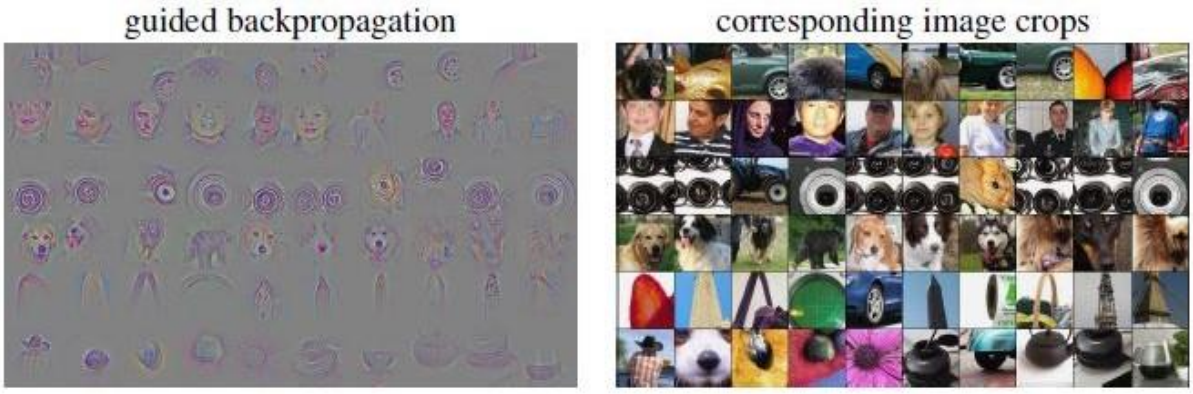

Figura 8 [19] – Esempio di visualizzazione generata da Guided Backpropagation. In particolare, vengono visualizzati (a sinistra) i pattern corrispondenti alle 10 immagini che causano le attivazioni maggiori a ciascun filtro (mostrate a destra), appresi al layer 9 della rete utilizzata in [19] e allenata su ImageNet. Ogni riga corrisponde ad un filtro.

2.2.2 Regioni di immagini in input discriminanti per una certa classe

In [16] viene effettuato un primo studio per quanto riguarda l'estrazione di regioni discriminanti in un'immagine di input per una certa classe, valutando come la classificazione finale della CNN cambi in relazione alla parte occlusa attraverso un box grigio nell'immagine di input: questa tecnica viene infatti denominata **Grey-box occlusion**, o semplicemente Occlusion (figura 9). Pur ottenendo risultati interessanti, questo metodo si rivela molto semplice ed inefficiente in quanto si basa su perturbazioni ripetute dell'immagine in input, richiedendo dunque un passaggio forward nella rete per ciascuna perturbazione. Inoltre, soffre di alcuni problemi come il riconoscimento da parte della rete dei bordi del box grigio, che potrebbe portare a risultati inaspettati. Come per i metodi della sezione 2.2.1 dunque, i risultati migliori e computazionalmente più efficienti si hanno con tecniche basate sulla backpropagation, come le seguenti.

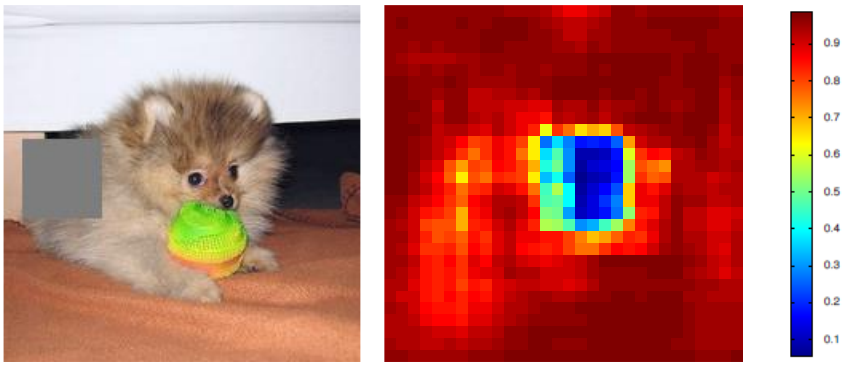

Figura 9 [16] – Esempio di visualizzazione generata dal metodo Grey-box occlusion. A sinsitra viene mostrato uno stadio della perturbazione dove una parte dell'immagine di input è coperta dal box grigio, mentre a destra viene rappresentata la probabilità che la rete utilizzata per classificazione dell'immagine in input in [16] sia ancora in grado di fornire in output la classe corretta in base alla posizione del box grigio, ovvero in base alla perturbazione effettuata.

CAM-GAP[3] [20] (figura 10), Class Activation Mapping con Global Average Pooling layer, è una tecnica di questo tipo che sfrutta appunto i GAP layer. Questi consistono nell'applicazione dell'operazione di media alle feature riconosciute da ogni nodo di un layer convoluzionale, che deve necessariamente precederlo. Posizionandoli in coda all'ultimo layer convoluzionale di una rete che risolve un problema di classificazione, i GAP layer fungono da regolarizzatori strutturali, andando a generare una feature media che risulta così rappresentativa di ogni classe [49]. In CAM-GAP quindi, eventuali fully connected layer in coda all'ultimo layer convoluzionale

[3] http://cnnlocalization.csail.mit.edu/

saranno rimossi e la rete dovrà invece terminare con un GAP layer seguito da un ultimo dense layer che corrisponde al SoftMax per ottenere la classificazione finale. La visualizzazione risultante corrisponderà dunque alla rappresentazione della media spaziale delle feature imparate nell'ultimo layer convoluzionale a partire da un'immagine in input, relative ad una certa classe, che può essere quella predetta dalla CNN per quell'immagine, ma anche qualsiasi altra tra quelle che la rete è in grado di riconoscere. L'utilizzo di questa tecnica, per via del particolare requisito riguardante gli ultimi layer, implica una modifica alla maggior parte delle CNN, e dunque un re-training delle stesse, che oltretutto influenza l'efficacia della classificazione.

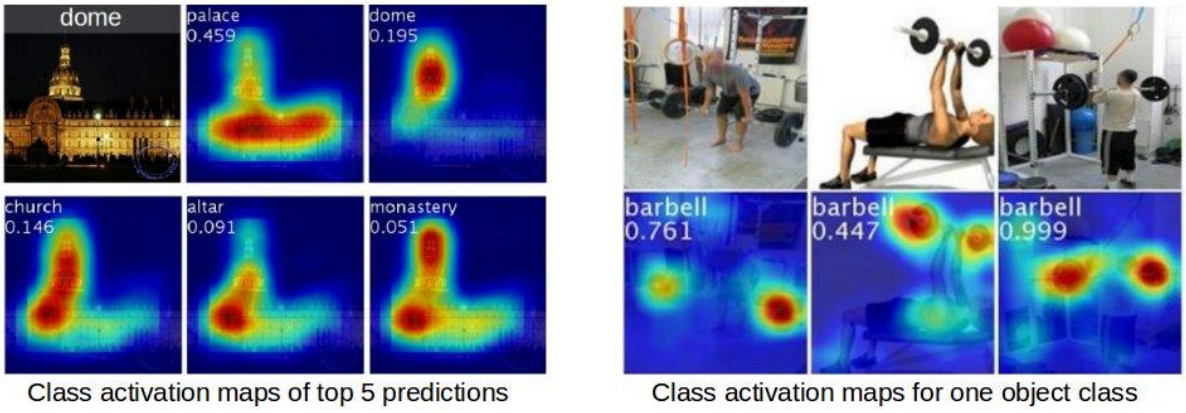

Figura 10 (http://cnnlocalization.csail.mit.edu/) – Esempio di visualizzazioni generate da CAM-GAP. A sinistra, per un'immagine di input con classe "dome", sono mostrate le rappresentazioni CAM-GAP corrispondenti alle top-5 previsioni della CNN utilizzata in [20], con i relativi punteggi. A destra vengono visualizzate le mappe relative allo stesso oggetto (stessa classe) in tre immagini differenti, con i relativi punteggi della classificazione.

Figura 11 – Esempio di visualizzazioni generate da GradCAM e Guided GradCAM. A sinistra viene mostrata la visualizzazione generata da GradCAM per la classe "boxer" del dataset ImageNet, utilizzando una rete VGG16. Al centro viene mostrata la relativa heatmap generata da Guided Backpropagation e infine a destra il risultato della moltiplicazione delle due mappe, ovvero l'output del metodo Guided GradCAM.

GradCAM[4] [21] (figura 11), Gradient-weighted Class Activation Mapping, fornisce lo stesso tipo di visualizzazione fornito da CAM, senza però richiedere né modifiche né re-training della CNN da visualizzare. Questo perché, al posto dei GAP layer, vengono utilizzati i gradienti relativi alla classe in esame rispetto al layer che si vuole visualizzare: in questo modo dunque è possibile non solo visualizzare le regioni discriminative per una certa classe all'ultimo layer convoluzionale, ma in qualsiasi layer convoluzionale. Ovviamente le visualizzazioni più interessanti si hanno verso il fondo della rete, dunque, così come in CAM, le mappe generate da questa tecnica risultano poco definite, in quanto i layer convoluzionali più profondi sono quelli che contengono i filtri più piccoli. Per migliorare la visualizzazione, GradCAM viene utilizzato in combinazione con tecniche più definite, come DeconvNet [16] e Guided Backpropagation [19], che da sole non sono in grado di creare lo stesso tipo di rappresentazione: per fare ciò, le mappe generate da GradCAM vengono up-scalate in modo da poter essere moltiplicate punto a punto con la sensitivity map prodotta dal metodo prescelto. Una delle combinazioni migliori risulta essere quella con Guided Backpropagation, chiamata **Guided GradCAM** [21] (figura 11).

2.2.3 Rilevanza dei pixel di immagini in input per una certa classe

I metodi visti finora mostrano, in maniere differenti, come i pixel di un'immagine determinino la classificazione finale: nella sezione 2.2.1 sono elencate tecniche che visualizzano i pixel che necessitano delle modiche minori per cambiare maggiormente il risultato della classificazione, mentre la sezione 2.2.2 contiene metodi che mostrano i pixel più salienti in un'immagine di input che determinano una certa classe in output. In generale questi metodi mostrano quali pixel hanno determinato maggiormente il risultato, risolvendo il problema di ottimizzazione della ricostruzione dell'immagine.

[4] http://gradcam.cloudcv.org/

LRP [22], Layer-wise Relevance Propagation, invece, è una tecnica in grado di generare una cosiddetta relevance map, che evidenzia il contributo di ogni pixel dell'immagine di input per la classificazione finale, che sia esso positivo o negativo. In questo caso dunque, l'obiettivo è la ricostruzione della decisione di classificazione, ovvero mostrare perché il risultato finale è stato prodotto a partire dai pixel dell'immagine in input: tecniche simili sono dette tecniche di tipo **attribution**, che a differenza dei metodi visti in precedenza, di tipo saliency, mostrano la rilevanza di ciascun pixel dell'immagine ai fini della classificazione e non semplicemente quali sono i pattern o le aree dell'immagine che hanno influenzato maggiormente il risultato. In particolare, LRP propaga all'indietro i risultati della classificazione della CNN per una certa immagine in input, utilizzando uno schema a messaggi basato sul principio di conservazione della rilevanza, per il quale la rilevanza totale in output alla rete può essere completamente redistribuita attraverso i suoi layer, fino agli input.

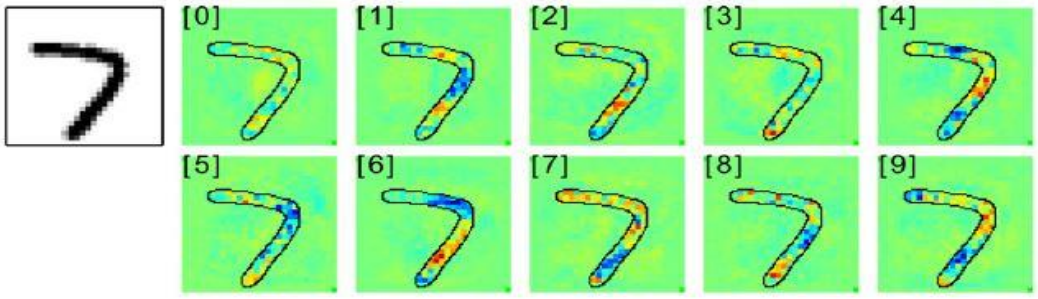

Figura 12 [22] – Esempio di visualizzazioni generate da LRP. In particolare, sono mostrate le visualizzazioni generate dal metodo LRP-z, per la rete neurale definita da [22] e allenata sul dataset MNIST, relativamente al numero indicato fra parentesi quadre. Le parti tendenti al blu corrispondono a rilevanza negativa rispetto alla classificazione, mentre le parti tendenti al rosso corrispondono a rilevanza positiva.

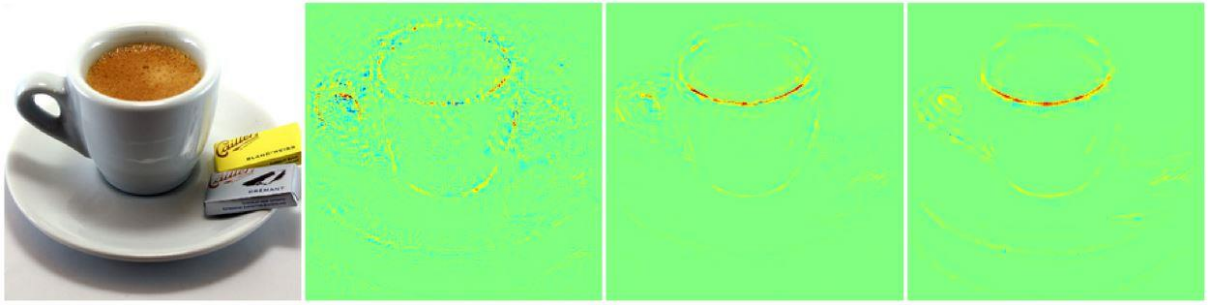

Figura 13 [22] – Esempio di visualizzazioni generate dal metodo LRP-ε e LRP-αβ. In particolare, la prima immagine a sinistra rappresenta l'input, la seconda e la terza rappresentano le visualizzazioni generate da LRP-ε rispettivamente con ε=0,01 e ε=100, mentre l'ultima immagine, a destra, è stata generata utilizzando LRP-αβ con α=2 e β=1. Tutte le visualizzazioni sono state generate a partire dalla rete neurale definita in [22] e allenata sul dataset ILSVRC.

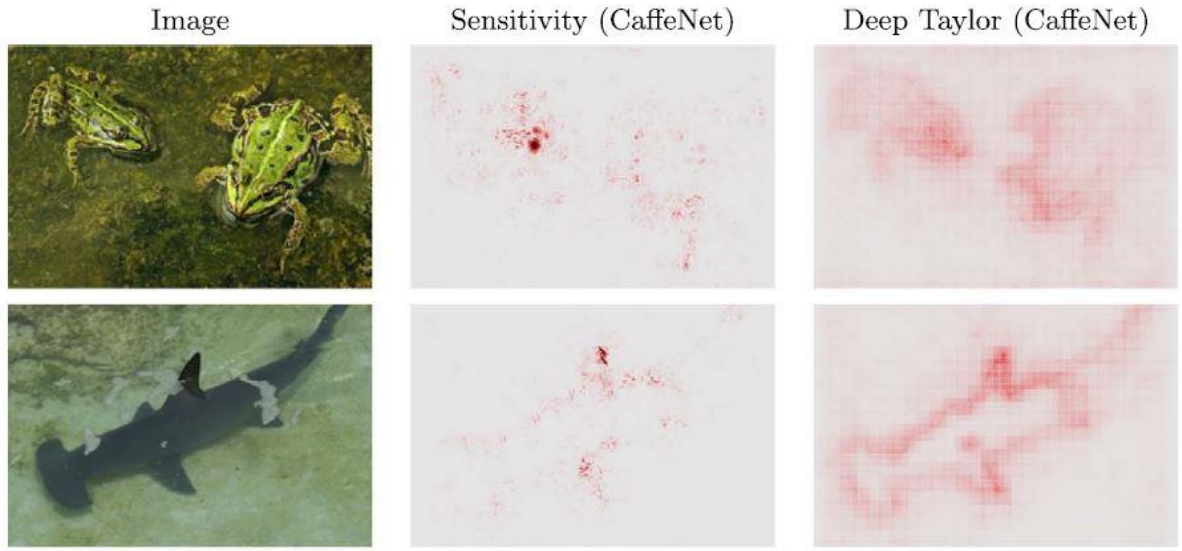

Figura 14 [37] – Esempio di visualizzazioni generate dal metodo DeepTaylor. A sinistra vengono mostrate due immagini di input appartenenti al dataset ILSVRC e a destra il relativo output del metodo DeepTaylor utilizzando la rete CaffeNet, messo a confronto con la sensitivity map generata dal metodo Gradient [17] (al centro).

In realtà, LRP non definisce una tecnica specifica ma una serie di vincoli per la redistribuzione della rilevanza tali per cui tutti gli algoritmi che li soddisfano possono essere definiti tecniche LRP: tra le principali esistono LRP-z (figura 12) ovvero la prima proposta di LRP in [22], LRP-ε (figura 13) dove viene utilizzato un termine ε per risolvere problemi numerici di LRP-z, LRP-αβ (figura 13) che separa i termini positivi (α) da quelli negativi (β) nella formula di redistribuzione della rilevanza [23] (LRP-z può infatti essere visto come un caso particolare di LRP-αβ, dove α vale 1 e β vale 0). La tecnica di LRP base (LRP-z) è inoltre paragonabile ad un altro metodo di visualizzazione, che dunque rientra in questa categoria pur essendo molto semplice e basato su Gradient [17] visto prima, nella sezione 2.2.1. Infatti, a meno di un fattore di scala, LRP-z risulta equivalente alla tecnica cosiddetta Input x Gradient.

Come suggerito dal nome, **Input x Gradient** (figura 16) prevede la moltiplicazione tra la saliency map generata dal metodo Gradient [17] con l'immagine di input. Questa operazione molto semplice permette di ottenere un risultato interessante: in questo modo infatti ai pixel della heatmap generata da Gradient viene attribuito un valore di importanza in base al valore dei pixel corrispondenti dell'immagine in input. Le heatmap generate da questo metodo risultano così attribution map, più significative rispetto alle saliency map del metodo Gradient, in

quanto contengono anche informazioni sul segno e sulla forza dei pixel dell'immagine in input [38].

Sulla base del metodo basato sulla decomposizione di Taylor, esplorato in [22] come base per LRP, è stata sviluppata **DeepTaylor** [37] (figura 14), una tecnica dove ogni neurone viene visto come una funzione che può essere scomposta sui suoi input e dunque aggregata con altre funzioni per creare il risultato finale della rete neurale di cui fa parte. In particolare, l'output viene suddiviso, layer per layer, attraverso i neuroni della rete, in funzioni e sotto-funzioni che possono dunque anche essere propagate all'indietro per assegnare a ciascun nodo la sua rilevanza per l'output finale, sulla base degli altri neuroni con cui è connesso. Per tenere conto anche del contributo che le attivazioni di un certo neurone hanno sulla rilevanza di un neurone di un layer successivo, viene utilizzato un modello di rilevanza, allenabile o meno, che determina proprio come tali attivazioni vengono distribuite.

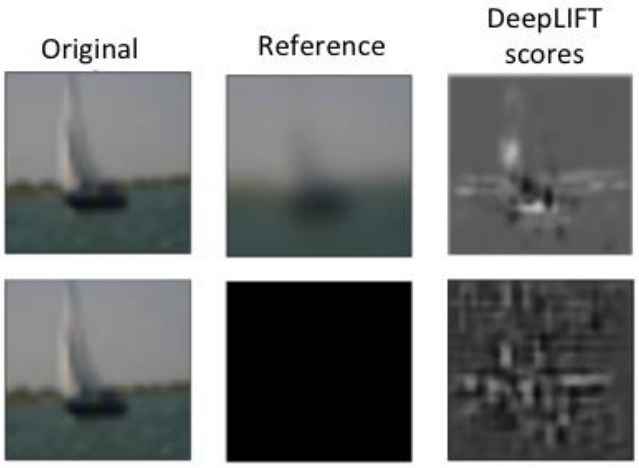

Figura 15 [38 – Appendix L] – Esempio di visualizzazione generata dal metodo DeepLIFT. In particolare, viene mostrato come (in alto) l'utilizzo di un input di riferimento scelto opportunamente in base al dominio del problema risolto dalla rete, produca visualizzazioni migliori rispetto all'utilizzo di un input di riferimento "standard" (in basso).

Un altro metodo basato sulla propagazione all'indietro della rilevanza di un certo output sugli input, è **DeepLIFT** [38] (figura 15). La rilevanza viene assegnata ai pixel dell'immagine di input sulla base del confronto con un input di riferimento diverso a seconda del problema, utilizzando alcune regole. In questo modo è possibile assegnare un valore di importanza anche nei casi in cui non fosse possibile basandosi solo sulle informazioni del gradiente, ad esempio a causa della sua discontinuità (un minimo cambiamento dell'input può corrispondere a grandi variazioni improvvise nel relativo gradiente) o anche semplicemente quando questo si annulla, azzerando tutte le informazioni provenienti dai neuroni corrispondenti. Fondamentale importanza ricade dunque sulla scelta dell'input di riferimento: per una CNN, un'immagine di

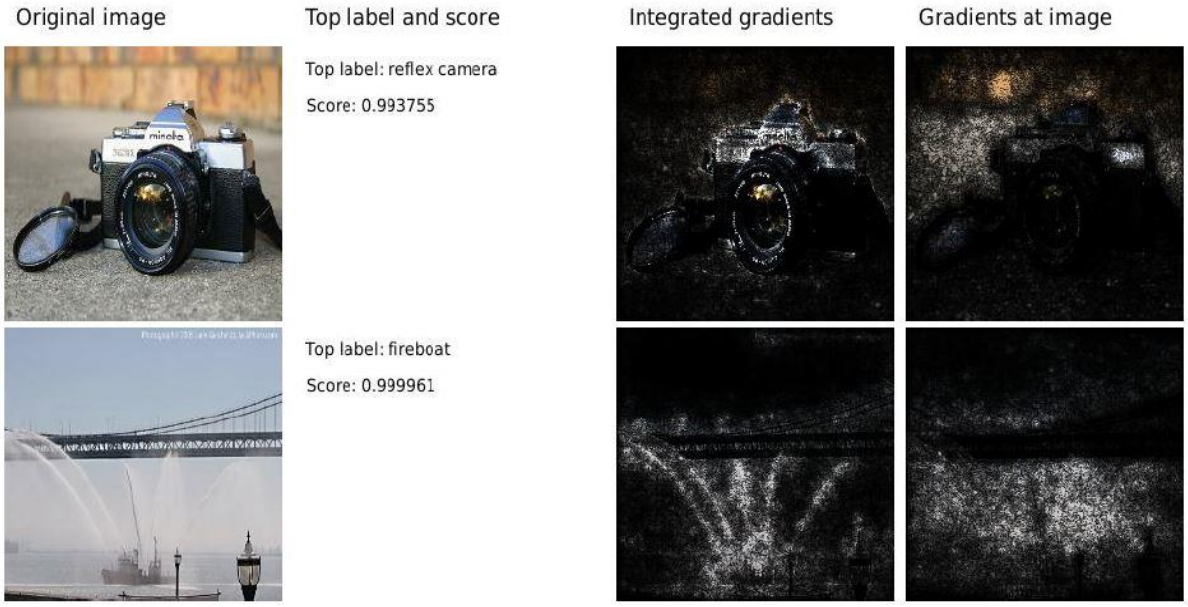

Figura 16 [39] – Esempio di visualizzazioni prodotte dai metodi Integrated Gradients e Input x Gradient. A sinistra vengono mostrate le immagini originali e i risultati della classificazione riportati in [39], a destra le heatmap relative ai due metodi.

riferimento formata semplicemente da tutti i pixel a 0 può andare bene in alcuni casi (ad esempio con il dataset MNIST, dove lo sfondo di ogni immagine è proprio composto da soli pixel a 0 [38]), ma la conoscenza del dominio è fondamentale per una scelta funzionale (ad esempio in [38], si è visto che con il dataset CIFAR10 i risultati migliori si hanno utilizzando come riferimento una versione sfocata dell'immagine in input che si sta analizzando). Inoltre, come per LRP, è possibile distinguere la rilevanza positiva da quella negativa nella visualizzazione prodotta.

Integrated Gradients [39] (figura 16) è invece una tecnica sviluppata a partire dal presupposto che è difficile creare tecniche di visualizzazione di tipo attribution, come quelle descritte in questa sezione, in maniera efficace, e questo perché risulta difficile valutarle. In questo caso quindi, si è partiti definendo due assiomi che, se rispettati, caratterizzerebbero una tecnica di visualizzazione attribution efficace: sensibilità, e invarianza di implementazione. Il primo determina che se un input possiede una feature differente da un input di riferimento e i due input corrispondono a previsioni diverse, allora la rilevanza di questa feature non può essere 0. Il secondo invece determina che se due reti sono funzionalmente equivalenti, cioè producono gli stessi output per tutti gli input possibili, qualsiasi sia la loro implementazione, allora devono

produrre sempre visualizzazioni equivalenti. Integrated Gradients è stato sviluppato in modo da soddisfare entrambi gli assiomi (mentre altre tecniche attribution ne soddisfano al massimo uno) e questo è possibile grazie all'utilizzo di un integrale sui gradienti, che permette di avere un'informazione discreta a partire da un punto di partenza, e non solo l'informazione istantanea del gradiente del punto in esame: il calcolo dell'integrale può però risultare computazionalmente molto pesante.

In [40] si osserva che il gradiente, elemento alla base delle tecniche di visualizzazione che utilizzano la backpropagation (come DeconvNet, Guided Backpropagation o Input x Gradient), in effetti non rappresenta il segnale caratteristico di una rete, ma piuttosto corrisponde alla relazione che c'è, all'interno dei dati di input, tra il segnale e il rumore che causa "distrazione"

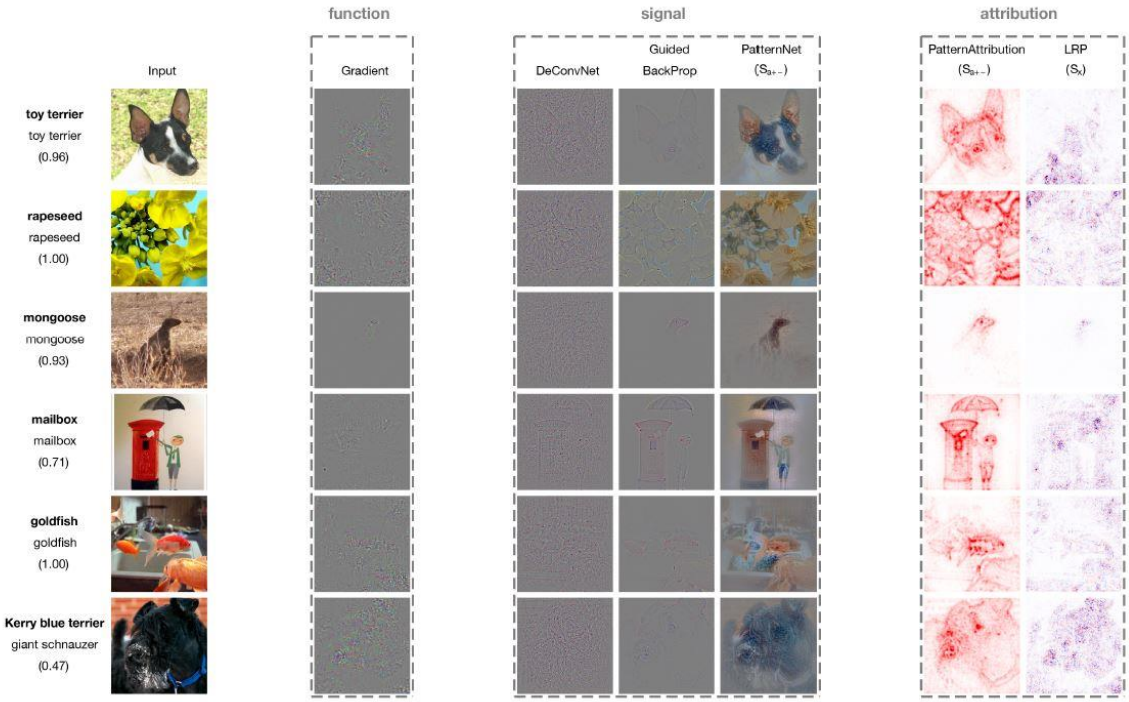

Figura 17 [40] – Esempio di visualizzazioni generate da PatternNet e PatternAttribution. Le immagini di input (a sinistra) sono prese casualmente dal validation set di ImageNet (insieme ai rispettivi punteggi di classificazione come in [40]): per ciascuna di queste immagini vengono mostrare visualizzazioni di tre tipi. La visualizzazione di tipo "function" corrisponde alla tecnica Gradient, che visualizza come il cambiamento di una feature all'ingresso di una rete cambierebbe l'output, poiché questo corrisponde all'analisi delle operazioni che la rete usa per calcolare l'output (quindi alla funzione della rete). Sono visualizzazioni di tipo "signal" sia DeconvNet, sia Guided Backpropagation, sia PatterNet appunto, perché permettono di analizzare le componenti dell'input che provocano un determinato output, ovvero il segnale della rete. Infine, PatternAttribution rientra tra i metodi di tipo "attribution", che attribuiscono importanza a determinate feature in input per l'output generato, e viene qui comparato (a destra) con LRP. [46]

nella rete. Viene quindi spiegata la necessità di utilizzare degli stimatori del segnale della rete, che dunque richiedono di essere allenati adeguatamente proprio per poter distinguere tale segnale in maniera chiara nell'input, isolandolo dai "distrattori". Dunque, vengono presentate due tecniche di visualizzazione: la prima, di tipo attribution, utilizza il segnale stimato per produrre una rappresentazione che risulta come un miglioramento di LRP, in quanto è in grado di proiettare all'indietro nella rete la rilevanza in maniera più precisa, avendo privato efficacemente il segnale dal rumore, chiamata **PatternAttribution** (figura 17). La seconda è chiamata **PatternNet** (figura 17), e rientra nella categoria alla quale appartengono DeconvNet e Guided Backpropagation (sezione 2.2.1), poiché in grado di visualizzare i pattern delle immagini in input che causano le attivazioni maggiori ad un nodo, utilizzando però per la backpropagation il segnale ripulito dai "distrattori", grazie all'utilizzo del segnale stimato. In questo modo, la rappresentazione ottenuta sarà molto più definita rispetto a quelle prodotte dal metodo DeconvNet ad esempio, proprio perché il segnale utilizzato per la sua generazione viene ripulito dal rumore.

Prediction Difference [24] (figura 18) invece, estende il meccanismo di occlusione dell'immagine studiato in [16], generando una relevance map con evidenziate le regioni dell'immagine

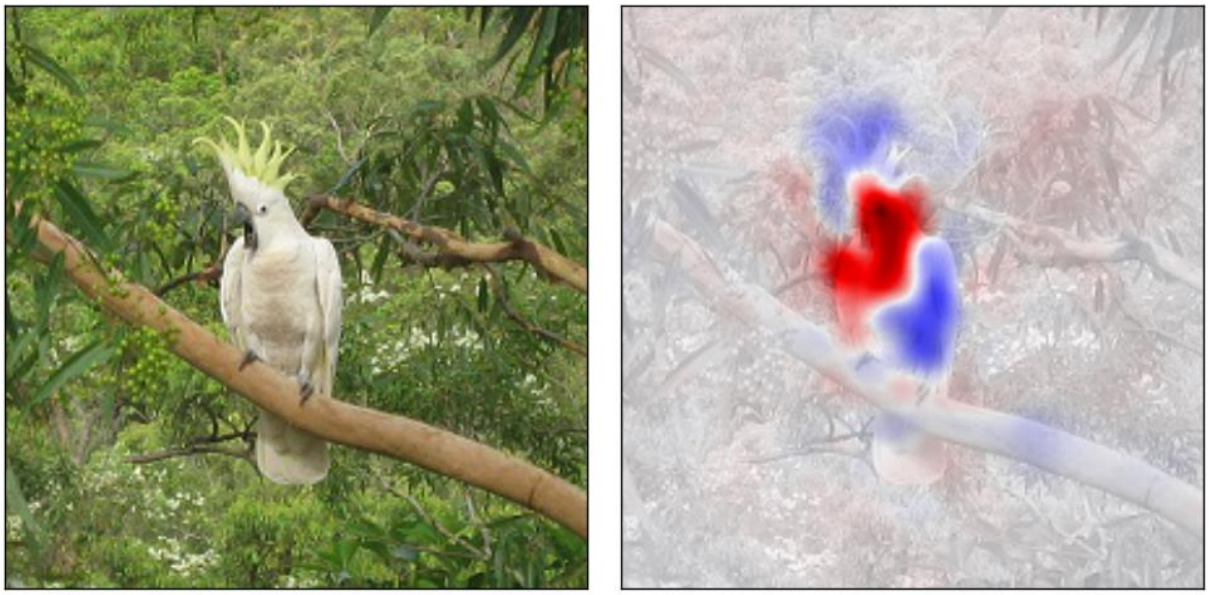

Figura 18 [24] – Esempio di visualizzazione generata dal metodo Prediction Difference. L'immagine in input viene classificata come "cockatoo" (cacatua) dalla rete GoogleNet e la visualizzazione mostra, in rosso, le parti che hanno condizionato positivamente il risultato finale, e in blu le parti contro la classe in output. In particolare, in questo caso, le parti del corpo del cacatua vengono considerate dalla rete come prove per la classe che risulta seconda nelle previsioni, ovvero "white wolf" [24].

in input alla CNN che forniscono prove a favore o a sfavore di una determinata classe per il nodo in esame. La relevance map viene creata basandosi su valori di rilevanza assegnati ad ogni feature dell'immagine in input: tali valori vengono calcolati simulando la mancanza della feature in questione e misurando come la previsione al nodo cambierebbe di conseguenza (considerando le feature dipendenti l'una dall'altra solo in un intorno limitato delle stesse, per non far esplodere i calcoli). Risultati migliori si ottengono effettuando le stesse misure considerando la mancanza di più feature alla volta, mediando poi la rilevanza finale di ogni feature. Questa tecnica risulta dunque perturbation-based, proprio come Occlusion [16], ed in effetti richiede una mole di calcoli non indifferente per generare la visualizzazione finale, il che la rende piuttosto lenta.

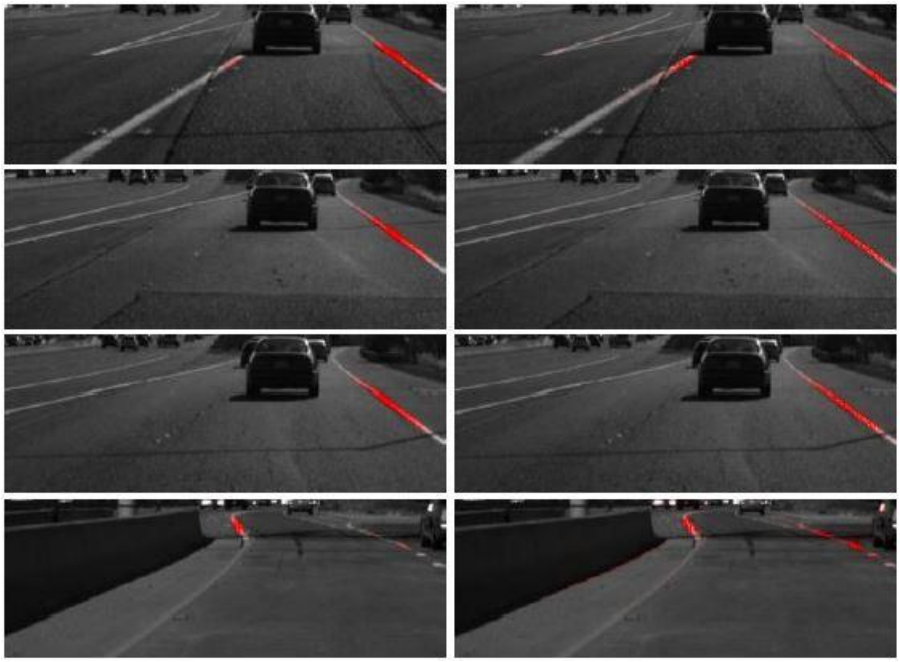

Figura 19 [25] – Esempio di visualizzazione generata da VisualBackProp (a sinistra) in confronto a quella generata da LRP (a destra) per una rete NetHVF [25]. Nonostante VisualBackProp sia molto più veloce di LRP, i risultati sono del tutto paragonabili.

VisualBackProp [25] (figura 19), infine, è una tecnica che fornisce risultati simili a quelli prodotti da LRP ma in maniera nettamente più veloce, in quanto è in grado di lavorare in real-time, risultando utilizzabile sia in fase di training sia in fase di inferenza. In particolare, questo metodo si basa sulla constatazione che gli ultimi layer di una CNN contengono le informazioni più rilevanti per la classificazione, ma che i primi layer sono quelli a risoluzione maggiore: dunque, vengono combinate le feature dell'ultimo layer convoluzionale con quelle del primo layer convoluzionale, tramite una fase di backpropagation basata sui valori (e non sui gradienti, dunque più veloce).

2.2.4 Aree di attenzione della CNN su immagini in input

Salient Relevance Mapping [26] (figura 20), è una tecnica basata sui modelli biologici cosiddetti di visual saliency, ovvero rappresentazioni derivate da studi neurologici su come le informazioni vengono processate nel cervello umano. Uno dei modelli di visual saliency principali per quanto riguarda l'estrazione di oggenti salienti e dei loro dintorni all'interno di un'immagine è il metodo di context-aware saliency (che non verrà approfondito in quanto esula dagli obiettivi di questo lavoro). Salient Relevance Mapping applica questo modello a partire da una relevance map creata tramite LRP per generare la cosiddetta SR map, che è in grado di mostrare le aree di maggior attenzione di una CNN all'interno di un'immagine di input. Questa tecnica si differenzia dunque da tutte le altre in quanto non ha come obiettivo quello di evidenziare aree di un'immagine importanti per una determinata classe, ma piuttosto mostrare gli oggetti che la CNN ha rilevato maggiormente, senza considerare una classe in particolare.

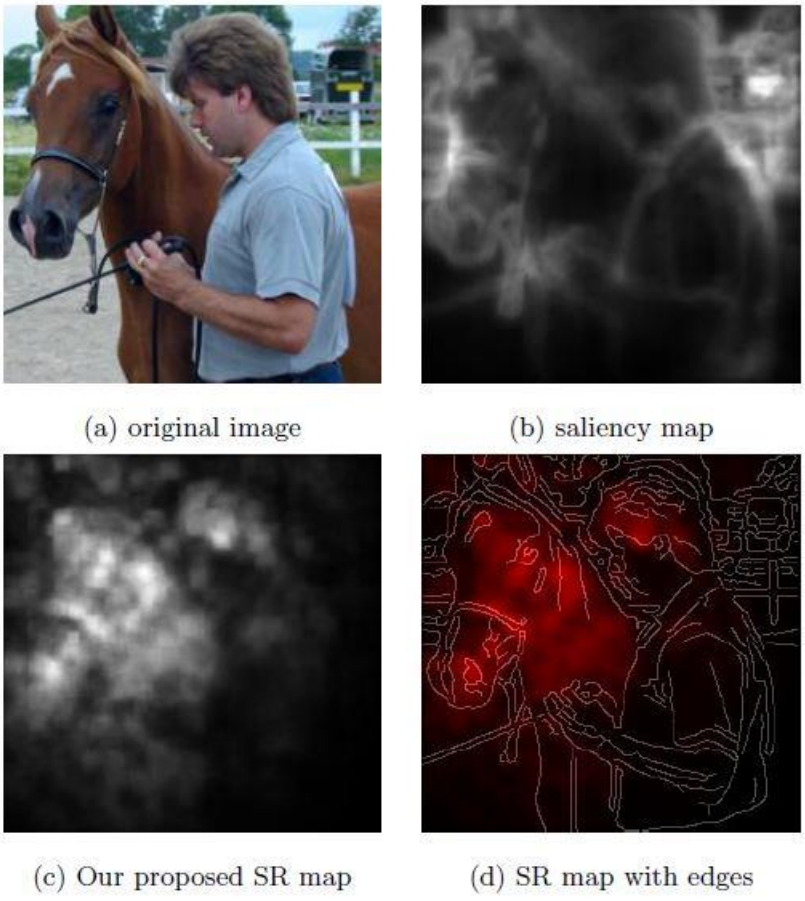

Figura 20 [26] – Esempio di visualizzazione prodotta dal metodo Salient Relevance Mapping. La figura (a) corrisponde all'immagine in input, (b) rappresenta la mappa generata dall'applicazione del modello biologico di context-aware saliency, (c) rappresenta invece il risultato di Salient Relevance Mapping, che viene evidenziato attraverso i colori e la rappresentazione dei bordi dei soggetti rappresentati nell'immagine (d). In particolare, confrontando (b) e (d) si notano chiaramente le differenze tra le aree di attenzione del cervello umano e quelle di una CNN come in [26] per un'immagine come quella in (a).

2.2.5 Feature imparate ad un certo layer

Le tecniche di visualizzazione delle feature imparate da una CNN si dividono principalmente in due categorie: quelle basate su **class activation maximization**, che sintetizzano un'immagine in modo che attivi maggiormente un certo nodo, al fine di rappresentare le feature imparate dalla rete rappresentative di una certa classe (non legate dunque ad un'immagine di input), e quelle cosiddette di **code inversion**, che hanno invece come obiettivo la generazione di un vettore di attivazioni quanto più vicino possibile a quello generato da una specifica immagine di input, senza però utilizzare le informazioni di quell'immagine ma solo le feature rilevate in essa, in modo da visualizzare come la CNN codifica l'input ad ogni layer.

Ad esempio, in [17] viene presentata una tecnica di class activation maximization per visualizzare le classi imparate dal modello (figura 21). Questa tecnica utilizza ottimizzazioni numeriche che, a partire da un'immagine nulla, sono in grado di portare alla sintetizzazione dell'immagine rappresentativa di una specifica classe.

UpConvNet [27] (figura 22), segue invece l'approccio di code inversion, permettendo di invertire una CNN con l'utilizzo di un'altra rete che, a differenza di DeconvNet [16], necessita di essere allenata in modo da comprendere l'importanza delle feature in input (ovvero quelle in output alla rete principale, che la rete inversa deve rappresentare). Con questa tecnica è possibile invertire le feature imparate a qualsiasi layer della CNN, in modo da ottenere un'immagine

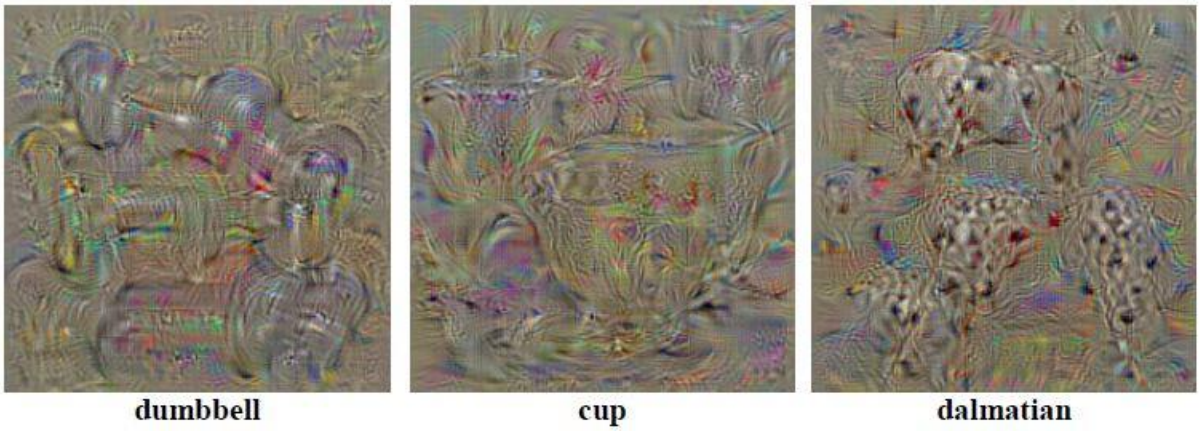

Figura 21 [17] – Esempio di visualizzazione generata con la tecnica di class activation maximization in [17]. Le immagini sono state prodotte attraverso ottimizzazioni numeriche, a partire da una specifica classe (indicata sotto a ciascuna di esse) e da una CNN allenata con il dataset ILSVRC-2013.

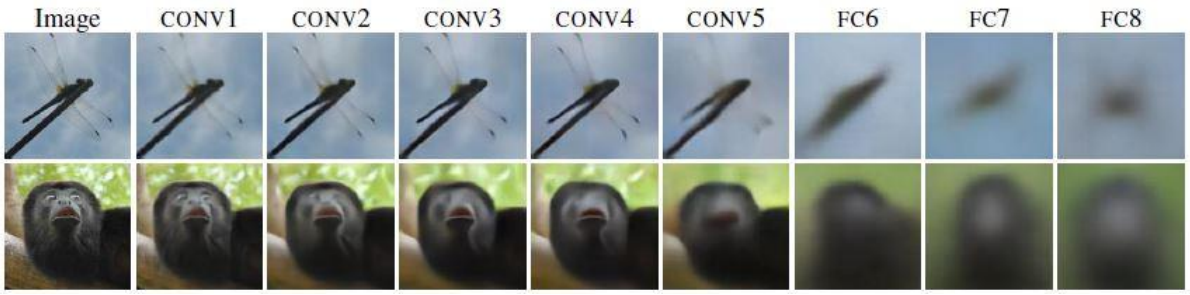

Figura 22 [27] – Esempio di visualizzazioni generate dal metodo UpConvNet. Vengono mostrate a sinistra due immagini di input, una per riga, e di seguito, andando verso destra, la ricostruzione delle due immagini prodotta da UpConvNet a partire dai layer indicati in alto della rete AlexNet.

sintetizzata che le rappresenta: la visualizzazione finale è dunque una media pesata di tutte le immagini naturali (utilizzate nella fase di training della rete inversa) che potrebbero aver prodotto le feature in esame. Il problema principale di questa tecnica consiste nella necessità di creare ed allenare una rete inversa ad hoc: questo risulta piuttosto oneroso, anche se una volta che la rete è pronta il processo che genera le visualizzazioni risulta piuttosto veloce.

Regularizer-based network inversion [28] (figura 23), anch'essa basata sul metodo di code inversion, rappresenta invece le feature imparate dalla CNN ricercando un'immagine che abbia una feature inversa il più vicino possibile alla feature in input, utilizzando le informazioni sui gradienti per ottimizzare un'immagine iniziale. Quest'ultima corrisponde a rumore casuale e dunque il calcolo dei gradienti rispetto a tale immagine risulta un'operazione onerosa: il processo per generare la visualizzazione infatti risulta più lento rispetto a quello di UpConvNet (escludendo la fase di creazione e training della rete inversa per quest'ultimo).

La differenza principale tra le tecniche di questa sezione e approcci come il metodo di deconvoluzione e gli altri descritti in 2.2.1, è che nel primo caso le informazioni utilizzate per la

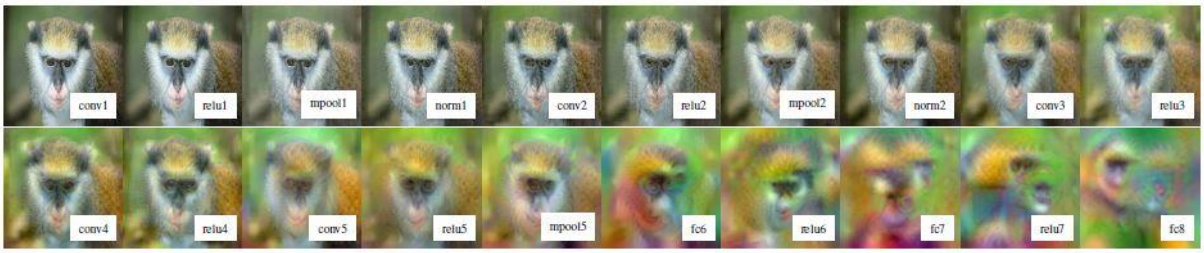

Figura 23 [28] – Esempio di visualizzazioni generate da Regularizer-based network inversion. In particolare, vengono mostrate le rappresentazioni prodotte da questa tecnica a partire da tutti i layer del modello Caffe-Alex definito in [28], per una data immagine di input (non mostrata qui ma del tutto analoga alla prima immagine in alto a sinistra).

rappresentazione delle feature dipendono esclusivamente dalla CNN, dunque dalle feature stesse: approcci di questo tipo vengono chiamati network-centric, in quanto necessitano solo di una rete allenata per funzionare. Nel secondo caso invece, sono necessarie informazioni extra riguardanti l'immagine in input (gli switch che memorizzano la posizione delle attivazioni ai Max Pooling layer per quanto riguarda DeconvNet, i gradienti delle attivazioni rispetto all'input per quanto riguarda i metodi gradient-based, entrambi in caso di Guided Backpropagation): metodi di questo tipo vengono detti dataset-centric, in quanto oltre alla rete allenata necessitano anche di informazioni riguardanti le immagini in input. In particolare, gli approcci descritti nella sezione 2.2.1 visualizzano come vengono ottenuti certi output della CNN, mentre con le tecniche descritte qui (class activation maximization e code inversion) vengono visualizzate le informazioni imparate dalla rete che sono preservate nell'output.

2.3 Software e librerie per la visualizzazione

In questo paragrafo si trattano le principali librerie e software (tabella 1) tra la moltitudine di progetti ricercati allo scopo di trovare un'implementazione standard e unificata per le svariate tecniche di visualizzazione trattate nei paragrafi 2.1 e 2.2. In particolare, sono inclusi i tool che risultano ad oggi tra i più utilizzati, come **TensorBoard** (2.3.1), **Embedding Projector** (2.3.6) e **ActiVis** (2.3.2, anche se solo internamente a Facebook), che spesso rappresentano lo standard de-facto per le rappresentazioni che rendono disponibili, oppure strumenti ritenuti interessanti sia per le soluzioni proposte sia per le caratteristiche del loro funzionamento, come il **Deep Visualization Toolbox** (2.3.3), **CNNVis** (2.3.4), **ReVACNN** (2.3.5), **DeepEyes** (2.3.7) ed **iNNvestigate** (2.3.8). Tra questi, alcuni risultano effettivamente disponibili per l'utilizzo, alcuni anche open-source, mentre altri esistono come pubblicazioni ma risultano non implementati o comunque non utilizzabili in quanto lo sviluppo non è terminato o è stato abbandonato. Infine, si giunge alla conclusione che iNNvestigate rappresenti la soluzione più completa ed interessante per quanto riguarda la visualizzazione delle attivazioni, e risulterà infatti alla base dello sviluppo di iNNvestigate-GUI, il nuovo tool presentato in questa tesi.

Tabella 1 - Riepilogo dei software trattati nel seguito e delle loro funzionalità

Visualizzazione	TensorBoard	CNNVis	Deep Visualization Toolbox	ActiVis	ReVACNN	DeepEyes	iNNvestigate
Grafo computazionale	✓			✓			
Pesi		✓			✓	✓	
Attivazioni, feature e gradienti		✓	✓	✓	✓	✓	✓
Embedding	✓			✓	✓		
Metriche	✓				✓	✓	
Framework supportati							
TensorFlow	✓						
Keras	✓				✓		✓
FBLearner				✓			
ConvNetJS					✓		
Caffe			✓			✓	
Environment							
GUI (web-app)	✓			✓	✓		
GUI			✓			✓	
Command line							✓
Disponibilità (open-source)							
Si	✓		✓		✓	✓	✓
No, proprietario				✓			
Non rilevato		✓					

2.3.1 TensorBoard

TensorBoard [29] è uno degli strumenti più completi per quanto riguarda la visualizzazione delle caratteristiche principali di una rete neurale. Questo tool fa parte di un progetto open-source portato avanti dagli sviluppatori di Google, completamente inglobato in **TensorFlow** e dunque vincolato ad esso. Nello specifico si tratta di un server locale molto leggero, in grado di riprodurre i grafici e le rappresentazioni relative alle visualizzazioni che rende disponibili, semplicemente basandosi su dati propriamente salvati attraverso l'aggiunta di poche linee di codice durante lo sviluppo del modello in TensorFlow.

Una delle visualizzazioni rese disponibili da TensorBoard è quella relativa ai grafi computazionali: questi possono essere molto complicati, e tipicamente lo sono, ed è dunque facile che abbiano molte regioni dove i link tra i vari nodi si sovrappongono in maniera confusa. Per risolvere tale problema, in TensorBoard i nodi contenenti strutture identiche vengono collassati

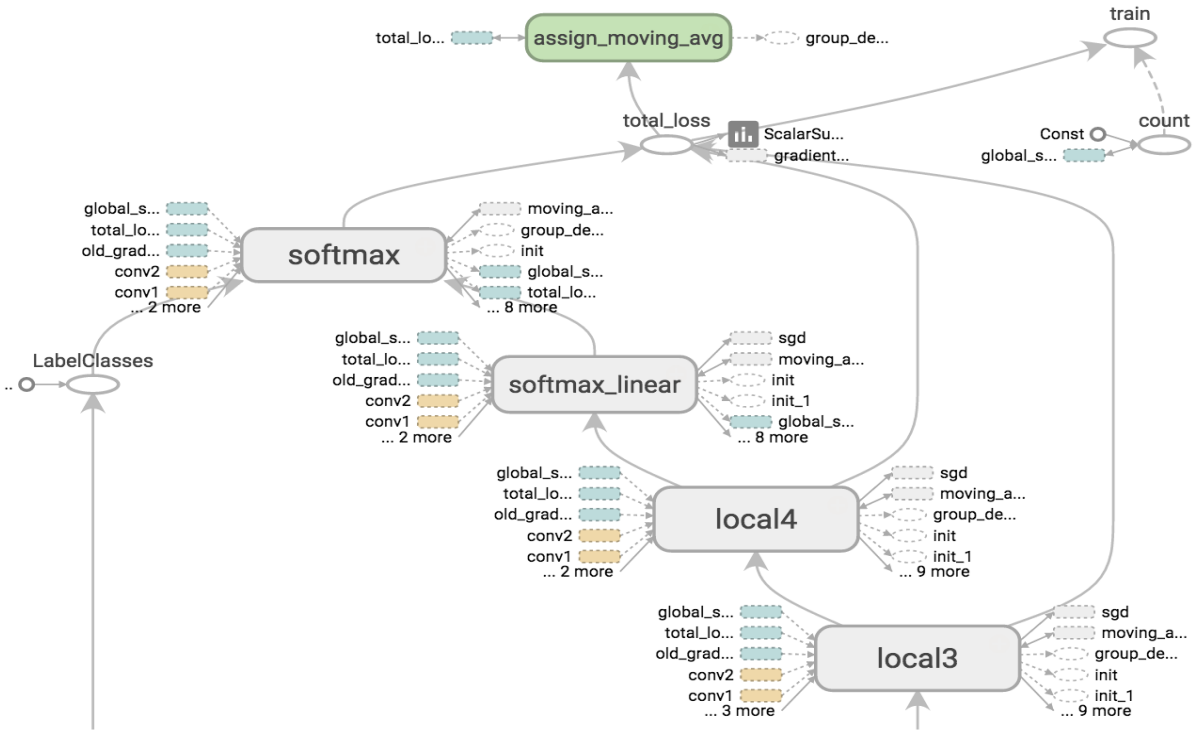

Figura 24 [29] – Esempio di visualizzazione di computational graph di una CNN in TensorBoard: i nodi principali sono visualizzati con tutti gli archi mentre i nodi con i contorni tratteggiati rappresentano dei raggruppamenti utili per semplificare la vista d'insieme, ma possono essere espansi qualora si vogliano analizzarne i dettagli.

in blocchi di alto livello (figura 24). Inoltre, i nodi con un gran numero di link uscenti o entranti vengono separati dal grafico generale per essere visualizzati in un'area separata dello schermo, facilitando la visualizzazione per potersi concentrare sulle sezioni principali del computational graph. Tali grafici sono interattivi, lasciando l'utente libero di muoversi all'interno degli stessi, ridimensionarli ed espanderne i vari raggruppamenti di nodi per visualizzare tutti i dettagli.

TensorBoard rende inoltre disponibile la visualizzazione di grafici corrispondenti a metriche interessanti del modello in esame, fornendo aggiornamenti in real-time durante l'apprendimento. Sono poi disponibili un certo numero di interazioni, che permettono ad esempio di visualizzare grafici di più modelli contemporaneamente per facilitarne il confronto, o raggruppare metriche, filtrare tra i diversi modelli, leggere il valore esatto delle metriche in determinati punti e ridimensionare i grafici per permettere un'analisi più approfondita.

2.3.2 ActiVis

Un altro strumento che permette di ottenere svariate visualizzazioni di una rete neurale è ActiVis [30], un tool web proprietario sviluppato da **Facebook** sulla piattaforma di machine learning di loro proprietà, **FBLearner**. Questo strumento si rivela a tutti gli effetti un'alternativa a TensorBoard, in quanto fornisce tutti i tipi di visualizzazione resi disponibili anche dal tool di Google. L'utilizzo di questo tool è immediato per quanto riguarda i modelli creati con FBLearner, mentre per quanto riguarda l'utilizzo con reti sviluppate con altre piattaforme, occorre aggiungere alcune linee di codice durante lo sviluppo del modello per fare in modo che durante l'addestramento della rete vengano prodotti i dati necessari alla visualizzazione.

Per quanto riguarda il computational graph, ActiVis richiede in ingresso una rete allenata ed utilizza un approccio differente rispetto a quello usato da TensorBoard, rappresentando sia le operazioni che le variabili come nodi (figura 25 A). Questo perché, oltre al grafo computazionale, cliccando su un nodo che identifica un layer, è possibile visualizzare tutte le attivazioni di quel layer ed effettuare ulteriori analisi su di esse (figura 25 B e C): dunque si è preferito

visualizzare il computationl graph in questo modo per renderlo più chiaro e ridurre il livello di complessità della vista d'insieme.

ActiVis infatti, fornisce la possibilità di effettuare instance- e subset-level observation, consentendo il confronto non solo tra le attivazioni di diverse istanze singole ma anche tra attivazioni di diverse istanze e gruppi di istanze. Questo è reso disponibile attraverso una visualizzazione (figura 25 B) che si attiva una volta selezionato il layer di interesse nel computational graph (figura 25 A), che mostra le attivazioni dei neuroni in quel layer suddivise per classe, oppure per istanza o ancora per gruppo di istanze, all'interno di una matrice di attivazioni dove le righe possono essere ordinate (ad esempio utilizzando il valore medio di attivazione per una determinata classe). Un'ulteriore vista (figura 25 C), permette di visualizzare nello specifico i dettagli

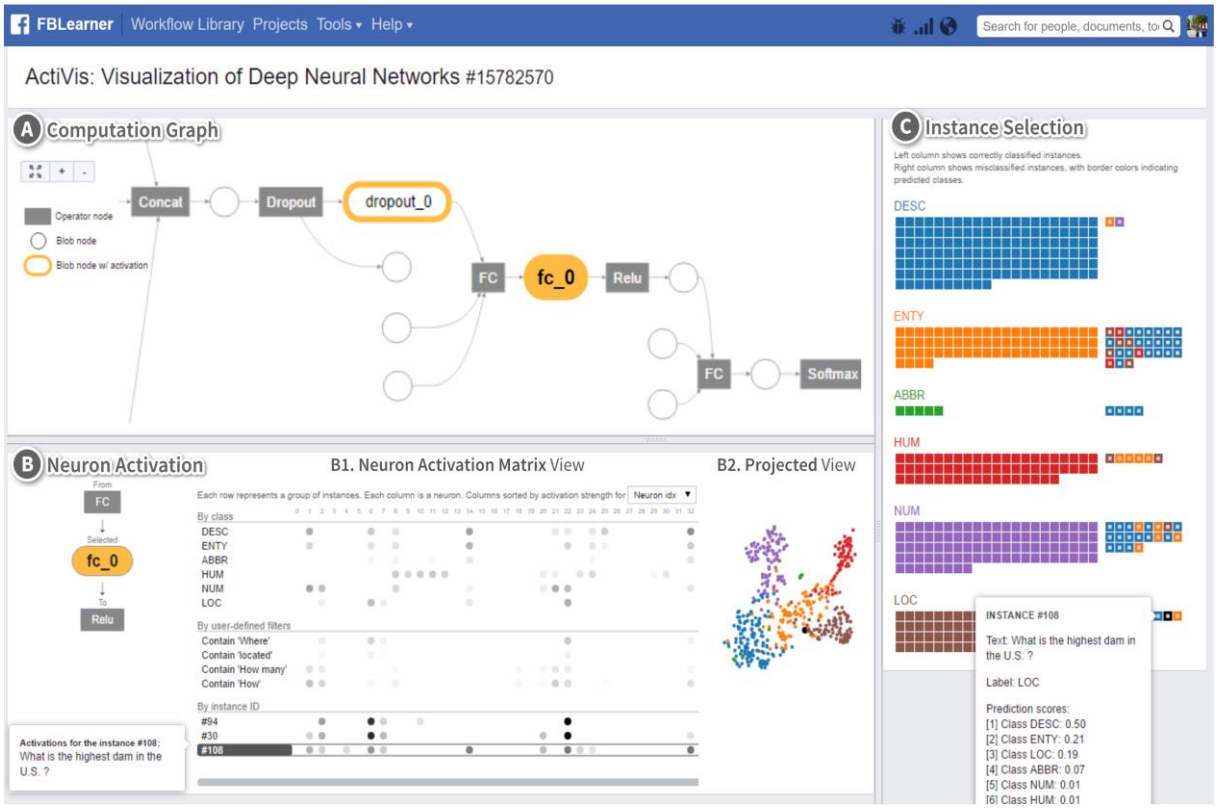

*Figura 25 [10] – **A**. Pannello di visualizzazione di computational graph di una CNN di esempio in ActiVis. **B**. Visualizzazione della matrice di attivazioni neurali per singole istanze, gruppi di istanze e classi (B1) e la corrispondente proiezione 2D delle attivazioni delle istanze utilizzando t-SNE (B2). **C**. Pannello di selezione istanze, dove vengono mostrati i risultati della classificazione: le istanze classificate correttamente sono mostrate a sinistra, mentre quelle sbagliate sono mostrate sulla destra. Cliccando su di un'istanza questa viene aggiunta alla matrice di visualizzazione delle attivazioni (B1) in modo da poter essere analizzata più approfonditamente.*

di ogni istanza, ed è da qui che tali istanze si possono ad esempio selezionare per i raggruppamenti utili ad effettuare confronti nella matrice di attivazioni.

Infine, ActiVis mette a disposizione una visualizzazione degli embedding. Infatti, una volta selezionato il layer di interesse nel computational graph, una delle viste disponibili riguarda la proiezione 2D dei relativi embedding, utilizzando la tenica t-SNE, dove le varie istanze sono colorate in base alla reale classificazione (da confrontare con la classificazione effettuata dalla CNN), ed è possibile analizzare il livello di apprendimento della rete a quel layer effettuando una analisi di vicinanza (figura 25 B).

2.3.3 Deep Visualization Toolbox

Uno dei tool fondamentali per quanto riguarda la visualizzazione di attivazioni e feature di una rete neurale, è il Deep Visualization Toolbox [31] (figura 26), open-source e che supporta nativamente Caffe (una tra le principali alternative a TensorFlow), ma può essere adattato per integrare qualsiasi altro framework per il Deep Learning. Questo strumento permette di inserire in input un video, tipicamente acquisito da una webcam, o un'immagine statica, e fornisce una rappresentazione interattiva delle attivazioni prodotte da ogni layer di una CNN allenata (inclusi i dense layer e quelli di pooling e normalizzazione).

Gran parte della finestra di visualizzazione è composta da un box dove è possibile selezionare un layer tra quelli che compongono la rete, e per tale layer vengono mostrati tutti i nodi con le rispettive attivazioni, dove le zone con le attivazioni più elevate sono evidenziate in bianco su sfondo nero. Oltre a questo, è presente uno spazio dove viene mostrato l'input (figura 26, in alto a sinistra) e, se viene selezionato un nodo tra quelli disponibili nel riquadro principale per il layer in esame, altri box completano la finestra di visualizzazione mostrando: una visione allargata della visualizzazione delle attivazioni del nodo selezionato, l'output del processo di deconvoluzione che parte dal layer in esame, ed infine 3 selezioni di 9 immagini ciascuna (figura 26, a destra). Le 3 selezioni riguardano rispettivamente: 9 immagini sintetiche rappresentative della classe relativa all'immagine in input, prodotte utilizzando la tecnica di class

activation maximization descritta in [17] con l'aggiunta di ulteriori ottimizzazioni, 9 immagini corrispondenti alle immagini di training che causano le attivazioni maggiori per il neurone selezionato ed infine 9 rappresentazioni di queste ultime 9 immagini di training, corrispondenti all'output del processo di deconvoluzione applicato su di esse a partire dal layer in esame.

Le immagini sintetiche generate dall'algoritmo di class activation maximization di cui sopra, identificano i maggiori stimoli per un particolare neurone, ma spesso corrispondono ad immagini che non rispecchiano la natura, che si riducono ad una composizione di "hack" generati ad-hoc dall'algoritmo per ottenere attivazioni elevate. Una combinazione di tecniche di regolarizzazione viene dunque utilizzata in [31] ed implementata nel Deep Visualization Toolbox al fine di migliorare il processo che genera l'immagine sintetica finale, in modo che tale output corrisponda il più possibile ad un'immagine realistica.

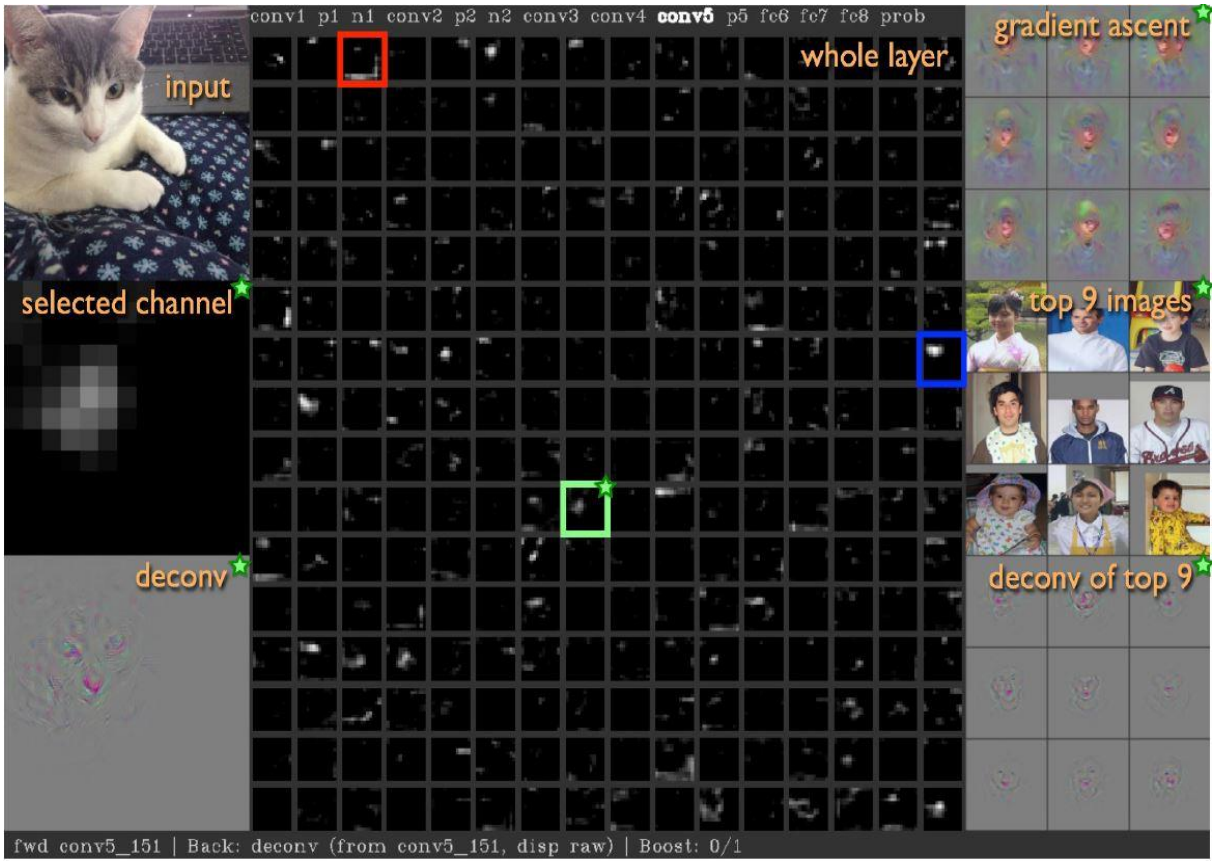

Figura 26 [31] – Deep Visualization Toolbox

2.3.4 CNNVis

Un altro progetto di tool per la visualizzazione delle attivazioni e delle feature imparate dalla rete è CNNVis [32] (figura 27), del quale però non è nota alcuna implementazione open-source utilizzabile, mentre è disponibile una demo[5].

Nella demo questo strumento utilizza come input una CNN allenata per risolvere un problema di classificazione e il suo training set, e permette di visualizzare anche modelli molto complessi. Infatti, grazie a tecniche di clusterizzazione, vengono rappresentati visivamente solo i layer più importanti e dunque i neuroni più importanti all'interno di ciascun layer. In particolare, per ciascun cluster di neuroni è possibile visualizzare le feature imparate, attraverso le immagini di input che causano le attivazioni maggiori per i neuroni all'interno di quel cluster. Tali immagini sono organizzate all'interno di un rettangolo dove la dimensione di ognuna dipende dalla sua importanza all'interno del cluster (attivazioni maggiori). È possibile inoltre visualizzare le

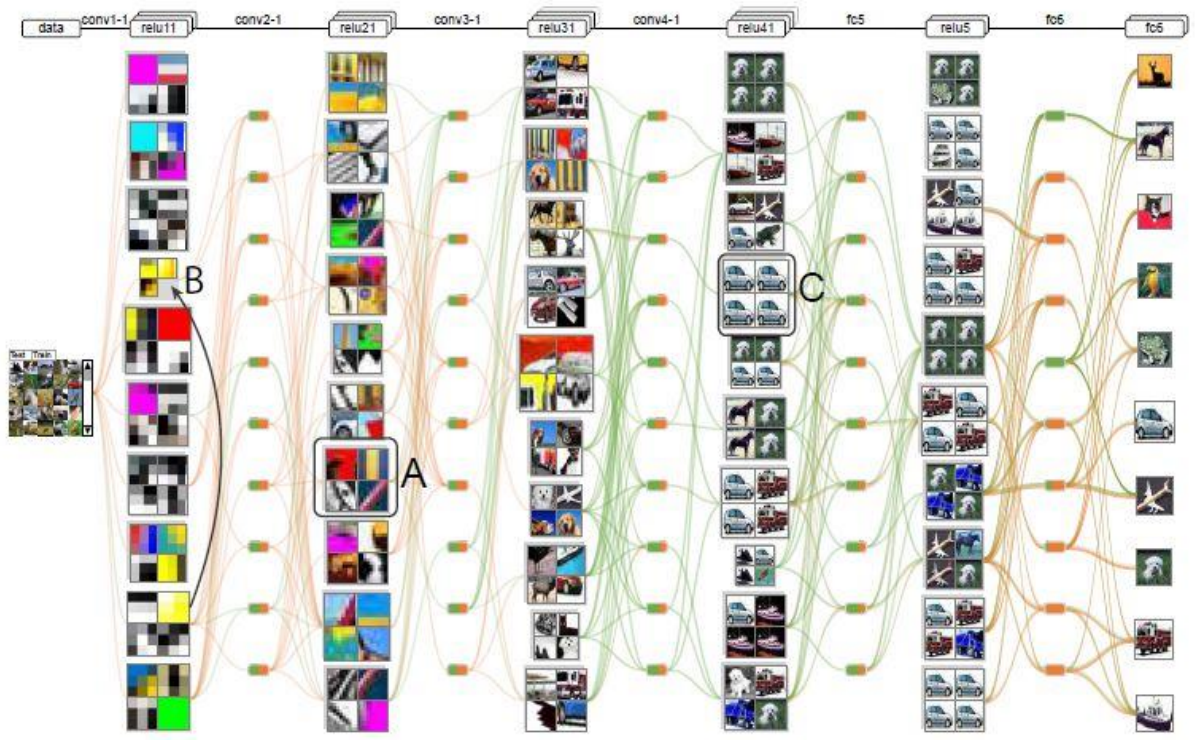

Figura 27 [32] – CNNVis

[5] http://shixialiu.com/publications/cnnvis/demo/

attivazioni di ogni neurone all'interno del cluster suddivise per classe, in una visualizzazione a matrice dove le righe, corrispondenti ai neuroni, sono ordinate in modo da facilitarne il confronto. Inoltre, per risolvere il problema della confusione prodotta dai molti link che collegano i vari neuroni, CNNVis utilizza una tecnica di bi-clusterizzazione raggruppando tali link e mostrando di default quelli con peso maggiore in valore assoluto, differenziando mediante colori diversi quelli positivi da quelli negativi. Ulteriori link possono essere visualizzati a scelta dall'utente, ma tipicamente quelli con peso maggiore sono i più interessanti da analizzare.

2.3.5 ReVACNN

ReVACNN [33] (figura 28), rientra tra i progetti di tool per la visualizzazione di attivazioni e feature, e in questo caso esiste un'implementazione open-source[6] sotto forma di applicazione web basata su ConvNetJS (framework JavaScript per il Deep Learning), ma risulta ad uno stadio piuttosto embrionale e ciò si traduce in uno strumento molto difficile, se non impossibile, da utilizzare.

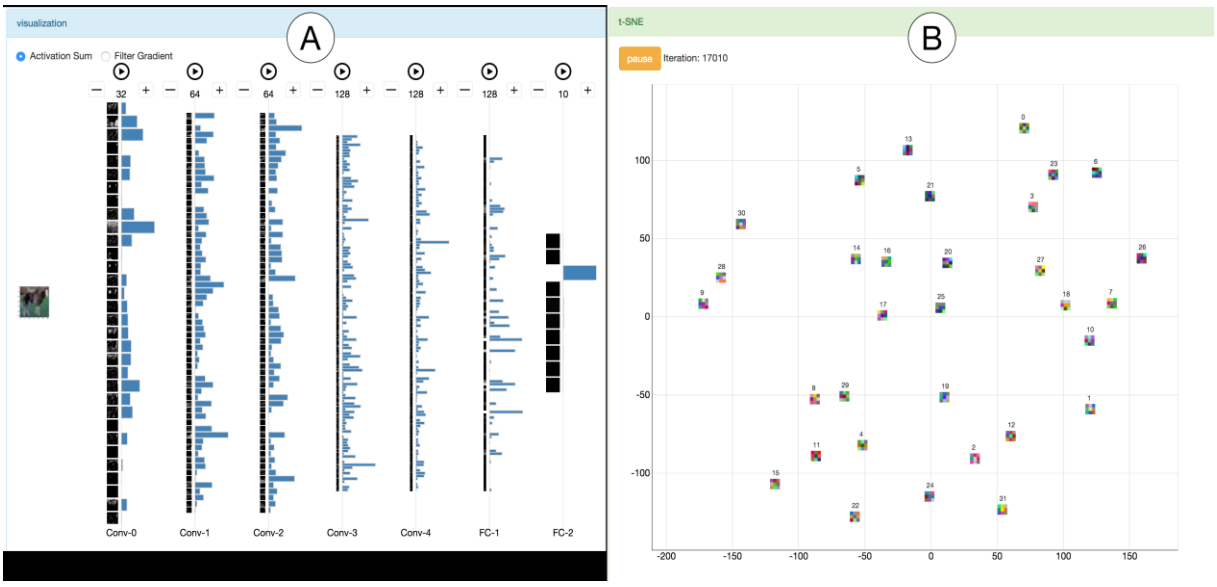

*Figura 28 [33] – ReVACNN. **A**. Network view. **B**. 2D Embedding View*

[6] http://github.com/sunghyo/revacnn

Nel progetto, il tool appare molto interessante in quanto permette di manipolare la rete durante l'allenamento e vedere in real-time l'effetto di eventuali modifiche. In particolare, una volta definita la rete e i dataset di input, ReVACNN è in grado di produrre due visualizzazioni principali, la cosiddetta *network view* e la *2D embedding view*. La prima (figura 28 A) fornisce una panoramica sulle attivazioni o i gradienti della rete, permettendo contemporaneamente di analizzare quali pattern vengono maggiormente riconosciuti dal modello, visualizzando le attivazioni (maggiori), e quali nodi risultano più stabili e convergenti in quello che imparano, visualizzando i gradienti (gradienti maggiori ad un filtro significa che quel filtro necessita di modifiche maggiori). Come detto, ReVACNN permette di visualizzare la rete durante l'addestramento e di modificarla per vedere gli effetti in real-time: nella *network view* è infatti possibile aggiungere o eliminare filtri negli hidden layer in maniera interattiva, ed anche congelare particolari nodi/layer che si ritengono inutili da allenare.

La *2D embedding view* (figura 28 B) invece, permette di esplorare i coefficienti dei filtri e le attivazioni usando una vista 2D degli embedding calcolata con l'algoritmo t-SNE, selezionando di volta in volta il layer di interesse. Questa modalità è utile per farsi un'idea di come i filtri siano collegati tra di loro, ed eventualmente riconoscere i filtri che non migliorano l'apprendimento della rete, imparando informazioni ridondanti, o quali informazioni non vengono invece imparate a causa della mancanza di filtri. Utilizzando questa conoscenza è possibile andare ad intervenire nella *network view*, ad esempio andando ad eliminare o ad aggiungere nodi o layer.

Infine, ReVACNN fornisce una visualizzazione che permette di monitorare l'andamento dell'addestramento della rete attraverso grafici a barre che si aggiornano nel tempo tenendo traccia ad esempio della perdita o dell'accuratezza della fase di training e della fase di validazione.

2.3.6 Embedding Projector

Uno dei tool più utilizzati per la visualizzazione di embedding è Embedding Projector [34] (figura 29), che utilizza le tecniche di riduzione di dimensionalità citate nella sezione 2.1.4,

PCA e t-SNE, ed inoltre fornisce una modalità personalizzata dove l'utente ha la possibilità di specificare come le varie proiezioni debbano essere effettuate. Quest'ultima tecnica è utilizzata soprattutto per permettere all'utilizzatore di trovare "direzioni" significative nello spazio degli embedding: ad esempio sottraendo l'embedding rappresentativo di "uomo" da quello riguardante la "donna", il vettore risultante rappresenta una direzione "femmina" nello spazio degli embedding [35].

Le visualizzazioni 2D o 3D prodotte da Embedding Projector, qualsiasi sia la tecnica utilizzata per generarle, sono interattive, in quanto l'utente è libero di spostarsi, zoomare, ruotarle, e di cliccare su punti in particolare per visualizzare informazioni riguardanti i punti più vicini, visualizzati anche graficamente, e le rispettive distanze. Inoltre, è anche possibile selezionare una serie di punti, come ad esempio tutti i vicini a partire da un punto particolare, e dunque isolarli per effettuare una visualizzazione più approfondita su di essi.

In generale Embedding Projector può essere utilizzato per qualsiasi tipo di visualizzazione e analisi di dati a grandi dimensioni, ma è ottimizzato per essere utile nei casi riguardanti il

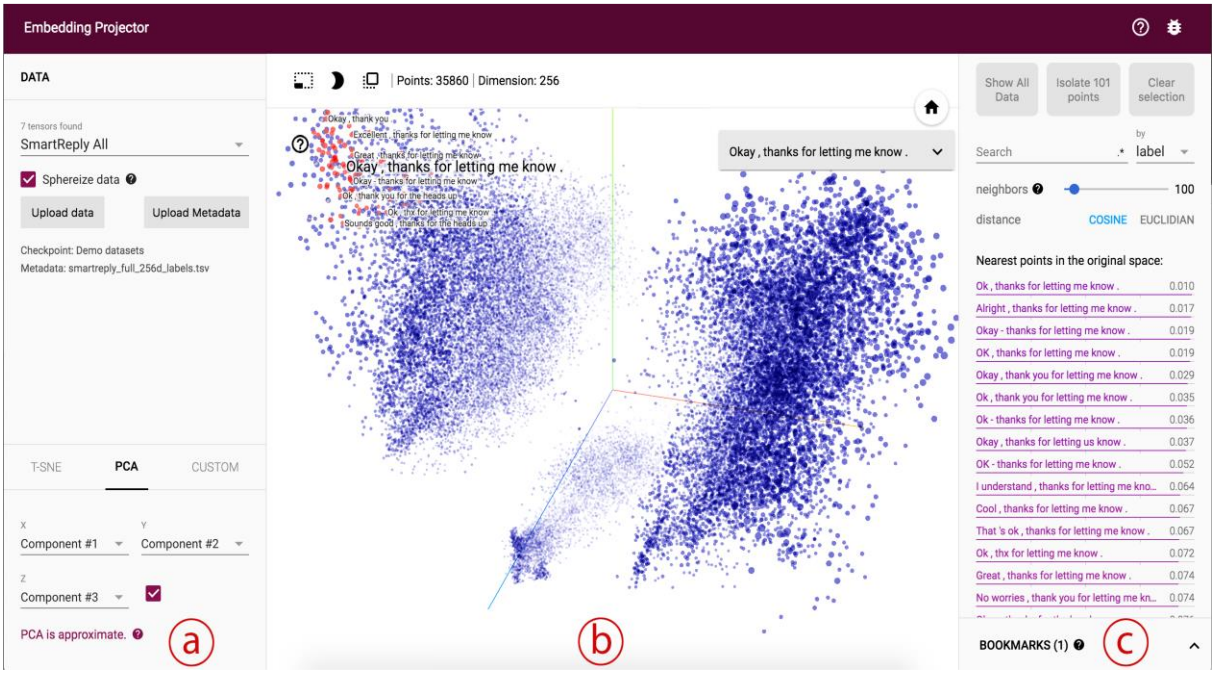

*Figura 29 [34] – Embedding Projector. **a**. Data panel, dove l'utente può scegliere quali dati mostrare ed etichettare i punti. **b**. Visualizzazione della proiezione. **c**. Inspector panel, dove l'utente può effettuare ricerche di punti in particolare e visualizzare una lista dei punti più vicini rispetto ad un certo punto.*

machine learning: tale tool infatti è disponibile sia come strumento a sé sia come facente parte del pacchetto TensorBoard in TensorFlow. Nel primo caso i dati vengono letti da un file di testo propriamente formattato, nel caso in cui si utilizzi in combinazione con TensorBoard invece, i dati vengono letti come spiegato nella sezione 2.3.1, dove viene trattato TensorBoard. In ogni caso è anche possibile esportare lo stato corrente della visualizzazione (ad esempio le coordinate calcolate per gli embedding in t-SNE) in modo da poter condividere tali viste.

2.3.7 DeepEyes

Un tool molto recente che fornisce diverse visualizzazioni interessanti è DeepEyes [36] (figura 30), un sistema cosiddetto di Progressive Visual Analitycs, poiché supporta il design di DNN direttamente durante la fase di training. Attraverso una serie di visualizzazioni di diverso tipo, questo strumento permette il soddisfacimento di una serie di obiettivi, alcuni dei quali mai

*Figura 30 [36] – Deep Eyes. **a**. Training Overview, mostra la perdita e l'accuratezza del modello in relazione al tempo. **b**. Perplexity Histograms, permettono di identificare i layer stabili. **c**. Activation Heatmap, permette di identificare i degenerated filters. **d**. Input Map, mostra le attivazioni dei filtri. **e**. Filter Map, mostra le relazioni tra i filtri del layer in esame. Input Map e Filter Map insieme, permettono di identificare layer sovradimensionati o inutili.*

affrontati da altri tool di visualizzazione. Nello specifico, DeepEyes permette l'identificazione di layer stabili dove vale maggiormente la pena concentrarsi per un'analisi più approfondita, così come l'identificazione di layer mancanti per risolvere il problema affrontato dalla rete in maniera più efficiente ed efficace. Permette inoltre di identificare i filtri che non contribuiscono alla soluzione (degenerated filters), e che possono dunque essere eliminati, così come i layer inutili. Ancora, vengono identificati i layer sovradimensionati, che contengono filtri inutilizzati che possono dunque essere eliminati, ed infine vengono segnalati i pattern non riconosciuti dalla rete, in modo da avvisare lo sviluppatore della DNN che potrebbero essere necessari ulteriori layer o filtri.

Per rendere tutto questo disponibile all'utente, vengono utilizzate diverse viste (figura 30): *Training Overview*, dove la perdita e l'accuratezza del modello vengono visualizzati in relazione al tempo; *Perplexity Histograms*, che permette di identificare i layer stabili (quelli che riconoscono un pattern in maniera stabile); *Activation Heatmap* per identificare i degenerated filters; infine *Input Map* e *Filter Map*, utilizzate rispettivamente per visualizzare le relazioni tra le attivazioni e i filtri di un certo layer e per mostrare come i filtri si attivano in maniera simile sull'input, e che insieme vengono utilizzate per l'identificazione dei layer sovradimensionati o inutili.

DeepEyes risulta dunque un tool molto completo e piuttosto unico nel suo genere per via della capacità di mostrare in maniera così approfondita tutto quello che potrebbe essere utile per migliorare il design di una DNN, e tutto questo in real-time. Supporta nativamente Caffe, ed è sviluppato in C++ per essere più efficiente possibile, utilizzando come design pattern il Model-View-Controller, in modo da poter essere facilmente esteso per supportare ulteriori librerie per il Deep Learning. È disponibile un'implementazione open-source[7] di DeepEyes, dove però il tool risulta ancora in fase di sviluppo.

[7] https://github.com/Nicola17/High-Dimensional-Inspector/tree/deepeyes

2.3.8 iNNvestigate

Come visto nel paragrafo 2.2, negli anni sono stati introdotti una moltitudine di metodi di visualizzazione delle attivazioni per cercare di comprendere il funzionamento delle reti neurali. Nella ricerca di strumenti che offrissero un'implementazione di riferimento per ciascuno di essi, per facilitarne l'integrazione nel processo di sviluppo di DNN, una delle soluzioni migliori trovate è rappresentata da **iNNvestigate**[8] [45], un tool Python con interfaccia a linea di comando basato su **Keras**[9] (uno dei più famosi framework open-source per il Deep Learning) con backend TensorFlow. In particolare, iNNvestigate rende disponibile un'interfaccia comune per gran parte dei metodi di visualizzazione visti nel paragrafo 2.2, di immediato utilizzo per la maggior parte delle DNN, e inoltre prevede una serie di API che lo rendono estensibile ad ulteriori metodi di visualizzazione ed integrabile in qualsiasi tipo di applicazione.

In realtà, iNNvestigate non è il solo tool del suo genere: i principali a porsi lo stesso obiettivo sono **keras-explain**[10] e **DeepExplain**[11], ed hanno caratteristiche simili. Nello specifico, supportano entrambi Keras con backend TensorFlow, come iNNvestigate, ma il primo prevede anche l'utilizzo di backend Theano (un altro framework per la definizione di reti neurali, ora meno utilizzato per via della grande diffusione di TensorFlow). Proprio keras-explain però, è compatibile solo con versioni di Keras comprese tra la 2.0 e la 2.2, a causa di un bug presente nelle versioni dalla 2.2 in poi, che ne impedirebbe il funzionamento. DeepExplain, dando più flessibilità all'utente in fase di input, risulta però più macchinoso degli altri nell'utilizzo, mentre keras-explain è piuttosto immediato, al pari di iNNvestigate. Rispetto agli altri due, iNNvestigate integra anche una serie di metodi di utility che semplificano ulteriormente il processo di analisi, come ad esempio un metodo per rimuovere la funzione di attivazione SoftMax nell'output layer di una rete, che spesso porta alla generazione di visualizzazioni falsate.

[8] https://github.com/albermax/innvestigate
[9] https://keras.io/
[10] https://github.com/primozgodec/keras-explain
[11] https://github.com/marcoancona/DeepExplain

Tutti e tre i tool implementano svariati metodi di visualizzazione, a volte coincidenti tra di loro, a volte sviluppati solo in uno dei tre. Ad esempio, la tecnica **LIME** (Local Interpretable Model-agnostic Explanations) [41], inclusa in keras-explain, produce visualizzazioni in grado di spiegare le previsioni del modello in input, approssimandolo attraverso un modello cosiddetto "interpretabile", ovvero in grado di fornire una rappresentazione immediatamente comprensibile per un essere umano (ad esempio un modello lineare o un albero di decisione).

DeepExplain include invece il metodo **Shapley Value Sampling** [42] che consiste in un'approssimazione, attraverso l'uso di campioni, del famoso algoritmo per la teoria dei giochi chiamato appunto Shapley Value [44], che risolve il problema della valutazione del contributo di ciascun giocatore partecipante ad un gioco cooperativo. In particolare, le feature di una rete neurale vengono fatte corrispondere ai giocatori, e utilizzando Shapley Value è possibile raggiungere una soluzione che rispetti tutti gli assiomi di un buon metodo di visualizzazione di tipo attribution [43], che però è impossibile da ottenere in pratica per una DNN a causa del costo temporale dell'algoritmo Shapley value che risulta NP-completo: [42] propone una tra le diverse tecniche di approssimazione sviluppate per risolvere questo problema. Questo metodo di visualizzazione rientra tra quelli perturbation-based, come il metodo di Grey-box occlusion [16], quest'ultimo implementato sia in keras-explain sia, tramite un'estensione, in DeepExplain.

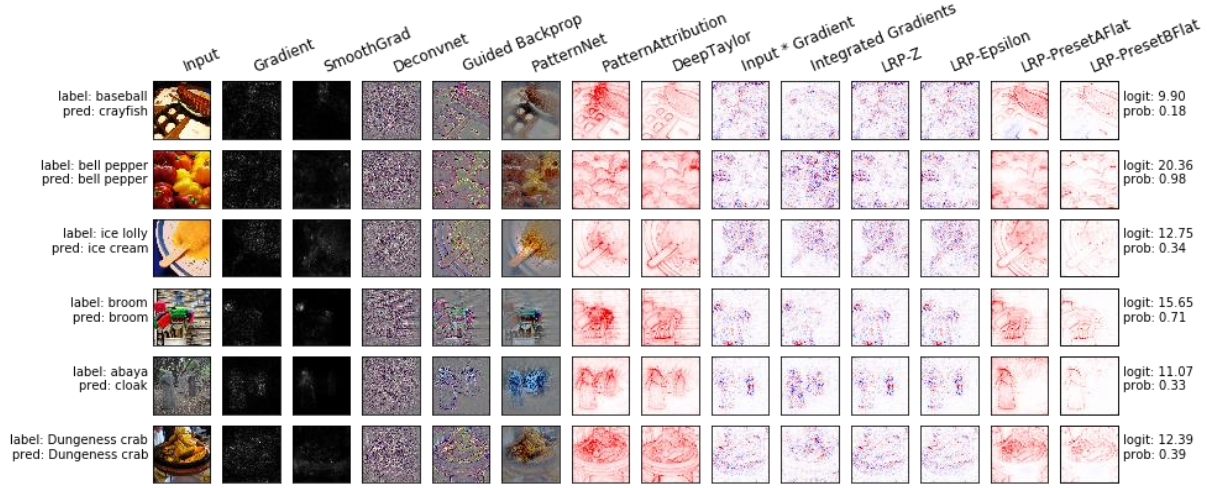

Figura 31 [46] – Vista d'insieme di alcuni dei metodi di visualizzazione implementati in iNNvestigate, applicati ad una serie casuale di immagini in input prese dal dataset di ImageNet, per la rete VGG16.

A parte i casi esposti sopra però, quasi tutti i metodi di visualizzazione delle attivazioni analizzati nel paragrafo 2.2 sono implementati esclusivamente in iNNvestigate, ad eccezione dei metodi PredictionDifference, GradCAM e Guided GradCAM che sono invece presenti solo in keras-explain. Inoltre, keras-explain e DeepExplain presentano un'implementazione di tipo LRP che però risulta in fase beta per il primo, e in una sola variante (LRP-ε) per il secondo: iNNvestigate invece contiene un gran numero di varianti LRP (che verranno trattate nel capitolo 3). La tabella 2 contiene un riepilogo di quanto detto finora riguardo ai metodi di visualizzazione implementati dai tre tool, mentre la figura 31 mostra alcuni dei metodi di visualizzazione implementati da iNNvestigate.

Fatte tutte queste considerazioni, iNNvestigate risulta il tool più completo, semplice da utilizzare e facilmente estensibile, per quanto riguarda la visualizzazione di attivazioni nelle varianti spiegate nel paragrafo 2.2, implementando molti dei metodi analizzati. Inoltre, il progetto risulta attivo, manutenuto dagli autori e completamente funzionante.

Tabella 2 – Tecniche di visualizzazione delle attivazioni implementate dai tre tool a confronto.

Tecnica di visualizzazione	KERAS-EXPLAIN	DEEPEXPLAIN	INNVESTIGATE
Gradient/Saliency maps	✓	✓	✓
SmoothGrad			✓
DeconvNet			✓
Guided Backpropagation	✓		✓
PatternNet			✓
GradCAM	✓		
Guided GradCAM	✓		
Input * Gradient		✓	✓
LRP	✓ (beta)	✓ (LRP-ε)	✓
Integrated Gradients	✓	✓	✓
DeepTaylor			✓
DeepLIFT		✓	✓
PatternAttribution			✓
Prediction Difference	✓		
Grey-box Occlusion	✓	✓ (un'estensione)	
LIME	✓		
Shapley Value Sampling		✓	

3 iNNvestigate-GUI

Nel capitolo 2 è stato trattato lo stato dell'arte per quanto riguarda la visualizzazione di reti neurali convoluzionali, che come detto sono al centro di questo lavoro per via della loro grande utilità e diffusione. Dunque, sono state approfondite le tecniche di visualizzazione delle attivazioni, argomento sul quale la ricerca si è focalizzata maggiormente negli ultimi anni per via della moltitudine di informazioni che possono venire rappresentate in questo modo, rendendo le CNN, e in generale le DNN, più comprensibili. Si è infine messo in evidenza come, sebbene esistano svariati tool, già utilizzabili o ancora in fase di sviluppo, che permettono di visualizzare le CNN utilizzando alcune delle tecniche disponibili, pochi risultino invece gli strumenti che offrono un'implementazione standard e unificata per molte delle suddette tecniche. Per quanto riguarda la visualizzazione delle attivazioni si è infine selezionato **iNNvestigate** [45] come miglior libreria di partenza in grado di fornire una soluzione al problema, costituendo un pacchetto pronto all'uso per la maggior parte delle CNN, che implementa le principali e più utilizzate tecniche.

Quello che si vuole fare con questo lavoro è presentare uno strumento che permetta di ottenere queste visualizzazioni **senza la necessità di scrivere codice ulteriore**, ponendosi nella maniera più trasparente possibile all'utente finale, prima di tutto attraverso un'**interfaccia grafica** di facile utilizzo. Inoltre, pur essendo molte le tecniche integrate in iNNvestigate, è stata individuata la mancanza di un metodo che si ritiene molto utile pensando soprattutto ad un'utenza non esperta, ovvero **GradCAM** (e **Guided GradCAM**). Infine, l'utilizzo di uno strumento come iNNvestigate presuppone di aver già individuato correttamente le immagini sulle quali approfondire l'analisi utilizzando le tecniche di visualizzazione a disposizione, ma è facile che questa non sia la situazione in cui potrebbero trovarsi utenti inesperti.

Viene quindi presentato iNNvestigate-GUI, il tool sviluppato per questa tesi che integra tutte le funzionalità di iNNvestigate in modo da rendere ancora più accessibile l'utilizzo delle tecniche di visualizzazione delle attivazioni disponibili, e ne estende ulteriormente le potenzialità, in un'interfaccia grafica web semplice ed intiuitiva. L'obiettivo primario è quello di fornire uno strumento che possa essere integrato completamente nel processo di sviluppo delle CNN, come ad esempio per la loro validazione, e che dunque fornisca un aiuto vero e proprio allo sviluppatore. In questo senso, iNNvestigate-GUI implementa una serie di **strumenti per l'analisi del dataset**, che permettono di ottenere suggerimenti sulle immagini da utilizzare come input per la generazione delle visualizzazioni, basandosi sugli altri input forniti, in modo da aiutare l'utente nella scelta dei casi più interessanti sui quali approfondire l'analisi.

Si può quindi considerare iNNvestigate-GUI come diviso in **due grandi blocchi**: il primo, che verrà analizzato nel seguito di questo capitolo, nel paragrafo 3.2, fornisce una GUI per iNNvestigate stesso, ampliando inoltre il numero di tecniche di visualizzazione disponibili e liberando l'utente dall'onere di svolgere determinate operazioni macchinose, necessarie in caso si utilizzi iNNvestigate in maniera isolata. Il secondo invece, riguarda le tecniche di analisi del dataset implementate, che verranno trattate nel capitolo 4.

3.1 Dettagli implementativi

3.1.1 Architettura web

iNNvestigate-GUI è stato sviluppato per essere semplice e più immediato possibile nell'utilizzo, portabile ed efficiente. Per fare ciò si è utilizzato il framework **Flask**[12], leggero ed estensibile, che permette di creare potenti applicazioni web con back-end Python. In particolare, in questo modo è possibile delegare la potenza di calcolo richiesta per le operazioni sulle reti

[12] http://flask.pocoo.org/

neurali e per la generazione delle visualizzazioni su macchine ad elevate prestazioni, mentre è sufficiente che il client supporti l'utilizzo di un browser web per accedere all'applicazione.

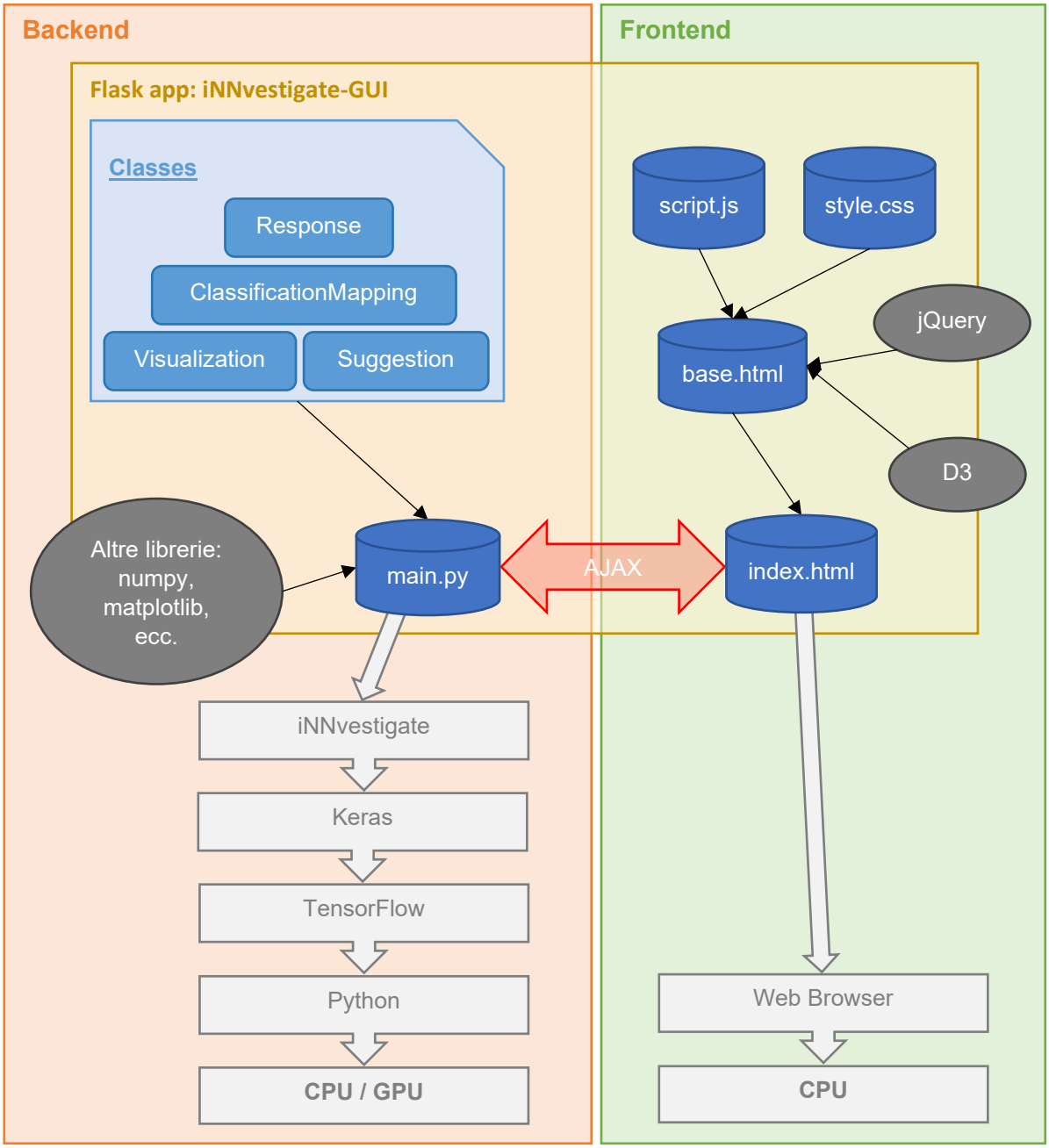

Figura 32 – Schema di funzionamento di iNNvestigate-GUI

Per il front-end sono stati utilizzati HTML, CSS, JavaScript, D3[13] e jQuery[14] per creare un'interfaccia grafica intuitiva e funzionale, mentre tutte le interazioni tra il client e il server avvengono tramite AJAX. L'implementazione in Python è basata su Keras con back-end TensorFlow, in modo da supportare tutte le funzionalità rese disponibili da iNNvestigate. Il software è composto da un unico Blueprint (main.py), ovvero un'astrazione del concetto di applicazione Flask che permette di avere maggiore flessibilità nello sviluppo, che in effetti costituisce il blueprint (planimetria) dell'applicazione stessa, definendo una serie di operazioni che vengono poi registrate dall'applicazione per il loro utilizzo. Questo contiene tutti i metodi per il funzionamento del programma e per la gestione delle richieste ricevute dal client, tramite la view associata al Blueprint, che in questo caso è costituita da un unico template html (index.html). Lo schema di funzionamento è visibile in figura 32. La cartella *`innvestigate-gui/static`* contiene tutte le risorse statiche del programma, come le librerie jQuery, D3 e le icone utilizzate nel tool, ed è inoltre presente la cartella *`output`* dove è possibile recuperare tutte le visualizzazioni prodotte da iNNvestigate-GUI.

3.1.2 Framework per il Deep Learning

Per il momento, l'unica versione supportata di **Keras** è la 2.2.4, richiesto come requisito per iNNvestigate, mentre a causa di alcuni problemi di incompatibilità con la funzione di logging utilizzata in iNNvestigate-GUI, è necessaria una versione di **TensorFlow** precedente alla 1.14. Le altre librerie richieste sono *numpy* e *scipy*[15] per l'utilizzo di funzioni scientifiche in Python, *opencv*[16] per la generazione di alcuni tipi di visualizzazione e *matplotlib*[17] per ottenere le immagini contenenti le visualizzazioni prodotte dal tool. I modelli di reti neurali supportati da iNNvestigate-GUI sono tutti quelli predefiniti in Keras, ed è possibile utilizzare qualsiasi

[13] https://d3js.org/
[14] https://jquery.com/
[15] https://www.scipy.org/
[16] https://opencv.org/
[17] https://matplotlib.org/

modello personalizzato, precedentemente esportato in formato *h5*. Le tecniche di visualizzazione implementate da iNNvestigate non sono tutte compatibili con qualsiasi tipo di layer, come ad esempio PatternNet e PatternAttribution, mentre SmoothGrad e IntegratedGradients attualmente non supportano i modelli contenenti Lambda Layer. L'unico altro vincolo riguarda ancora i modelli contenenti Lambda Layer che, per poter essere utilizzati, devono essere stati esportati con la stessa versione di Python in uso per l'esecuzione del tool stesso: questo perché la serializzazione/de-serializzazione risulta differente in versioni di Python diverse, e per i Lambda Layer, che contengono codice arbitrario, questo determina un problema in fase di export/import.

3.2 Visualizzazione delle attivazioni con iNNvestigate-GUI

All'apertura, iNNvestigate-GUI si presenta come mostrato in figura 33. La prima barra in alto è la barra dei **tab**, che è possibile utilizzare per muoversi all'interno degli output prodotti dal tool. Il primo è sempre il tab Home, che costituisce la schermata principale del tool, dove è possibile inserire gli input e scegliere che tipo di operazioni eseguire. Questo contiene, in alto, la zona dove vengono caricate le immagini, mentre in basso sono presenti i due grandi bottoni "***Suggest input images***" e "***Visualize***", relativi alle due macro-funzionalità del tool. Al centro

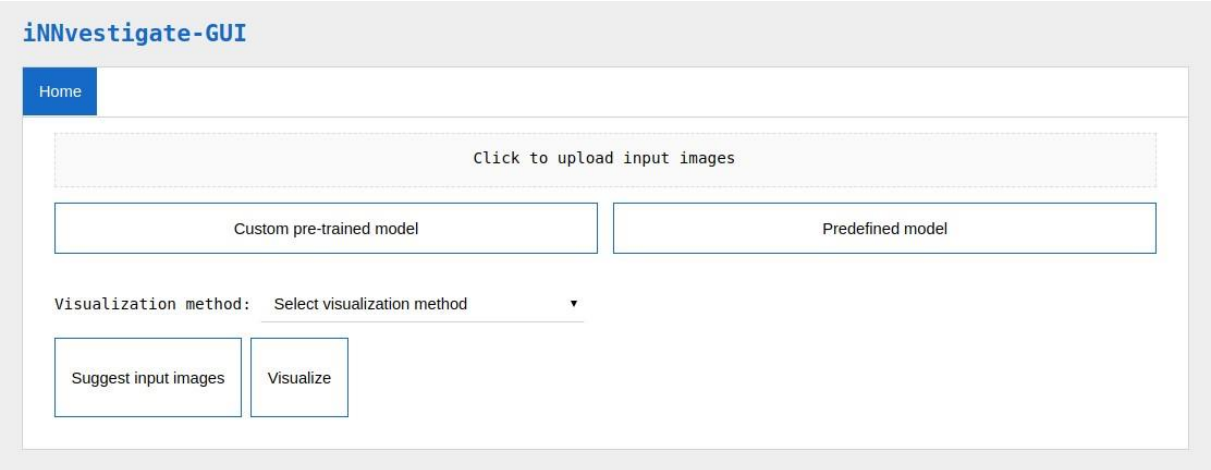

Figura 33 – Schermata iniziale di iNNvestigate-GUI

sono presenti altri due grandi bottoni, per selezionare l'utilizzo del tool con modelli custom oppure con modelli Keras predefiniti: l'utente può alternativamente passare da una all'altra modalità o di volta in volta cambiare il metodo di visualizzazione dal menu a tendina subito sotto, e gli output verranno generati in nuovi tab a mano a mano che "*Suggest input images*" o "*Visualize*" verranno cliccati. Come detto, questa parte si concentrerà sul funzionamento legato al bottone "*Visualize*".

3.2.1 Scelta dei modelli da utilizzare

Una volta cliccato su uno dei due bottoni per la scelta del tipo di modelli da utilizzare, si apriranno due schermate diverse a seconda dell'opzione: nel caso si selezioni "***Predefined model***" (figura 34), sarà possibile selezionare da una lista uno o più tra i modelli Keras predefiniti, e

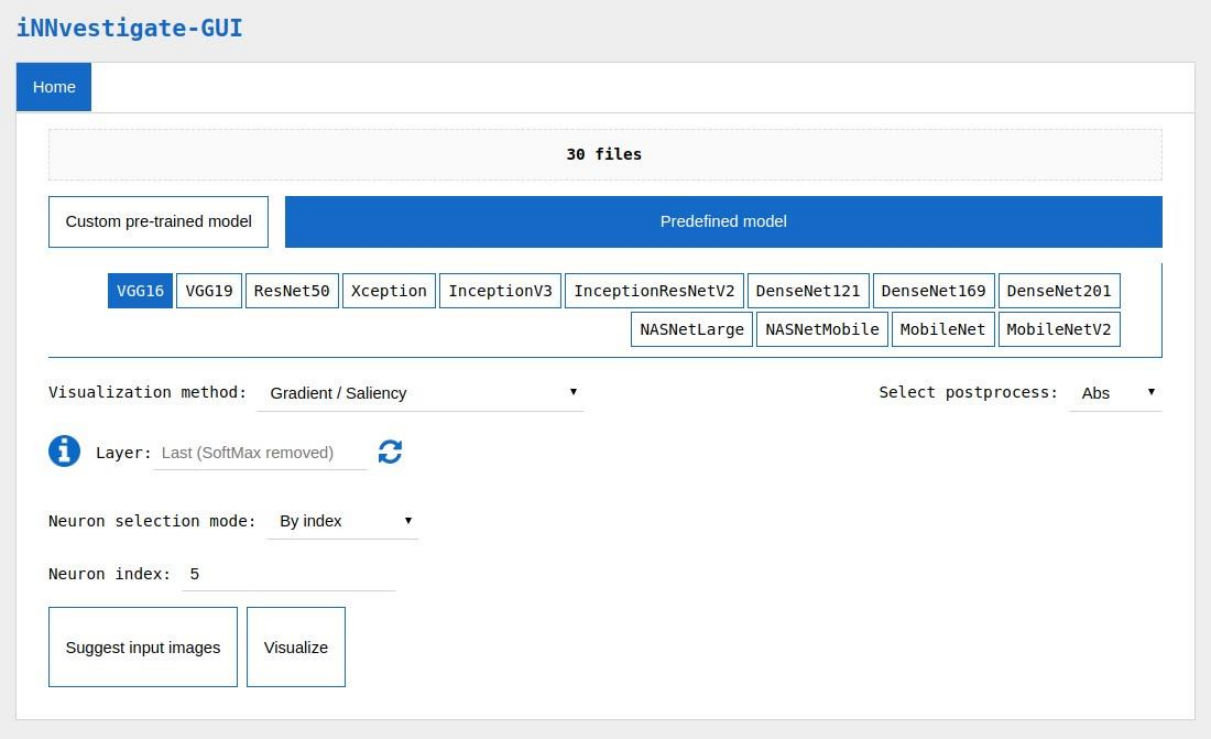

Figura 34 – Caricamento input per iNNvestigate-GUI in modalità "Predefined model". In particolare, in questo caso è stato scelto il modello VGG16 tra i predefiniti di Keras, e come metodo di visualizzazione "Gradient / Saliency". A destra sono inclusi i parametri aggiuntivi inerenti a tale metodo ("Select postprocess"), mentre a sinistra sono presenti i campi per la selezione del layer e del neurone da utilizzare per la visualizzazione.

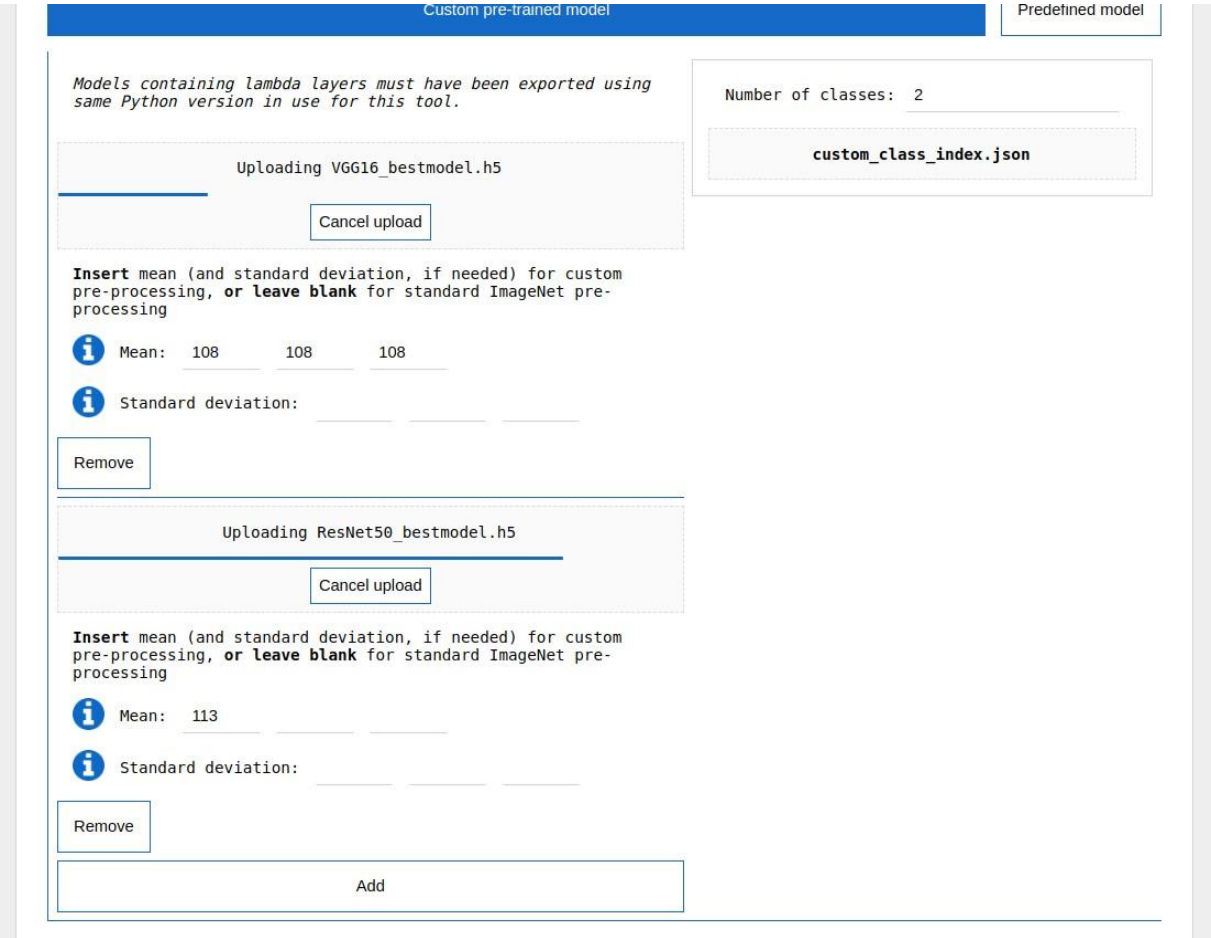

Figura 35 – Caricamento input per iNNvestigate-GUI in modalità "Custom model". In particolare, in questo caso due modelli sono in fase di caricamento: il primo chiamato "VGG16_bestmodel.h5", al quale sono stati associati i 3 valori RGB della media da utilizzare per il pre-processing delle immagini in input al tool. Per il secondo modello invece, si è scelto di utilizzare un valore d'insieme per la media da utilizzare per il pre-processing delle immagini. Infine, a destra sono presenti gli input per definire la classificazione effettuata dai modelli caricati: le classi in output saranno in questo caso 2, descritte dal CLASS_INDEX presente nel file "custom_class_index.json".

inserire anche gli input opzionali, se previsti dal modello. In questo caso, il pre-processing effettuato automaticamente sulle immagini in input, sarà quello predefinito in Keras per il modello selezionato, basato su **ImageNet**[18] (uno dei più famosi dataset per quanto riguarda le CNN), e la classificazione mostrata in output a fianco alle visualizzazioni sarà anch'essa basata sul dataset ImageNet. Utilizzando modelli predefiniti dunque, non resta altro che selezionare il

[18] http://www.image-net.org/

metodo di visualizzazione da utilizzare ed inserire gli eventuali altri input richiesti, che verranno trattati nelle sezioni 3.2.3, 3.2.4 e 3.2.5.

Nel caso in cui, invece, si scelga di utilizzare **modelli custom** (figura 35), ovvero modelli di reti neurali sviluppati personalmente in Keras con backend TensorFlow, sarà possibile caricare uno o più modelli di CNN, precedentemente esportati in formato *h5* tramite l'apposita funzione Keras[19] e, per ciascuno dei modelli caricati, sarà possibile inserire la media e la deviazione standard da utilizzare per il **pre-processing custom**. Utilizzando modelli personalizzati infatti, se l'utente non inserisce alcun valore per il pre-processing, le immagini in input verranno processate attraverso una funzione Keras predefinita basata sul dataset ImageNet, esattamente come avviene per i modelli predefiniti. Se le immagini del dataset da utilizzare non sono però dello stesso tipo di quelle incluse in ImageNet (ad esempio se ritraggono altri tipi di oggetti, o se ad esempio sono tutte in bianco e nero) questo potrebbe non rappresentare una buona soluzione, in quanto il pre-processing è un'operazione necessaria proprio per adattare i pixel dell'immagine rispetto alle caratteristiche del dataset, in modo da facilitare la rete nella classificazione. L'utente può quindi inserire i tre valori RGB della media e, se necessario, anche della deviazione standard, calcolate su un qualsiasi dataset personale che verrà dunque utilizzato nel tool, o ancora inserire un singolo valore di media d'insieme e analogamente per la deviazione standard: in ogni caso, in questo modo il pre-processing applicato sulle immagini di input sarà una sottrazione della media seguita dalla divisione per la deviazione standard (se inserita).

Oltre al caricamento dei modelli e dei valori per il pre-processing custom, sulla parte destra della schermata è presente un ulteriore box che permette di inserire **informazioni riguardanti la classificazione** che effettuano i modelli caricati. Questo box risulta unico anche se i modelli caricati sono più di uno, proprio perché utilizzare più modelli contemporaneamente è utile per effettuare confronti, e risultano ovviamente interessanti confronti tra modelli che risolvono lo stesso problema, ovvero, in questo caso, modelli con associate le stesse classi. Le informazioni sulle classi non sono obbligatorie nel caso in cui si vogliano utilizzare tecniche di visualizzazione che non dipendono da esse (come ad esempio Gradient), mentre per alcuni metodi sono

[19] https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

indispensabili (come ad esempio GradCAM), come si vedrà nella sezione 3.2.4. In particolare, se necessario, sarà obbligatorio inserire nel box sia il **numero delle classi**, sia il file JSON che rappresenta il **CLASS_INDEX** personalizzato, e le informazioni dovranno essere coerenti, ovvero il numero di classi indicato dovrà corrispondere al numero di classi effettivamente presenti nel file. Nello specifico, il CLASS_INDEX prevede una struttura simile a quello di ImageNet[20], e dovrà essere composto da tanti elementi quante sono le classi predicibili dal modello (o dai modelli), dove il nome di ogni elemento sarà l'indice, nel vettore di classificazione in output al modello, del neurone corrispondente alla classe in output, e il valore di ogni elemento sarà un array composto da un solo elemento, ovvero il nome della classe corrispondente. Un esempio di contenuto di un file CLASS_INDEX custom è il seguente:

```
{
    "0": ["rural"],
    "1": ["urban"]
}
```

dove vengono definite le classi di un modello binario che può classificare le immagini in input come appartenenti alla categoria "rurale" o "urbana".

3.2.2 Tecniche di visualizzazione supportate

Una volta scelta la modalità di utilizzo, se con modelli predefiniti o modelli custom, per ottenere le visualizzazioni occorre selezionare la tecnica da utilizzare. Qui entra in gioco iNNvestigate, che include l'implementazione di una serie di tecniche di visualizzazione delle attivazioni tra quelle trattate nel capitolo 2. In particolare, i metodi Gradient [17], SmoothGrad [18], DeconvNet [16], Guided Backpropagation [19], PatternNet [40], Input x Gradient [38], LRP [22], Integrated Gradients [39], DeepTaylor [37], e PatternAttribution [40] sono tutti disponibili in

[20] Il CLASS_INDEX non è altro che un file JSON che descrive l'associazione tra un numero e la classe corrispondente in un modello di classificazione. Un file CLASS_INDEX prevede un numero di elementi pari al numero delle classi del modello, indicizzati a partire da 0, ciascuno dei quali corrispondente ad un array contenente almeno il nome della classe corrispondente (più eventualmente altre informazioni relative alla classe).

iNNvestigate, e dunque utilizzabili in iNNvestigate-GUI. DeepLIFT [38] invece, è implementato in iNNvestigate come un wrapper al di sopra del codice originale, il che lo rende molto più lento e, per la nostra esperienza, utilizzabile solo con modelli molto piccoli e leggeri per via della grandissima capacità di risorse richieste: per questi motivi, considerando anche che le visualizzazioni prodotte sono molto simili a quelle prodotte dalle tecniche LRP, DeepLIFT non è incluso tra le tecniche utilizzabili in iNNvestigate-GUI.

Per quanto riguarda LRP invece, questo non corrisponde ad un solo algoritmo, bensì ad una famiglia di tecniche di visualizzazione caratterizzate dalla retropropagazione della rilevanza relativa ad una classificazione in output alla rete, come spiegato nella sezione 2.2.3. In particolare, le tecniche di visualizzazione delle attivazioni che rientrano nella famiglia LRP implementate in iNNvestigate, e dunque tutte disponibili in iNNvestigate-GUI sono: LRP-z, LRP-ε, LRP-αβ, già citate nel paragrafo 2.2.3, e per ciascuna di esse esiste anche la versione *IgnoreBias* che ignora il valore del bias nella formula di propagazione della rilevanza [22]; LRP-z^+ e LRP-z^+ Fast, due varianti LRP-z dove si considerano solo input maggiori o uguali a 0; LRP-z^B (LRP-Bounded), ancora una variante di LRP-z utilizzata nel caso in cui gli input siano racchiusi all'interno di un intervallo di valori definito [23]; LRP-w^2 (LRP-wSquare) che utilizza il valore dei pesi elevato al quadrato [37] nella formula iniziale della propagazione della rilevanza [22]; LRP-Flat che ignora il contributo del modello e dei dati in input per la propagazione della rilevanza [23]; e infine LRP Sequential Preset A, LRP Sequential Preset B, LRP Sequential Preset A Flat e LRP Sequential Preset B Flat, che rappresentano configurazioni speciali di LRP ad-hoc per l'utilizzo con le CNN, nella versione standard e nella versione Flat citata sopra[21]. Come per LRP, anche per DeepTaylor iNNvestigate implementa la variante da utilizzare nel caso in cui gli input siano compresi in un intervallo di valori definito, chiamata DeepTaylor-Bounded.

[21] https://innvestigate.readthedocs.io/en/latest/modules/analyzer.html#module-innvestigate.analyzer.relevance_based.relevance_analyzer

L'integrazione di GradCAM e Guided GradCAM

Oltre a tutte queste tecniche implementate da iNNvestigate, per ampliare le tipologie di visualizzazione disponibili, iNNvestigate-GUI include due ulteriori metodi di visualizzazione delle attivazioni tra quelli trattati nel capitolo 2, GradCAM e Guided GradCAM [21]. L'implementazione di GradCAM si basa su quella sviluppata in https://github.com/eclique/keras-gradcam, e corrisponde all'introduzione di una funzione aggiuntiva nel codice di iNNvestigate-GUI, i cui principali parametri in input sono il modello, l'id della classe da visualizzare (corrispondente ad uno degli id delle classi definite nel file CLASS_INDEX relativo al modello) e l'indice del layer convoluzionale all'interno del modello scelto da utilizzare per la visualizzazione. La selezione della classe verrà approfondita nella sezione 3.2.4, in quanto il funzionamento risulta lo stesso sia per quanto riguarda GradCAM e Guided GradCAM, sia per tutte le altre tecniche di visualizzazioni integrate direttamente in iNNvestigate. La selezione del layer è invece una funzionalità aggiuntiva rispetto ad iNNvestigate, che è stato necessario sviluppare per permettere l'introduzione di GradCAM e Guided GradCAM tra le tecniche disponibili in iNNvestigate-GUI, per le quali è necessario l'utilizzo di layer convoluzionali come spiegato nella sezione 2.2.2. Di questo si tratterà nella sezione 3.2.3.

Il funzionamento di GradCAM risulta dunque del tutto analogo a quello delle altre tecniche di visualizzazione offerte da iNNvestigate, richiamando l'apposita funzione invece di basarsi su quelle disponibili in iNNvestigate appunto. Per quanto riguarda Guided GradCAM invece, che corrisponde ad una moltiplicazione della mappa creata da GradCAM con la sensitivity map generata dal metodo Guided Backpropagation, come visto nella sezione 2.2.2, è stata utilizzata l'implementazione di Guided Backpropagation di iNNvestigate per ottenere la sensitivity map al layer selezionato, sempre restringendo la scelta ai soli layer convoluzionali. Parallelamente, è stata generata la mappa con il metodo GradCAM come descritto sopra ed infine le due mappe sono state moltiplicate insieme ottenendo l'output di Guided GradCAM.

Per verificare che i due metodi fossero stati integrati correttamente, si è utilizzato il tool online reso disponibile a http://gradcam.cloudcv.org/classification dagli autori del paper relativo a GradCAM [21], che permette di ottenere le rappresentazioni GradCAM di qualsiasi immagine

tramite la loro implementazione originale disponibile a https://github.com/ramprs/grad-cam. La scelta di utilizzare l'implementazione disponibile a https://github.com/eclique/keras-gradcam, come detto sopra, è stata dettata dal confronto, mediante i Jupyter Notebook Python[22], dei risultati di questa con altre implementazioni trovate online, che infine si è rivelata la più affidabile rispetto all'implementazione originale.

3.2.3 Selezione del layer

L'interfaccia a riga di comando di iNNvestigate non permette di scegliere il layer da utilizzare per la generazione della visualizzazione in output, ma considera sempre solo l'ultimo layer della rete per produrre il risultato finale. Questo può andare bene in svariati casi, ma volendo ad esempio integrare una tecnica di visualizzazione come GradCAM che, come visto nella sezione 2.2.2, risulta efficace solo su layer convoluzionali, il funzionamento imposto da iNNvestigate risulta inutilizzabile, a meno di scrivere del codice ad-hoc. Inoltre, in molti casi, è interessante poter consultare visualizzazioni generate anche con le tecniche già implementate in iNNvestigate a partire però da un layer arbitrario all'interno della rete, e non per forza l'ultimo. Ad esempio, i layer convoluzionali di una CNN sono in grado di riconoscere feature di diverso tipo: i primi layer si occupano di riconoscere caratteristiche più semplici delle immagini in input, focalizzandosi ad esempio su singole linee, angoli, o figure geometriche di base, mentre gli ultimi layer convoluzionali sono in grado di riconoscere figure più complesse, a partire da specifici pattern geometrici, fino ad esempio a riconoscere un volto. Ottenere le visualizzazioni ad un layer arbitrario all'interno di una CNN può quindi essere utile ad esempio per capire cosa viene riconosciuto maggiormente da quel specifico layer o viceversa se il layer non si focalizza su nessuna feature in particolare.

Per rendere anche questa operazione immediata, una volta selezionato il modello e la tecnica di visualizzazione da utilizzare, iNNvestigate-GUI permette la scelta del layer, come mostrato in

[22] https://jupyter.org/

figura 36. La scelta della tecnica di visualizzazione determina quali sono i layer del modello in uso disponibili per la visualizzazione stessa: ad esempio, per quanto riguarda GradCAM e Guided GradCAM, all'utente non vengono mostrati i layer della rete presenti dopo l'ultimo layer convoluzionale, in quanto non utilizzabili per questi metodi. La scelta del modello, invece, è necessaria per poter caricare correttamente i layer che lo compongono. La selezione del layer avviene per nome, ed è presente un pulsante per ricaricare i layer selezionabili qualora si cambi il modello in uso tra una visualizzazione e l'altra. Nello specifico, la selezione del layer viene implementata troncando il modello in uso, in modo che l'ultimo layer del nuovo modello corrisponda al layer selezionato: in questo modo è possibile utilizzare le funzionalità di iNNvestigate per produrre le visualizzazioni sul layer specificato, andando ad agire direttamente sul nuovo modello troncato.

Poiché la maggior parte dei metodi di visualizzazione non supporta l'utilizzo della funzione di attivazione SoftMax, solitamente utilizzata per generare l'output finale dell'ultimo layer delle

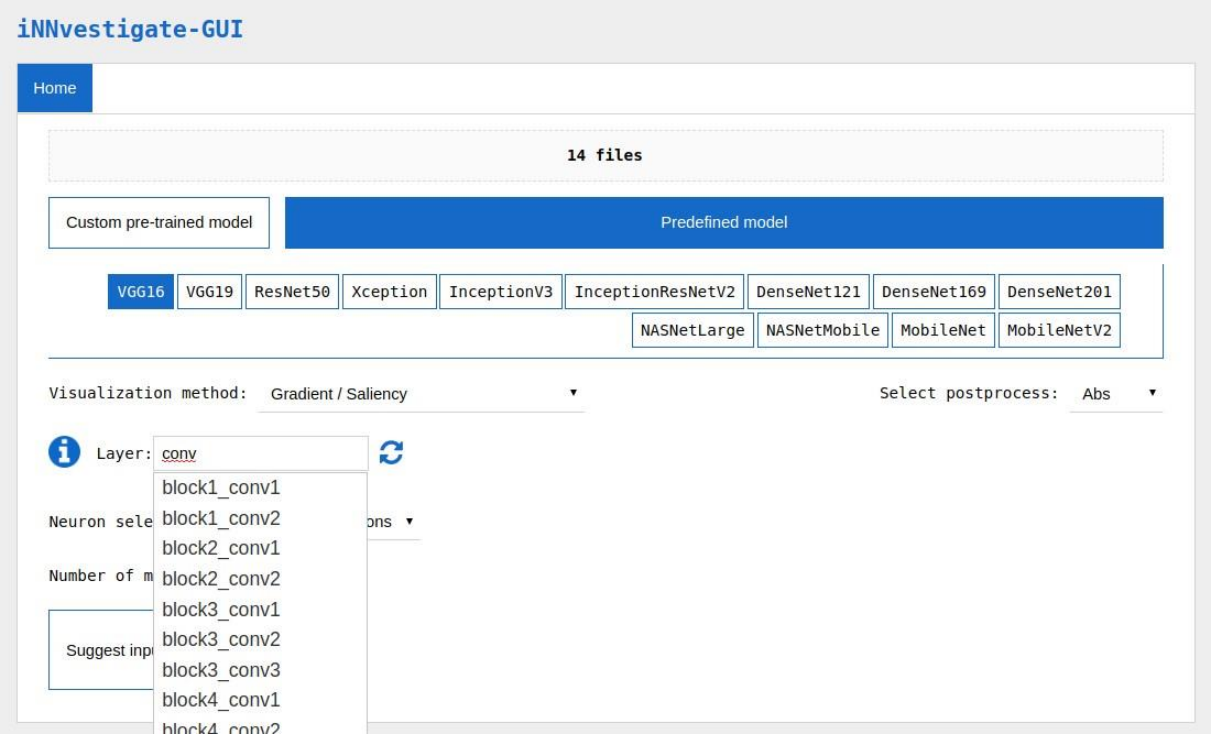

Figura 36 – Inserimento input per la tecnica di visualizzazione "Gradient / Saliency". In particolare, si è scelto di utilizzare 14 immagini in input e il modello predefinito VGG16. La scelta del layer da utilizzare per la visualizzazione avviene tramite l'inserimento del nome del layer in un box in autocomplete che suggerisce automaticamente i layer disponibili per la visualizzazione prescelta.

reti di classificazione, iNNvestigate mette a disposizione una funzionalità che trasforma l'ultimo layer originale in un layer identico, privato però della funzione di attivazione SoftMax finale. Se non si effettua alcuna scelta per la selezione del layer in iNNvestigate-GUI, per la visualizzazione verrà utilizzato l'ultimo layer (come default in iNNvestigate) dopo aver applicato il meccanismo di rimozione della funzione di SoftMax finale, a meno che il metodo di visualizzazione selezionato non sia GradCAM o Guided GradCAM: in questo caso, il comportamento default di iNNvestigate-GUI per generare la visualizzazione è l'utilizzo dell'ultimo layer convoluzionale presente nel modello in uso. Per le altre tecniche di visualizzazione invece, nel caso in cui si voglia utilizzare l'output dell'ultimo layer originale del modello, senza quindi sfruttare la funzionalità di rimozione della funzione di attivazione di SoftMax finale, basterà selezionare l'ultimo layer nella lista dei layer disponibili.

La selezione del layer è una funzionalità disponibile esclusivamente nel caso in cui si utilizzi il tool con un solo modello alla volta. Infatti, nel caso in cui si selezioni più di un modello predefinito, o si carichi più di un modello custom, la selezione del layer non è possibile, e viene utilizzato il layer di default come spiegato prima. Questo perché l'utilizzo di più modelli contemporaneamente è stato implementato appositamente per facilitare i confronti tra le visualizzazioni relative che si sceglie di generare, ma tali confronti hanno senso se riguardano il modello nella sua interezza: confrontare le visualizzazioni generate ad un layer arbitrario di un modello con quelle generate ad un layer arbitrario di un altro modello non ha senso.

3.2.4 Selezione del neurone o della classe

iNNvestigate permette la selezione del neurone da visualizzare. Il comportamento di default prevede la generazione della visualizzazione utilizzando il neurone corrispondente all'attivazione massima all'interno dell'ultimo layer della rete. Come detto, con iNNvestigate-GUI è possibile selezionare il layer da utilizzare, perciò il tool supporta sia la selezione del layer sia quella del neurone da visualizzare all'interno del layer prescelto.

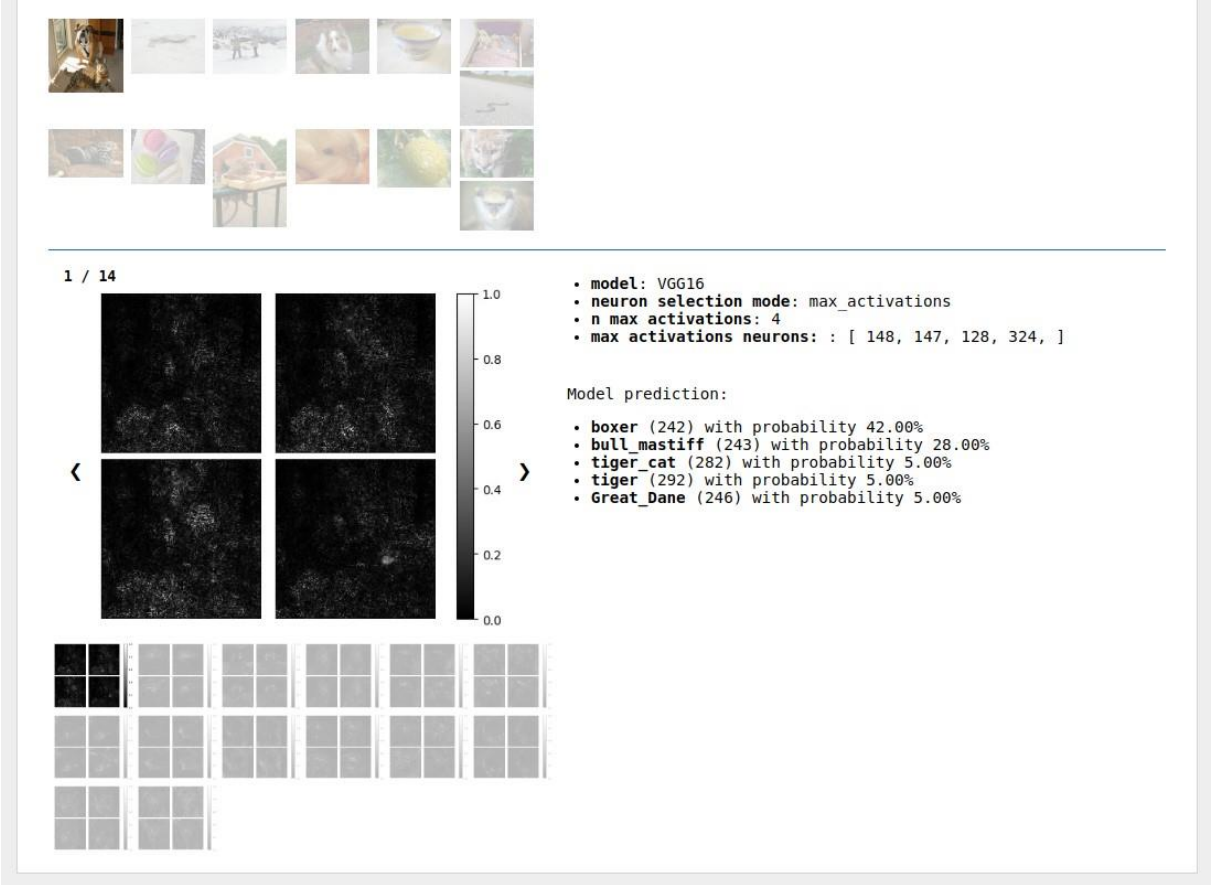

Figura 37 – Output generato dalla tecnica "Gradient / Saliency" utilizzando il modello predefinito VGG16 con 14 immagini in input e scegliendo la visualizzazione dei 4 neuroni con attivazione massima ("Max activations") nell'ultimo layer della rete (nessun layer è stato specificato in input). In particolare, l'immagine a sinistra corrisponde alle 4 rappresentazioni generate a partire dalla prima immagine in input (visibile solo parzialmente come miniatura, in alto) relative ai 4 neuroni con attivazione massima nell'ultimo layer del modello in input, identificati automaticamente dal tool come quelli con indice 148, 147, 128 e 324, come si vede nella parte destra dello screenshot, che contiene le informazioni sulla visualizzazione in corso.

La **scelta del neurone** può di per sé risultare interessante per alcune tecniche di visualizzazione, come Gradient, SmoothGrad, PatternNet, DeconvNet, Guided Backpropagation (anche se per questi ultimi due metodi la scelta del neurone da visualizzare produce risultati interessanti solo se effettuata su un layer convoluzionale), dove la visualizzazione generata rappresenta il pattern che causa attivazioni maggiori ad un determinato nodo, come spiegato nella sezione 2.2.1. Per questi metodi, iNNvestigate-GUI non solo integra la funzionalità base di scelta del neurone fornita da iNNvestigate, comunque potenziata in quanto effettuabile all'interno di un layer arbitrario, ma prevede anche una modalità di scelta automatica dei neuroni da visualizzare, sulla base dei rispettivi valori di attivazione. Le modalità di selezione del neurone, una volta

selezionato il layer da visualizzare, sono dunque due: la prima, "***By index***", fornita da iNNvestigate; la seconda, "***Max activations***", che permette di selezionare il numero, compreso tra 1, 4, 9 o 16, di neuroni da visualizzare corrispondenti alle attivazioni massime all'interno del layer selezionato. Un esempio di output generato utilizzando quest'ultima funzionalità è mostrato in figura 37. Per gli stessi motivi spiegati nella sezione 3.2.3, anche in questo caso la scelta del neurone per indice è vincolata all'utilizzo di un solo modello alla volta: nel caso in cui si lavori con più modelli contemporaneamente, sarà possibile scegliere solamente il numero di attivazioni massime da visualizzare.

Esclusi Gradient, SmoothGrad, PatternNet, DeconvNet e Guided Backpropagation, per gli altri metodi di visualizzazione inclusi in iNNvestigate-GUI però, la scelta del singolo neurone ha un riscontro utile solo per quanto riguarda l'ultimo layer della rete, dove ogni neurone corrisponde ad una delle possibili classi in output. Infatti, questi metodi mostrano le aree discriminanti in un'immagine di input (GradCAM e Guided GradCAM) e la rilevanza dei pixel delle immagini in input (tutti gli altri metodi, ovvero quelli di tipo attribution) per una certa classe: per queste tecniche di visualizzazione dunque, iNNvestigate-GUI rende disponibile una nuova funzionalità basata sul meccanismo di selezione dei neuroni fornito da iNNvestigate, ovvero la possibilità di **scegliere una classe** da utilizzare per la visualizzazione, tra quelle predicibili dal modello in uso. Questo vale anche nel caso in cui si inseriscano più modelli in input, e anzi può essere molto utile confrontare come una determinata classe venga rappresentata in un modello piuttosto che in un altro. Per utilizzare questa funzionalità con modelli custom, è richiesto l'inserimento dell'input relativo alle classi del modello (o dei modelli), come si può vedere nell'esempio di figura 35, ovvero il numero delle classi e il file CLASS_INDEX, come spiegato nella sezione 3.2.1. Se invece si utilizzano modelli predefiniti, le informazioni sulla classificazione sono quelle di default di Keras, dunque non occorrono ulteriori input: la scelta della classe da visualizzare potrà perciò in questo caso ricadere su una delle mille classi del dataset ImageNet.

Come detto, effettuare la scelta della classe da visualizzare sull'ultimo layer della rete, corrisponde alla selezione del neurone relativo a tale classe nell'ultimo layer della rete appunto. Quello che rende disponibile iNNvestigate-GUI è però l'utilizzo di tale funzionalità per

qualsiasi layer: questo avviene andando a selezionare, nel layer scelto, i neuroni che presentano gradienti con valore maggiore rispetto alla media del valore dei gradienti per tutti i neuroni di quel layer, relativamente alla classe selezionata. In questo modo vengono selezionati i neuroni rappresentativi, ad un determinato layer, della classe selezionata: per ciascuno di questi neuroni viene dunque generata la visualizzazione tramite la funzionalità base di selezione del neurone di iNNvestigate, e il risultato finale sarà la media di tali visualizzazioni. Nel caso in cui non venga specificata alcuna classe, verrà visualizzata come default la classe predetta in output dal modello. Come detto, la possibilità di scegliere la classe da visualizzare è disponibile anche nel caso in cui si utilizzino più modelli contemporaneamente: anche in questo caso però, non sarà possibile selezionare un layer specifico per ogni modello, ma si utilizzerà in maniera automatica l'ultimo layer della rete oppure, nel caso di visualizzazioni GradCAM o Guided GradCAM, l'ultimo layer convoluzionale della rete (per gli stessi motivi spiegati nella sezione 3.2.3).

3.2.5 Parametri aggiuntivi della visualizzazione

Una volta selezionata la tecnica di visualizzazione da utilizzare, oltre agli input riguardanti la selezione del layer e del neurone o della classe da utilizzare per la generazione dell'output, è possibile che vengano mostrati altri input riguardanti eventuali parametri aggiuntivi, obbligatori e non, a seconda del metodo di visualizzazione scelto. I parametri aggiuntivi possono essere: "*postprocess*", per scegliere come processare la heatmap risultante dai metodi Gradient, SmoothGrad e Integrated Gradients, se utilizzando il valore assoluto dell'output, il quadrato, o se non processare affatto i valori in uscita; per SmoothGrad è anche possibile specificare il parametro "*n*" che definisce il numero di perturbazioni da applicare sulle immagini in input, mentre scegliendo Integrated Gradients è possibile specificare anche il numero di passi di integrazione da effettuare. Per PatternNet e PatternAttribution occorre caricare le immagini necessarie per allenare lo stimatore del segnale della rete utilizzato in entrambi i metodi, mentre per DeepTaylor-Bounded è necessario specificare il limite inferiore e il limite superiore dell'intervallo dei valori in input. A seconda del particolare metodo LRP scelto invece, è possibile specificare i parametri

che ne definiscono il funzionamento: per LRP-ε, LRP Sequential Preset A e LRP Sequential Preset B è possibile specificare ε, per LRP-αβ è necessario inserire i valori di α e β, mentre per LRP-z^B (LRP-Bounded) è possibile specificare i limiti dell'intervallo dei valori in ingresso. Infine, quando presente, è possibile impostare il parametro "*Bias*" per i metodi LRP a *True* o a *False* a seconda che si voglia considerare o meno il bias nell'algoritmo, altrimenti "*Bias*" è automaticamente impostato a *False* o a *True* a seconda che il metodo includa o meno la dicitura *IgnoreBias*.

3.2.6 L'output generato

Dopo aver inserito tutti i parametri necessari, cliccando sul bottone "***Visualize***" il programma genererà le visualizzazioni richieste. In particolare, per ogni click su tale bottone, verrà creato

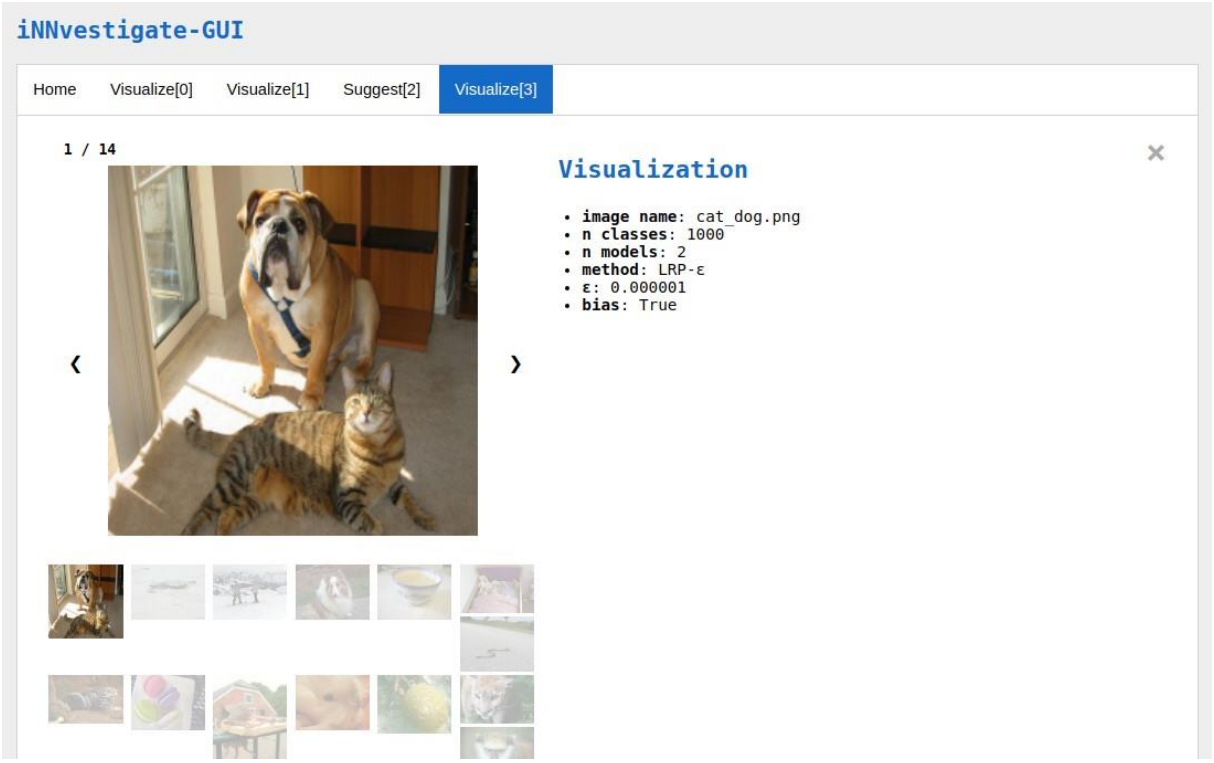

Figura 38 – Visualizzazione a tab di iNNvestigate-GUI. In particolare, nello screenshot è visibile la prima riga di un tab che conterrà le visualizzazioni generate dal metodo LRP-ε sulle 14 immagini in input mostrate a sinistra, mentre a destra sono presenti le informazioni sugli input che il tool ha utilizzato per la generazione delle visualizzazioni.

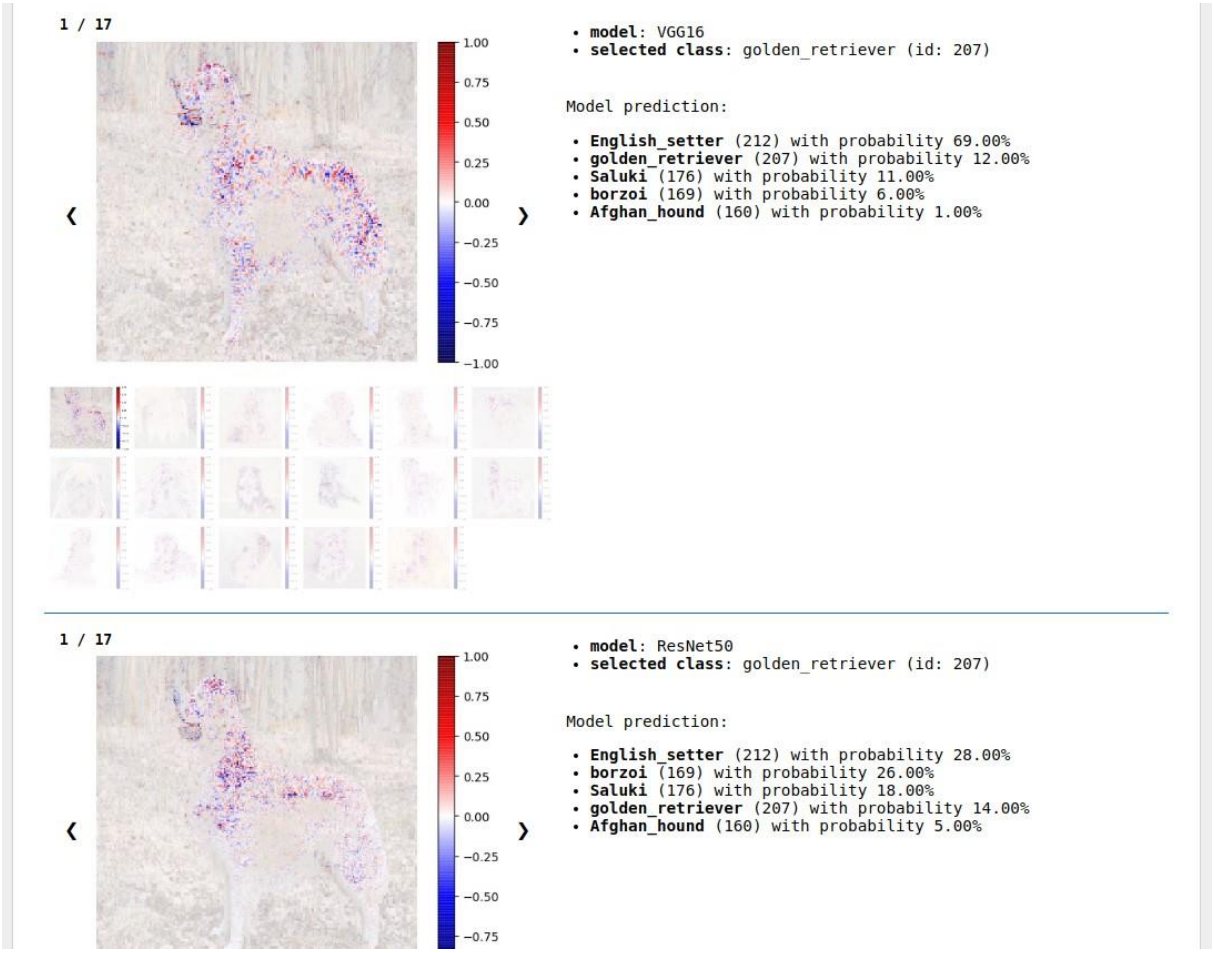

Figura 39 – Output generato dalla tecnica LRP-z utilizzando i modelli predefiniti VGG16 e ResNet50 su 17 immagini in input per la classe "golden_retriever". Lo screenshot rappresenta una parte del tab creato per questa visualizzazione, dove è possibile notare la separazione tra le due righe corrispondenti ai due modelli inseriti in input.

un nuovo tab, inserito in coda a destra dell'ultimo tab presente, che verrà subito reso attivo una volta che l'output è pronto. Il nuovo tab sarà composto da una prima riga (figura 38), sempre presente, che permette di visualizzare sulla sinistra le immagini inserite in input in uno slideshow, mentre a destra sono mostrati dati generici, quali informazioni sull'input attuale, il numero di modelli in uso, il numero di classi del/i modello/i, il metodo di visualizzazione scelto. A seconda del numero di modelli in uso, il tab è poi composto da altre righe, una sotto l'altra, ciascuna relativa alla visualizzazione generata da uno specifico modello (figura 39). Ognuna di queste righe presenta sulla sinistra lo slideshow con le visualizzazioni generate dal tool a partire dalle immagini in input utilizzando il modello specifico di tale riga, mentre sulla destra sono

presenti informazioni specifiche della visualizzazione associata a tale riga, ovvero in particolare informazioni riguardanti il modello corrispondente, quali: nome del modello, nome del layer (se selezionato), indice del neurone selezionato ("*By index*") o indici relativi ai neuroni corrispondenti alle attivazioni massime ("*Max activations*") o ancora nome della classe (per le tecniche di visualizzazione che lo permettono, se selezionata). Infine, sempre a destra, se si stanno utilizzando modelli predefiniti sarà sempre presente anche la previsione effettuata dal modello su ciascun input, in particolare mostrando sempre le 5 top-prediction. Nel caso in cui si lavori invece con modelli custom, la previsione viene mostrata solo nel caso in cui sia stato inserito anche l'input necessario alla traduzione della previsione numerica effettuata dal modello in classi (ovvero il numero delle classi e il CLASS_INDEX: figura 35, sezione 3.2.1).

Scorrendo lo slideshow di immagini in una delle righe facenti parte di un tab, anche gli altri slideshow del tab scorreranno di conseguenza, in modo da avere sempre visibili a colpo d'occhio le informazioni riguardanti lo stesso input. Per i metodi di visualizzazione che prevedono la scelta del neurone da visualizzare, come visto nella sezione 3.2.4, qualora si scelga di visualizzare i 1, 4, 9 o 16 neuroni corrispondenti alle attivazioni massime di un determinato layer, l'immagine in output corrisponderà ad un mosaico di 1, 4, 9 o 16 visualizzazioni generate a partire dalla stessa immagine di input, ciascuna delle quali corrispondente ad uno dei neuroni automaticamente identificati dal tool come aventi le attivazioni maggiori nel layer selezionato. Un esempio di questo tipo di output è visibile in figura 37. Per alcuni metodi di visualizzazione, lo slideshow contenente le visualizzazioni prodotte presenta anche una barra di legenda per permettere di interpretare meglio il risultato: ad esempio per i metodi LRP, le zone dell'immagine evidenziate con colori tendenti al blu sono quelle con rilevanza negativa per la classe per la quale si è generata la visualizzazione, mentre le aree rilevanti per tale classe sono quelle evidenziate con colori tendenti al rosso (figura 39).

É possibile mantenere aperti tutti i tab generati in modo da effettuare confronti tra le visualizzazioni prodotte con metodi di visualizzazione differenti, oppure tra quelle generate utilizzando la stessa tecnica ma su layer differenti dello stesso modello, oppure ancora per neuroni differenti di uno stesso layer o a partire da una classe differente, a seconda della tecnica di visualizzazione

scelta. Infine, ogni tab può essere eliminato utilizzando l’apposito bottone posto in alto a destra nel tab stesso.

4 Strumenti per l’analisi del dataset

Come anticipato nel capitolo 3, iNNvestigate-GUI rende disponibili due macro-funzionalità: la prima risulta un potenziamento di iNNvestigate, che lo arricchisce di un’interfaccia grafica intuitiva, ulteriori tecniche di visualizzazione delle attivazioni e altre funzioni come la selezione del layer, la selezione automatica dei neuroni corrispondenti alle attivazioni massime in un layer e la scelta della classe da visualizzare ad ogni layer. La seconda macro-funzionalità riguarda invece il **suggerimento delle immagini** che, all’interno di un dataset, rappresentano i casi più interessanti da utilizzare come input per effettuare ulteriori analisi con il tool stesso, della quale si parlerà in questo capitolo.

4.1 Motivazioni e funzionamento generale

iNNvestigate-GUI è stato sviluppato per l’utilizzo con CNN che risolvono problemi di classificazione. I dataset utilizzati in questo contesto sono costituiti da immagini spesso in numero molto elevato, sia per quanto riguarda il training e la validazione, per fare in modo che la rete raggiunga alti livelli di precisione, sia per quanto riguarda l’inferenza, per sfruttare appieno le potenzialità della rete laddove ad esempio un essere umano non potrebbe operare, proprio per via della grande mole di dati da esaminare. L’utilizzo di un tool come iNNvestigate-GUI per la validazione delle reti richiede in input le immagini che rappresentano i casi più interessanti da analizzare, ad esempio i casi in cui i modelli non effettuano correttamente le classificazioni, o le immagini sulle quali i modelli sembrano non riconoscere alcuna caratteristica saliente, o ancora i casi in cui i modelli risultano meno sicuri del risultato prodotto. Infatti, pensare di visualizzare le attivazioni di uno o più modelli per tutte le immagini all’interno di un dataset, viste

le dimensioni abituali degli stessi, risulta troppo lungo e difficilmente implementabile in un contesto di sviluppo.

D'altra parte, l'identificazione di casi interessanti come i suddetti, richiede per forza di cose un'analisi delle immagini in input, che può essere fatta manualmente, nei casi migliori e più evidenti anche ad occhio, quando il team di persone risulta sufficientemente esperto e le dimensioni del dataset lo consentono, oppure scrivendo del codice ad hoc che ricavi statistiche ed elabori informazioni d'insieme sulle immagini del dataset per raggiungere lo scopo. Comunque, l'operazione risulta soggetta ad errori umani, soprattutto nel primo caso, ed onerosa, andando ad aggiungersi alle motivazioni per cui la visualizzazione delle reti neurali viene spesso completamente tralasciata dagli sviluppatori.

Qui entra in gioco iNNvestigate-GUI, che permette l'integrazione di questa operazione con la possibilità di utilizzare le informazioni apprese da questo processo per effettuare direttamente analisi mirate, avvalendosi di numerose tecniche di visualizzazione immediatamente disponibili. Utilizzando la stessa interfaccia grafica, con le stesse modalità di input spiegate nel capitolo 3 legate alla visualizzazione vera e propria ed in particolare al funzionamento legato al bottone "*Visualize*", iNNvestigate-GUI include quattro diverse viste interattive sviluppate proprio per facilitare l'utente nell'individuazione dei casi interessanti per la visualizzazione, ognuna delle quali viene proposta, a seconda degli input forniti, alla pressione del bottone "***Suggest input images***".

Ciascuna vista può essere utilizzata per risolvere diversi problemi. Innanzitutto, per comprendere quali feature vengono riconosciute ad un determinato layer di un modello, occorre identificare le immagini che, all'interno del dataset, lo attivano maggiormente: per aiutare l'utente in questo problema viene proposta la vista contenente un istogramma che mostra la distribuzione dell'attivazione media provocata dalle immagini in input ad un determinato layer. Le immagini corrispondenti ad attivazioni medie maggiori per il layer in esame, saranno quelle maggiormente riconosciute dal layer, mentre sarà possibile identificare anche le immagini che invece sembrano non stimolare il layer, e capire il perché analizzandole con i metodi di visualizzazione. Questa vista può essere utile anche per estrarre informazioni sul comportamento del layer

selezionato con il dataset in uso, ad esempio identificando i layer che corrispondono ad un'attivazione media bassa, che quindi potrebbero non essere in grado di riconoscere nessuna caratterista saliente dell'immagine in input, o viceversa i layer corrispondenti ad un'attivazione media alta. Queste informazioni possono poi essere utilizzate per selezionare un layer specifico per la visualizzazione, per poter approfondire le analisi. Gli stessi discorsi valgono anche per un singolo neurone all'interno di un layer di un modello, dunque viene proposta una seconda vista analoga che contiene un istogramma che mostra la distribuzione dell'attivazione ad un determinato neurone per tutte le immagini in input.

Per valutare quali caratteristiche (ad esempio quali parti dell'oggetto o della scena) delle immagini in input vengono utilizzate per la classificazione da parte di reti differenti, iNNvestigate-GUI propone una terza vista, che contiene un grafico a dispersione che mostra la relazione che c'è tra confidenza e coerenza nelle classificazioni effettuate da queste reti per tutte le immagini del dataset, rappresentate come punti del grafico. Con **confidenza** si intende la sicurezza con il quale i modelli predicono una certa classe per un'immagine in input, ovvero il valore corrispondente alla probabilità assegnata da ciascuna rete alla sua top-1 prediction. La **coerenza** viene invece misurata contando il numero di classi differenti predette dalle diverse reti per una stessa immagine in input: se un'immagine viene assegnata ad una sola classe da tutte le reti significa che la coerenza tra le loro classificazioni è massima. Questo tipo di vista può essere utile anche per verificare l'andamento generale delle reti in esame sul dataset in input, analizzando le zone di maggior concentrazione nel grafico a dispersione.

L'ultima vista proposta da iNNvestigate-GUI viene mostrata, insieme a quella appena descritta, nel caso in cui il dataset in input contenga immagini appartenenti alla stessa classe. Questa contiene un istogramma che mostra la distribuzione della distanza dello score medio delle classificazioni effettuate delle reti in esame, rispetto alla classe di appartenenza del dataset, per tutte le immagini in input, e può essere utilizzata per verificare se reti neurali differenti si basano su caratteristiche dell'immagine diverse per riconoscere scene o oggetti di una determinata classe. Questa vista permette inoltre di individuare le reti che hanno predetto correttamente le immagini, sulla base della reale classe di appartenenza, ed eventualmente individuare le immagini

per le quali le reti non hanno prodotto il risultato corretto, per utilizzarle per approfondire l'analisi per apportare modifiche mirate ai modelli.

Il funzionamento specifico di ciascuna delle viste proposte e qui brevemente analizzate relativamente ai problemi per i quali possono essere utilizzate, verrà analizzato nel seguito di questo capitolo, nel paragrafo 4.2.

4.2 Soluzioni proposte

I suggerimenti per le immagini da utilizzare in input per le analisi successive vengono offerti da iNNvestigate-GUI attraverso diverse viste. Ognuna di queste viste viene mostrata in un nuovo tab che si accoda all'ultimo presente nella barra dei tab del tool, similmente a quanto avviene per le visualizzazioni, come descritto nel capitolo 3 (vedi ad esempio la figura 38). Come detto, le viste possono essere di quattro tipi differenti, a seconda dei dati forniti in input oltre al dataset di immagini, che possono essere suddivise in **due categorie**: quelle riguardanti i suggerimenti forniti sulla base di un solo modello in input (4.2.1), e quelle invece che utilizzano più di un modello (4.2.2), che siano essi i modelli predefiniti offerti dal tool o modelli custom. Si cerca così di proporre un set di metriche con i relativi grafici, che rappresentino le informazioni in una forma che permetta all'utente di confrontarsi con i risultati ottenuti sia nel caso in cui i dati a disposizione riguardino un solo modello, riducendo i suggerimenti a formule più semplici, sia nel caso più complesso che implica più modelli in input, ma che risulta essere quello più interessante.

4.2.1 Suggerimenti basati su un solo modello in input

Oltre al dataset, è sufficiente indicare un solo modello tra quelli predefiniti disponibili, oppure caricare un proprio modello custom, per poter ottenere due tipi di suggerimenti per le immagini in input, corrispondenti a due diverse viste, che rappresentano i casi più semplici. Questi si

basano infatti sul valore delle attivazioni generate dalle immagini in input ad un determinato layer del modello, oppure ad uno specifico neurone di un layer.

Come spiegato nella sezione 3.2.3, i layer selezionabili per la visualizzazione all'interno di un modello, dipendono ovviamente dall'architettura del modello stesso, ma anche dal metodo di visualizzazione scelto: a seconda dell'analisi da effettuare, sarà interessante utilizzare l'uno o l'altro tra i tanti disponibili. Per ottenere le viste trattate in questa sezione, che come detto si basano sulle attivazioni, è necessario effettuare le scelte opportune del layer o dello specifico neurone di un layer che si vuole utilizzare per la generazione dei suggerimenti. La scelta del metodo di visualizzazione è dunque necessaria per poter effettuare tale selezione, ma non è vincolante se non per la selezione del layer stesso (ad esempio scegliendo GradCAM non si potranno selezionare layer successivi all'ultimo layer convoluzionale).

Una volta scelto il metodo di visualizzazione è possibile dunque selezionare il layer, oppure semplicemente lasciare la scelta di default che, come spiegato nella sezione 3.2.3, ricade

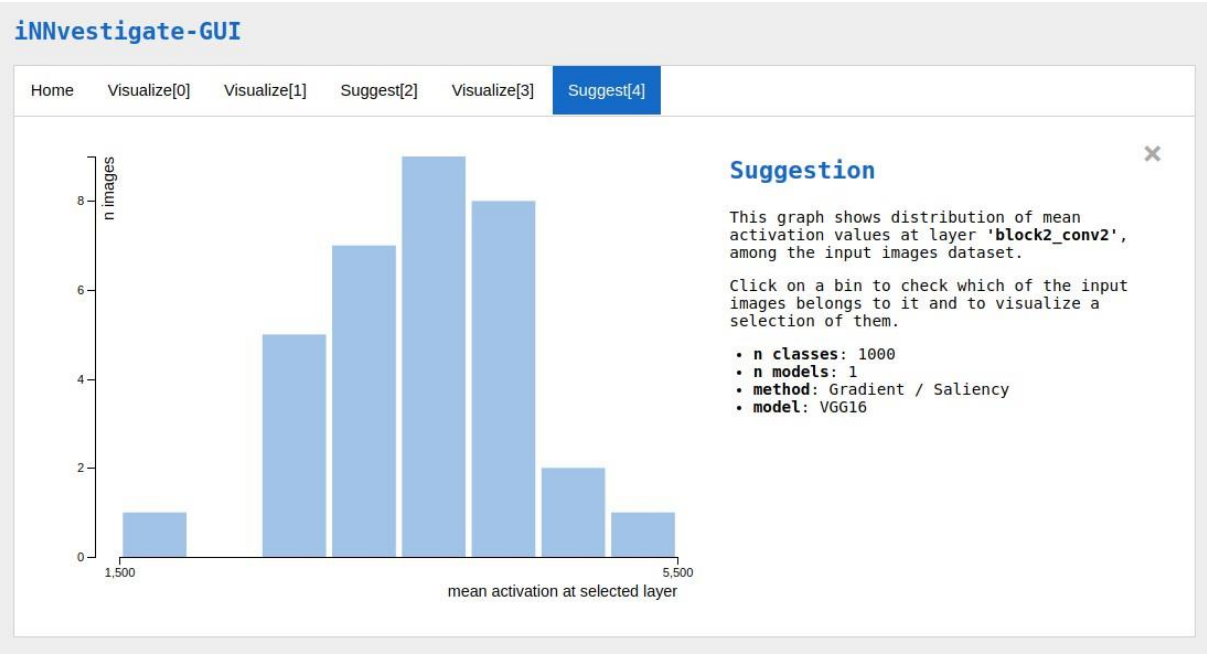

Figura 40 – Esempio di vista creata per il suggerimento di immagini in input basato sull'attivazione media ad un determinato layer. In particolare, sulla sinistra è visibile l'istogramma che mostra la distribuzione dell'attivazione media all'interno del layer selezionato per tutte le immagini in input, mentre sulla destra è presente una descrizione della vista offerta insieme con le informazioni riguardanti gli input forniti.

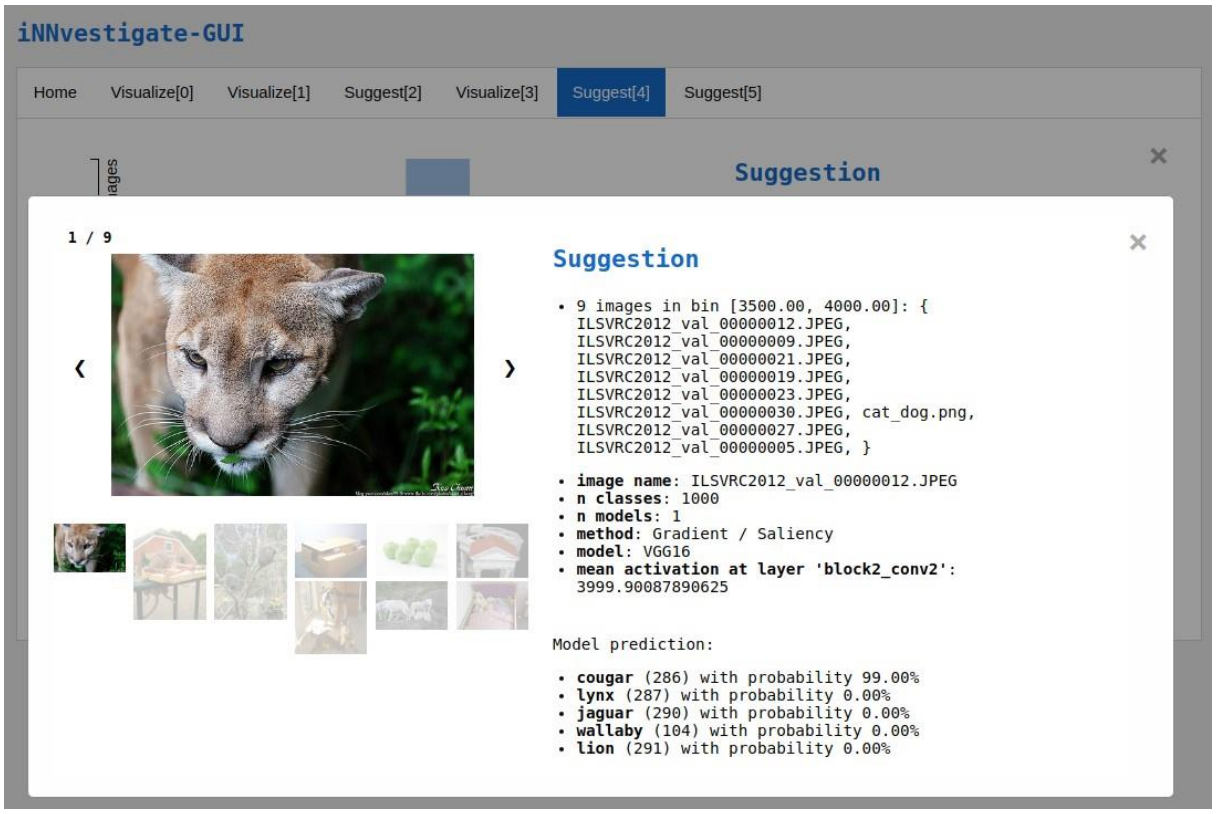

Figura 41 – Esempio di finestra aperta cliccando su uno dei bin dell'istogramma mostrato in figura 40. Sulla sinistra è possibile vedere uno slideshow delle immagini contenute nel bin, che può mostrare al massimo le prime dieci immagini del bin. Sulla destra è invece possibile leggere l'intervallo di valori associato al bin e i nomi delle immagini contenute in esso, insieme con altre informazioni riguardanti l'input fornito, le previsioni effettuate dal modello per quell'immagine, e in questo caso il valore dell'attivazione media al layer selezionato per l'immagine corrente.

sull'ultimo layer convoluzionale del modello se il metodo di visualizzazione scelto è uno tra GradCAM e Guided GradCAM, mentre corrisponde all'ultimo layer del modello in tutti gli altri casi. Premendo il bottone "*Suggest input images*" verrà generata la vista associata al primo tipo di suggerimenti disponibili. In particolare, questa corrisponde ad un istogramma interattivo che mostra la **distribuzione dell'attivazione media all'interno del layer selezionato per tutte le immagini in input**, come è possibile vedere in figura 40. A destra del grafico è mostrata una descrizione della vista corrente e le informazioni riguardanti gli input inseriti.

Ogni bin dell'istogramma rappresenta il numero di immagini in input per le quali l'attivazione media all'interno del layer selezionato ricade nell'intervallo di valori associato al bin stesso, visualizzabile per ogni bin spostandosi con il mouse al di sopra del relativo rettangolo. Cliccando poi su ogni bin è possibile visualizzare una finestra che contiene il nome di tutte le

immagini in input che fanno parte del bin in questione e mostra le prime dieci tra queste immagini, con le relative informazioni associate (figura 41). Tali informazioni riguardano gli input inseriti e le previsioni effettuate dal modello, come avviene per gli output generati dai metodi di visualizzazione, come visto nel capitolo 3, e includono il valore puntuale della metrica che ha fatto sì che ricadessero in un determinato bin, in questo caso cioè sarà presente il valore dell'attivazione media al layer selezionato per ogni immagine visualizzata.

Se oltre a scegliere il layer da utilizzare (o utilizzando quello di default) si seleziona uno specifico neurone tramite la funzione "*By index*" descritta nella sezione 3.2.4, premendo il bottone "*Suggest input images*" verrà generato il secondo tipo di vista per i suggerimenti. In particolare, questa corrisponde ad un istogramma interattivo che mostra la **distribuzione dell'attivazione al neurone selezionato per tutte le immagini in input**, che risulta analogo a quello visto in figura 40. Anche in questo caso alla destra del grafico sono presenti informazioni relativi alla vista corrente e agli input forniti, ed è possibile passare con il mouse sui rettangoli che rappresentano i bin per visualizzare i valori dell'intervallo di attivazioni corrispondente. Cliccando sui bin viene mostrata una finestra analoga a quella di figura 41, contenente i nomi delle immagini che ricadono al suo interno e dove è possibile visualizzare le prime dieci tra queste, con a fianco informazioni aggiuntive, analogamente a quanto visto per il caso precedente. In questo caso però, ogni bin rappresenta il numero di immagini per le quali il valore dell'attivazione al neurone selezionato ricade all'interno dell'intervallo associato al bin, e tale valore viene mostrato, insieme alle altre informazioni, sulla destra di ogni immagine visualizzata nella finestra che si apre cliccando sul bin corrispondente.

Analizzando questi due diversi tipi di grafici, a seconda delle scelte effettuate in input, l'utente ha la possibilità di riconoscere quali immagini, tra quelle contenute nel dataset, attivano maggiormente un determinato layer o un determinato neurone di un layer, e quali invece non corrispondono a stimoli particolarmente interessanti nella situazione analizzata. Nel primo caso, questo può essere utile ad esempio per avviare un'analisi, anche attraverso i metodi di visualizzazione, per comprendere le caratteristiche delle immagini che fanno sì che il modello si attivi maggiormente nel determinato layer o neurone selezionato, ed eventualmente riconoscere

possibili pattern problematici. Identificare invece i casi in cui le attivazioni risultano basse, può ad esempio essere utile per individuare eventuali malfunzionamenti di un modello che non risulta in grado di riconoscere caratteristiche visive in particolari immagini ad un determinato layer o neurone, attraverso l'analisi più approfondita proprio di questi casi.

Questo tipo di suggerimenti però, presuppongono un certo livello di conoscenza del modello in uso per essere efficaci, in modo da utilizzarli per analisi mirate ad uno specifico layer o ad uno specifico neurone a cui l'utente risulta già interessato grazie ad analisi precedenti o per esperienza, il che potrebbe non essere sempre il caso. Fornendo invece più modelli in input, ed utilizzando contemporaneamente le informazioni ricavate dai vari modelli sul dataset in uso, l'individuazione di casi particolari può essere eseguita anche in situazioni in cui non si abbiano altri dati a disposizione per poter usare con efficacia i suggerimenti trattati in questa sezione. Di questo si parlerà di seguito, nella sezione 4.2.2.

4.2.2 Suggerimenti basati su più modelli in input

Se i suggerimenti visti nella sezione 4.2.1 si basano sulle attivazioni generate dalle immagini in input ad un layer o ad un neurone di un singolo modello, le viste generate da iNNvestigate-GUI per i suggerimenti trattati in questa sezione si basano invece sulle classificazioni prodotte dai diversi modelli in input. In particolare, si distinguono due casi: quello in cui il dataset in input contiene immagini tutte appartenenti alla stessa classe e quello in cui invece le immagini appartengono a classi differenti. Per i suggerimenti trattati in questa sezione infatti, i modelli possono essere scelti tra quelli predefiniti disponibili senza bisogno di fornire ulteriori input, mentre per l'utilizzo con modelli custom è richiesto l'inserimento degli input riguardanti la classificazione, ovvero il numero delle classi e il file CLASS_INDEX (come visto nella sezione 3.2.1). Questo perché, una volta scelti i modelli, cliccando sul bottone "*Suggest input images*" si aprirà una nuova finestra (figura 42), dove è possibile specificare la classe alla quale appartengono tutte le immagini in input, che sarà una delle 1000 classi ImageNet qualora la scelta

Figura 42 – Finestra aperta alla pressione del bottone "Suggest input images" quando viene fornito in input al tool un dataset di immagini e più di un modello, predefinto o custom che sia. Nel caso in cui si utilizzino modelli custom è necessario inserire anche l'input riguardante la classificazione poiché in questa finestra è necessario indicare la classe alla quale appartengono tutte le immagini del dataset e dunque premere "Go" per la generazione dei relativi suggerimenti. Nel caso in cui le immagini contenute nel dataset non siano invece tutte appartenenti alla stessa classe basterà cliccare sul bottone "Images belong to different classes" per generare la vista di suggerimenti corrispondente.

ricadesse sui modelli predefiniti disponibili nel tool oppure sarà una delle classi specificate nel CLASS_INDEX fornito con i modelli custom. Se invece il dataset contiene immagini appartenenti a classi differenti, basterà indicarlo cliccando sull'apposito bottone, visibile in figura 42: la scelta dell'una o l'altra opzione darà il via alla generazione delle rispettive viste.

Nel caso in cui il dataset contenga immagini appartenenti a classi differenti, viene generata una vista che corrisponde ad un grafico a dispersione interattivo, che mostra la **relazione tra la confidenza e la coerenza nelle classificazioni effettuate dai modelli selezionati per tutte le immagini in input**, visibile nell'esempio di figura 43. Come nei casi trattati nella sezione 4.2.1, a destra del grafico è presente una spiegazione della vista corrente insieme con i dati inseriti in input. Ciascun punto del grafico rappresenta un'immagine in input, della quale è possibile visualizzare un'anteprima, insieme con il suo nome, passando con il mouse sopra al relativo punto. L'asse delle ordinate rappresenta il numero di classi differenti con il quale i diversi modelli selezionati hanno classificato la stessa immagine in input, dunque i punti nella parte alta del grafico rappresentano i casi per i quali i modelli sono stati più discordanti fra di loro. L'asse

delle ascisse rappresenta invece la confidenza dei modelli, dunque ogni punto è posizionato rispetto a tale asse sulla base dello score delle classificazioni effettuate dai modelli in input.

In particolare, se tutti i modelli classificano la stessa immagine allo stesso modo, la posizione del punto associato a tale immagine rispetto all'asse della confidenza corrisponde alla media tra gli score ottenuti dai vari modelli. Se invece i modelli predicono classi diverse per la stessa immagine, questi vengono raggruppati per classe in output e viene calcolato lo score medio per ciascuna classe, dunque il punto associato a tale immagine sarà posizionato rispetto all'asse della confidenza in corrispondenza del valore medio degli score ottenuti per la classe predetta dal maggior numero di modelli. Infine, nel caso in cui ogni modello effettui una classificazione differente per la stessa immagine, la confidenza del punto associato corrisponderà al valore dello score più alto ottenuto. In questo modo il punto viene sempre posizionato rispetto all'asse della confidenza in modo che rispecchi la sicurezza complessiva dei modelli, sia nel caso in cui

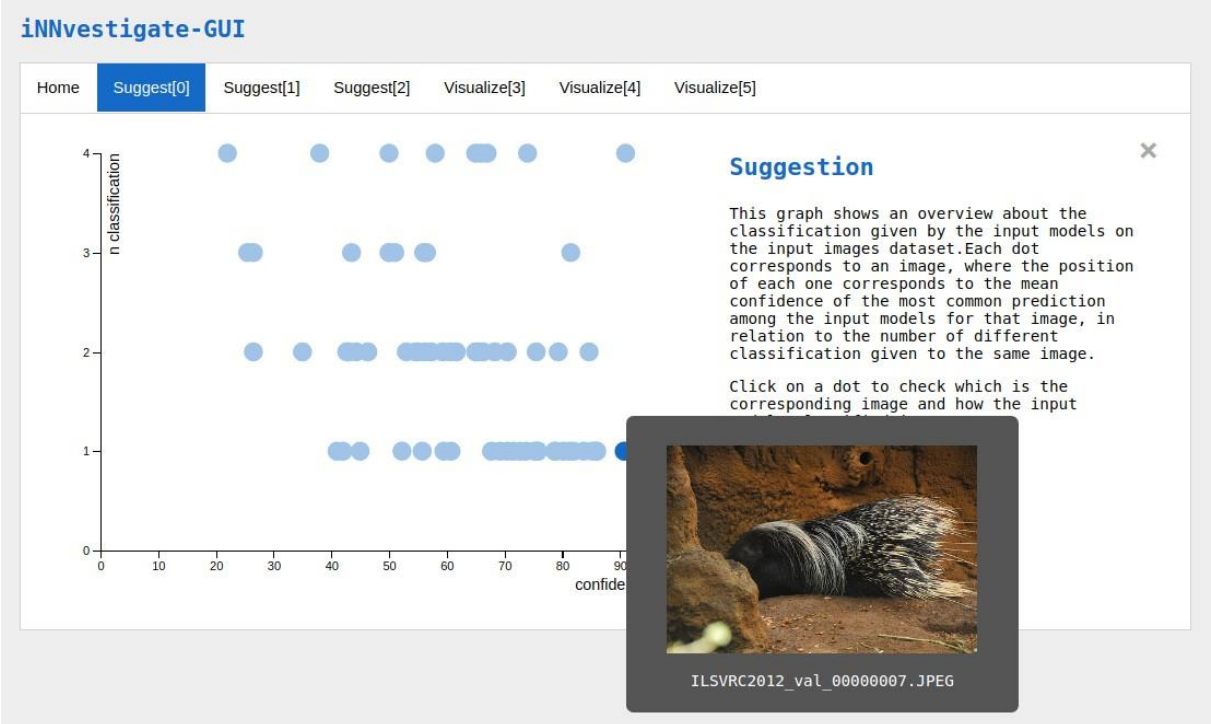

Figura 43 – Esempio di vista creata per i suggerimenti delle immagini in input basata sulla relazione tra la confidenza e la coerenza nelle classificazioni effettuate dai modelli selezionati per tutte le immagini in input. Sulla sinistra è possibile visualizzare il grafico a dispersione corrispondente, insieme con un'anteprima generata al passaggio del mouse su uno dei punti del grafico. Sulla destra invece è possibile vedere alcune delle informazioni relative alla vista offerta, parzialmente coperte dall'anteprima generata.

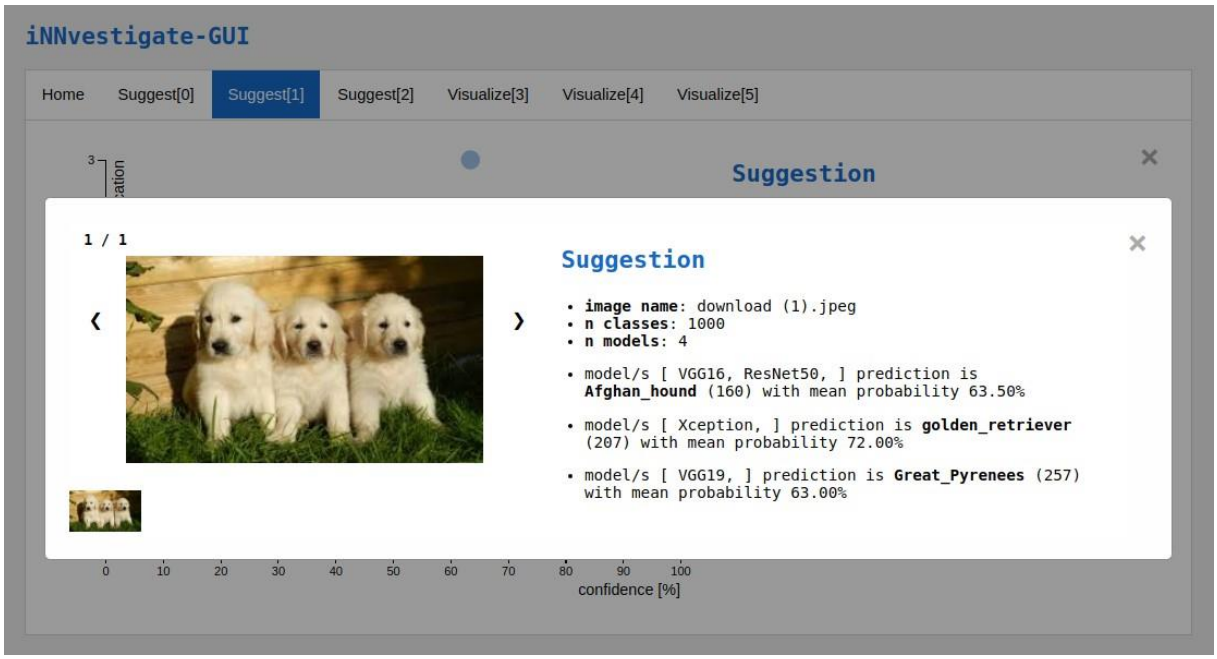

Figura 44 – Esempio di finestra aperta cliccando su uno dei punti del grafico a dispersione mostrato in figura 43. Sulla sinistra è possibile visualizzare l'immagine corrispondente, mentre sulla destra sono mostrate le informazioni relative agli input forniti e alle previsioni dei modelli selezionati. In particolare, i modelli vengono raggruppati per classe predetta e viene mostrato, per ogni gruppo di modelli, la classe predetta e lo score medio delle classificazioni effettuate.

siano tutti d'accordo, mostrandone la "sicurezza media", ma anche in caso siano discordanti, mostrandone la confidenza relativa alla "classe più predetta", dove tali concetti corrispondono appunto al funzionamento appena descritto.

Cliccando sui punti rappresentati nel grafico, è possibile visualizzare una finestra contenente l'immagine corrispondente, insieme con le informazioni associate ad essa e agli altri input, come visto nei casi trattati nella sezione 4.2.1. Come detto, i modelli vengono raggruppati a seconda della classificazione effettuata per la stessa immagine: questi gruppi vengono mostrati, insieme con la classe predetta e con il valore medio degli score ottenuti per tale classe dai modelli del gruppo, tra le informazioni a fianco dell'immagine visualizzabile nella finestra che si apre cliccando sul punto del grafico corrispondente, come si vede in figura 44.

L'ultimo tipo di suggerimenti offerti da iNNvestigate-GUI riguarda infine il caso in cui il dataset in input contenga immagini tutte appartenenti alla stessa classe. Dopo aver inserito la classe di appartenenza nella finestra che si apre alla pressione del bottone "*Suggest input images*", vengono in questo caso generate due viste per i suggerimenti, in due tab differenti. La prima

corrisponde allo stesso tipo appena trattato per il caso in cui le immagini del dataset in input appartengano a classi differenti: questo perché può essere utile avere, anche in questo caso, lo stesso tipo di visione d'insieme sul dataset. La seconda vista invece, corrisponde ad un istogramma interattivo simile a quelli trattati nella sezione 4.2.1 come ad esempio quello visibile in figura 40, che mostra la **distribuzione della distanza dello score medio delle classificazioni effettuate dai modelli selezionati, rispetto alla classe di appartenenza del dataset, per tutte le immagini in input**. Anche in questo caso, sulla destra sono presenti informazioni riguardanti la vista corrente e i dati inseriti in input, ed è possibile passare con il mouse sopra ogni bin dell'istogramma per visualizzare gli estremi dell'intervallo di valori corrispondente.

Ogni bin rappresenta il numero di immagini che sono state classificate dai modelli in input con uno score che dista in media di un valore che ricade nell'intervallo associato al bin stesso, rispetto alla previsione ideale corrispondente ad uno score del 100% per la classe inserita in input. Per ogni immagine infatti, utilizzando la formula della distanza coseno[23], viene calcolata la distanza tra la classificazione prodotta da uno dei modelli in input e la classificazione ideale, ovvero tra il vettore che rappresenta la previsione di quel modello per quell'immagine e il vettore che corrisponde alla previsione ideale vista sopra, e viene fatta la media delle distanze così ricavate per ciascuno dei modelli in input. Il valore di tale media viene utilizzato per posizionare l'immagine corrispondente all'interno del bin corretto nell'istogramma.

Cliccando sui bin è possibile visualizzare una finestra contenente il nome di tutte le immagini che ne fanno parte e che mostra le prime dieci di queste immagini, con associate le relative informazioni e i dati riguardanti gli input, similmente a quanto visto nella sezione 4.2.1 e a quanto rappresentato in figura 41. In più, viene mostrato il valore medio della distanza degli score dalla previsione ideale per i modelli in input, ottenuto come spiegato sopra, ed è inoltre presente l'elenco dei modelli che, per tale immagine, hanno in effetti predetto la classe alla quale tutte le immagini in input appartengono, specificata in input.

[23] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html

Si nota subito che il tipo di suggerimenti e di grafici associati trattati in questa sezione, si basano su informazioni maggiormente elaborate, avendo a disposizione più dati a partire dai diversi modelli utilizzati in input. Questo li rende immediatamente più efficaci laddove non si abbiano particolari conoscenze sui dettagli dei modelli in uso, situazione in cui sarebbe più difficoltoso l’utilizzo dei suggerimenti trattati nella sezione 4.2.1, che richiedono invece di selezionare un layer o un neurone specifico per fornire risultati efficaci. A partire dalle viste proposte in questa sezione invece, è facile riconoscere casi interessanti, ad esempio posizionandosi alle estremità dei grafici a dispersione, o ancora, avendo a disposizione un dataset contenente immagini tutte appartenenti alla stessa classe, è possibile utilizzare l’istogramma offerto concentrandosi ad esempio sui bin alle due estremità, che contengono le immagini sulle quali i modelli hanno effettuato previsioni molto vicine o molto lontane alla classe specificata in input.

Una volta individuati i casi interessanti con questo tipo di viste, è possibile approfondire l’analisi su di essi utilizzando uno o più modelli specifici, attraverso i metodi di visualizzazione forniti ma anche eventualmente attraverso i suggerimenti trattati nella sezione 4.2.1, nel caso ci si stia infine focalizzando su un solo modello. Appare perciò ancora più evidente l’obiettivo di integrazione dei processi principali legati alla visualizzazione delle reti neurali, su cui si è incentrato lo sviluppo di iNNvestigate-GUI, cercando di definire un **workflow** che mancava in questo ambito, o che comunque risultava limitato all’esperienza personale.

5 Efficacia ed usabilità

L'obiettivo primario che ha guidato lo sviluppo del tool presentato in questa tesi, iNNvestigate-GUI, è sempre stato la creazione di uno strumento che permettesse l'integrazione del processo di visualizzazione di una rete neurale durante il suo sviluppo o il suo utilizzo, nella maniera più semplice ed immediata possibile, per il maggior numero possibile di utenti e casistiche. Come spiegato nel corso di questo lavoro, la categoria di reti sulla quale è incentrato iNNvestigate-GUI sono le CNN per la classificazione di immagini, e il framework in uso per le operazioni di Deep Learning è Keras con backend TensorFlow: questo lo rende disponibile per un gran numero di problemi in molti diversi ambiti. L'interfaccia grafica è stata sviluppata in modo da essere di facile comprensione e immediato utilizzo, senza richiedere all'utente lo studio di nuovi processi né di conoscere approfonditamente tutte le tecniche di visualizzazione disponibili.

Provvedendo sia al suggerimento delle immagini da utilizzare per l'analisi, come spiegato nel capitolo 4, sia all'integrazione di numerose tecniche di visualizzazione per effettuare le analisi stesse, come spiegato nel capitolo 3, il tool mette a disposizione un vero e proprio workflow, che può essere facilmente inglobato nel lavoro di qualsiasi progettista ed utilizzatore di reti neurali, equipaggiandolo in ogni caso di funzionalità interessanti. Nonostante questo, sicuramente il target principale di iNNvestigate-GUI risulta essere **l'utenza meno esperta**, che ad esempio non ha ancora a disposizione le conoscenze necessarie per poter sostituire le funzionalità offerte dal tool con codice sviluppato ad hoc, o che non ha ancora avuto modo di sfruttare i vantaggi offerti dalle tecniche di visualizzazione delle reti. Per questo motivo, si è scelto di focalizzarsi proprio su questo tipo di utenza per **valutare l'efficacia e l'usabilità del tool** sviluppato, coinvolgendo un gruppo di studenti in una sessione di test.

Tra tutte le operazioni per le quali può essere utilizzato il workflow offerto da iNNvestigate-GUI, ne sono state identificate due per le quali il tool risulta più utile, che corrispondono ai **due task proposti** ai partecipanti, che verranno trattati nel paragrafo 5.1 insieme ai dettagli del test svolto. Nel paragrafo 5.2 invece, si analizzeranno i risultati ottenuti.

5.1 Il test svolto

5.1.1 Il metodo utilizzato

Per lo svolgimento del test di validazione di iNNvestigate-GUI, è stato coinvolto un gruppo di nove studenti di Ingegneria del Politecnico di Torino, senza ulteriori requisiti se non la conoscenza di cos'è e come funziona una CNN per la classificazione di immagini. Lo scopo del test svolto è stato quello di verificare l'effettiva utilità e l'efficacia di iNNvestigate-GUI nell'esecuzione di due task, trattati nella sezione 5.1.2, soprattutto in relazione alla possibilità di effettuare task dello stesso tipo in assenza di questo tool, e di misurarne l'usabilità da parte di utenti non esperti. Per verificare l'efficacia, una serie di domande guidavano i partecipanti in situazioni che si sarebbero potute verificare in uno scenario reale, in modo da ottenere risposte sull'utilizzo che avrebbero fatto in quel caso delle funzionalità disponibili in iNNvestigate-GUI, mentre per misurare l'usabilità sono state incluse le domande del SUS, **System Usability Scale** [50], che fornisce un approccio rapido per ottenere una misura quantitativa dell'usabilità di un sistema.

Ai partecipanti è stata fornita una guida rapida all'utilizzo di iNNvestigate-GUI, che ne illustrava le principali funzionalità, dunque il **questionario**, consultabile nell'appendice 1, con le domande da seguire per lo svolgimento dei task. Vista la poca esperienza dei partecipanti, l'unica altra forma di assistenza consentita è stata la possibilità di chiedere aiuto riguardo alla comprensione delle domande del questionario, o ad eventuali malfunzionamenti del tool. Non era consentito chiedere aiuto riguardo alle modalità di utilizzo delle funzionalità del tool per rispondere alle domande del questionario.

I task sono stati svolti in una sessione di un'ora e mezza, utilizzando un browser a scelta dei partecipanti su macchine client aventi come sistema operativo Windows 7, connettendosi ad istanze AWS di tipo *p2.xlarge*[24], dotate di una GPU Nvidia Tesla K80, con Keras 2.2.4 e TensorFlow 1.13.2 installato ed iNNvestigate-GUI in esecuzione.

5.1.2 I task proposti

La scelta dei task da proporre ai partecipanti per il test di validazione di iNNvestigate-GUI è stata dettata da diversi fattori, primo fra tutti il tempo a disposizione per il test. Una possibile valutazione del tool poteva essere effettuata chiedendo ai partecipanti di sviluppare uno o più modelli di reti neurali personali, per risolvere lo stesso problema di classificazione, fornendo a tutti lo stesso dataset di immagini in input e dunque far loro utilizzare iNNvestigate-GUI per identificare eventuali problemi o malfunzionamenti dei modelli sviluppati. Questa è una delle principali possibilità di utilizzo delle funzionalità messe a disposizione dal tool, ma la durata del test ha reso questo tipo di valutazione impossibile.

Dunque, si è scelto di valutare il tool mediante l'utilizzo di modelli predefiniti, simulando una situazione in cui, tra più reti a disposizione, l'utente possa verificare il comportamento di ognuna di esse con il dataset in uso, per individuare eventuali problemi con i modelli o con le immagini in input. iNNvestigate-GUI offre la possibilità di utilizzare più modelli contemporaneamente, il che è necessario per utilizzare alcuni strumenti per l'analisi del dataset (come quelli esposti nella sezione 4.2.2) ma può essere utile per confrontare le visualizzazioni ottenute a partire da reti diverse contemporaneamente. Al crescere del numero dei modelli in input però, la generazione delle visualizzazioni richiede tempi sempre maggiori, e per via della durata limitata del test, non è stato possibile lasciare l'utente libero di scegliere i modelli da utilizzare. Sono stati quindi imposti tre modelli per lo svolgimento dei due task, piuttosto differenti tra loro in modo da simulare al meglio una situazione reale, ovvero **VGG16**, **ResNet50** e **Xception**.

[24] https://aws.amazon.com/it/ec2/instance-types/p2/

La scelta di definire i modelli in uso dai partecipanti era anche un requisito necessario per poter verificare agevolmente le risposte alle domande del questionario.

Considerando il fatto che i partecipanti erano tutti utenti non esperti, oltre all'impostazione dei modelli predefiniti da utilizzare, sono stati proposte quattro tecniche di visualizzazione delle attivazioni tra quelle proposte dal tool: **Gradient / Saliency**, **Guided Backpropagation**, **GradCAM** e **LRP-z**. Queste sono state scelte poiché non richiedono la conoscenza dei dettagli di funzionamento per essere utilizzate ed inoltre forniscono output molto differenti tra loro, in modo da dare la possibilità ai partecipanti di esplorare tutte le categorie di visualizzazioni offerte dal tool, e verificare quale ritenessero più utile per lo svolgimento dei task proposti.

Una volta selezionati i modelli e le tecniche di visualizzazione da utilizzare, si è scelto di valutare il tool tramite due dei workflow per il quale può essere utilizzato, ciascuno basato su un diverso tipo di dataset di immagini in input.

Il primo task consisteva nel **valutare quali caratteristiche delle immagini in input (ad esempio quali parti dell'oggetto o della scena raffigurati) vengono utilizzate per la classificazione da parte di reti differenti**. Lo scenario prevede quindi un **dataset contenente immagini appartenenti a qualsiasi classe** tra quelle predicibili dai modelli in uso, e i modelli stessi da utilizzare. L'operazione richiesta può dunque essere svolta facilmente con iNNvestigate-GUI, andando prima ad utilizzare la funzionalità "*Suggest input images*" sul dataset e i modelli scelti, per identificare quali sono le immagini più interessanti sulle quali focalizzare l'analisi. In particolare, come spiegato nella sezione 4.2.2, in questo caso viene generato un grafico a dispersione che mostra la relazione tra la confidenza e la coerenza delle classificazioni prodotte dai modelli selezionati per tutte le immagini in input, rappresentate come punti nel grafico. In questo modo è facile rilevare i casi più interessanti andando a verificare la parte di grafico dove si concentrano i punti e soffermandosi soprattutto sui casi al di fuori di tale area, o comunque sui casi alle estremità del grafico, come spiegato nella sezione 4.2.2. Una volta selezionate le immagini, è sufficiente analizzarle utilizzando le tecniche di visualizzazione disponibili con la funzionalità "*Visualize*", trovando dunque quella che produce rappresentazioni migliori per identificare le caratteristiche visive utilizzate dai modelli per la classificazione, come richiesto.

Il dataset utilizzato per questo task è stato scelto a partire dalle immagini della ILSVRC2012 di ImageNet, scegliendo un sottoinsieme che non fosse troppo grande, per non avere tempi di esecuzione troppo elevati, ma che fosse sufficiente per poter apprezzare le potenzialità del tool. In particolare, sono state quindi selezionate cento immagini in modo che comprendessero casi facilmente identificabili, nella vista proposta da "*Suggest input images*" con i modelli scelti, per rispondere alle domande del task 1, analizzate nel paragrafo 5.2.

Il secondo task invece, consisteva nel **verificare se reti neurali differenti si basino su caratteristiche dell'immagine diverse per riconoscere scene o oggetti di una determinata classe**. In questo scenario dunque è previsto un **dataset di immagini appartenenti alla stessa classe**, i modelli da utilizzare e la classe alla quale appartengono le immagini in input. Anche in questo caso il workflow prevede l'utilizzo della funzionalità "*Suggest input images*" che, come spiegato nella sezione 4.2.2, una volta selezionata la classe delle immagini in input genererà, oltre al grafico a dispersione come per il primo task, anche l'istogramma che mostra la distribuzione della distanza dello score medio delle classificazioni effettuate dai modelli selezionati, rispetto alla classe di appartenenza del dataset, per tutte le immagini in input. In questo modo è possibile identificare i casi più interessanti sui quali effettuare le analisi, andando ad esempio a concentrarsi sulle immagini per le quali le classificazioni risultano più vicine alla classe in input, o più lontane. Come per il primo task dunque, non resta altro che analizzare le immagini selezionate con i metodi di visualizzazione disponibili, verificando quale risulta il più utile per trovare le caratteristiche visive corrispondenti ad una determinata classe per i diversi modelli, come richiesto. Il dataset che è stato proposto per questo task contiene quindi 15 immagini casuali che raffigurano cani appartenenti alla razza Golden Retriever, dunque alla classe ImageNet *207 - golden_retriever*, per le quali è stato verificato che il comportamento dei modelli scelti permettesse di identificare, nelle viste proposte da "*Suggest input images*", le immagini sulle quali effettuare le analisi per rispondere alle domande del task 2. In questo caso, non era necessario avere un gran numero di immagini come per il primo task, poiché le domande del task 2 sono concentrate non tanto sulla vista legata al grafico a dispersione, della quale si apprezza l'utilità soprattutto per dataset più grandi, quanto invece sulla vista che contiene l'istogramma, in modo da sfruttare le informazioni aggiuntive offerte dal tool proprio per il fatto che le immagini

appartengono tutte alla stessa classe. Anche in questo caso, ulteriori informazioni sono disponibili nel paragrafo 5.2, dove si analizzano le risposte al questionario.

5.2 Risultati

5.2.1 Primo task

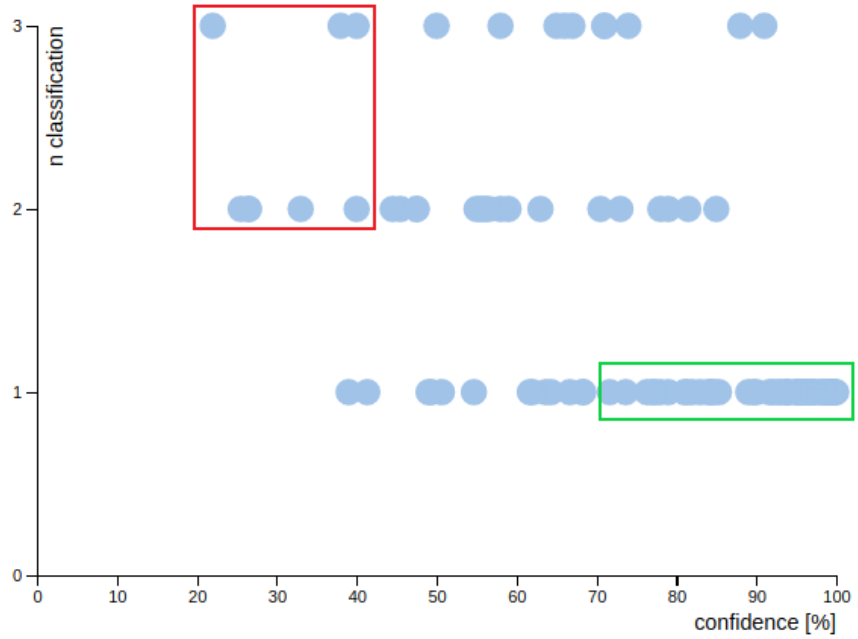

Figura 45 – Grafico a dispersione generato dalla funzionalità "Suggest input images" descritta nel paragrafo 4.2.2. In particolare, è segnalato in verde il riquadro corrispondente alla zona dove i partecipanti del test hanno correttamente individuato le immagini corrispondenti a classificazioni confidenti e coerenti, mentre è segnalato in rosso il riquadro relativo alle immagini classificate con più insicurezza e incoerenza dai modelli in input. Inoltre, è possibile verificare, visualizzando l'andamento generale dei punti del grafico, che i modelli forniscono complessivamente classificazioni piuttosto confidenti e coerenti tra di loro, e che il dataset risulta dunque adeguato ai modelli in uso: per migliorare i modelli e il dataset possono essere analizzati proprio i casi particolari rilevabili grazie alla vista offerta da grafici come questo.

Per quanto riguarda il primo task, riguardante un dataset con immagini appartenenti a classi diverse, inizialmente i partecipanti hanno dovuto focalizzarsi sull'individuazione di immagini per le quali i modelli risultassero confidenti e coerenti tra loro sulla classificazione effettuata. Tutti i partecipanti hanno segnalato correttamente immagini per le quali i modelli in input hanno effettuato la stessa classificazione, come si vede in figura 45, segnalando nella maggior parte dei casi immagini relative a classificazioni con confidenza maggiore al 99%, dimostrando che è

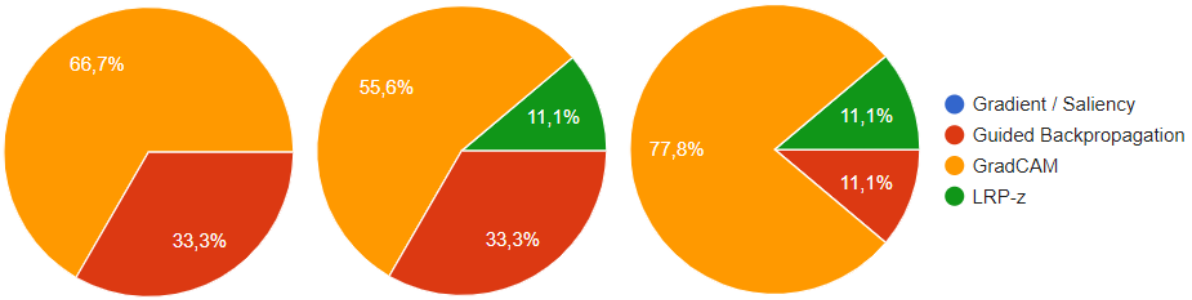

Figura 46 – Diagrammi a torta che mostrano le percentuali dei metodi di visualizzazione preferiti dai partecipanti, tra i quattro suggeriti, durante lo svolgimento del task 1 per l'individuazione delle caratteristiche visive delle immagini sulle quali i modelli utilizzati risultavano più confidenti e coerenti (a sinistra), più insicuri e incoerenti (al centro) ed infine (a destra) per l'individuazione di caratteristiche visive di ulteriori immagini ritenute interessanti dai partecipanti, grazie alla vista del grafico a dispersione (figura 45). Come previsto, il metodo preferito in generale è risultato essere GradCAM, ed in seconda posizione Guided Backpropagation, mentre Gradient / Saliency si è confermato non essere adatto a questo tipo di task.

stato semplice individuare i casi richiesti basandosi sulla vista fornita dal grafico. Dunque, i partecipanti hanno dovuto individuare immagini per le quali i modelli risultassero incoerenti e poco confidenti sulle classificazioni effettuate, ed in questo caso tutte le risposte rappresentano correttamente casi situati all'estremo opposto del grafico. In questo caso inoltre, l'immagine segnalata maggiormente corrisponde proprio all'estremità del grafico in alto a sinistra, per la quale i tre modelli hanno effettuato tre diverse classificazioni, con score massimo 22%. Maggiori dettagli sono disponibili nella figura 45.

Per l'individuazione delle caratteristiche visive utilizzate per la classificazione effettuata dai diversi modelli, il metodo preferito è stato GradCAM in ogni caso, come si vede in figura 46. Si può analizzare ad esempio l'immagine identificata dalla maggior parte dei partecipanti per rispondere alla prima domanda, che chiedeva di individuare almeno un'immagine per la quale i modelli risultassero confidenti e coerenti tra loro nelle classificazioni ottenute, ovvero la ILSVRC2012_val_00000074, che raffigura una giostra a carosello. Dalla figura 47, che contiene le relative rappresentazioni GradCAM, è facile vedere come le caratteristiche visive utilizzate dai modelli, che classificano tutti correttamente e allo stesso modo l'immagine in questione, si focalizzino principalmente sul bastone di supporto del cavallino tipico delle giostre a carosello e sulle luci della giostra. Non si può dire che i tre modelli hanno evidenziato esattamente le stesse caratteristiche, ma i risultati GradCAM sono molto simili tra loro. Alla domanda "*le caratteristiche visive utilizzate dai modelli per la classificazione sono state le stesse per tutti*

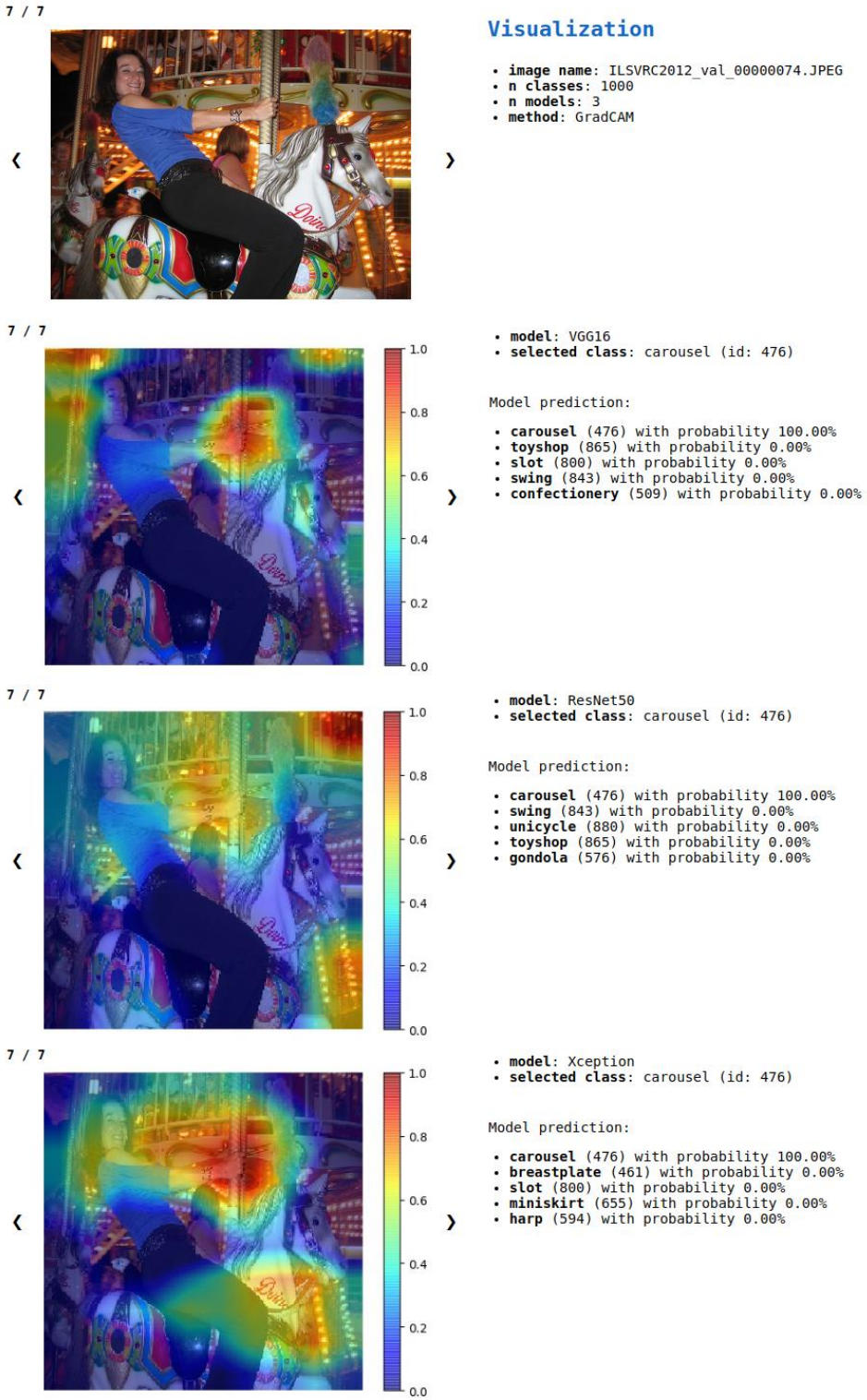

Figura 47 – Rappresentazioni prodotte dal metodo GradCAM sull'immagine identificata maggiormente dai partecipanti al test per rispondere alla domanda 1 (vedi appendice 1).

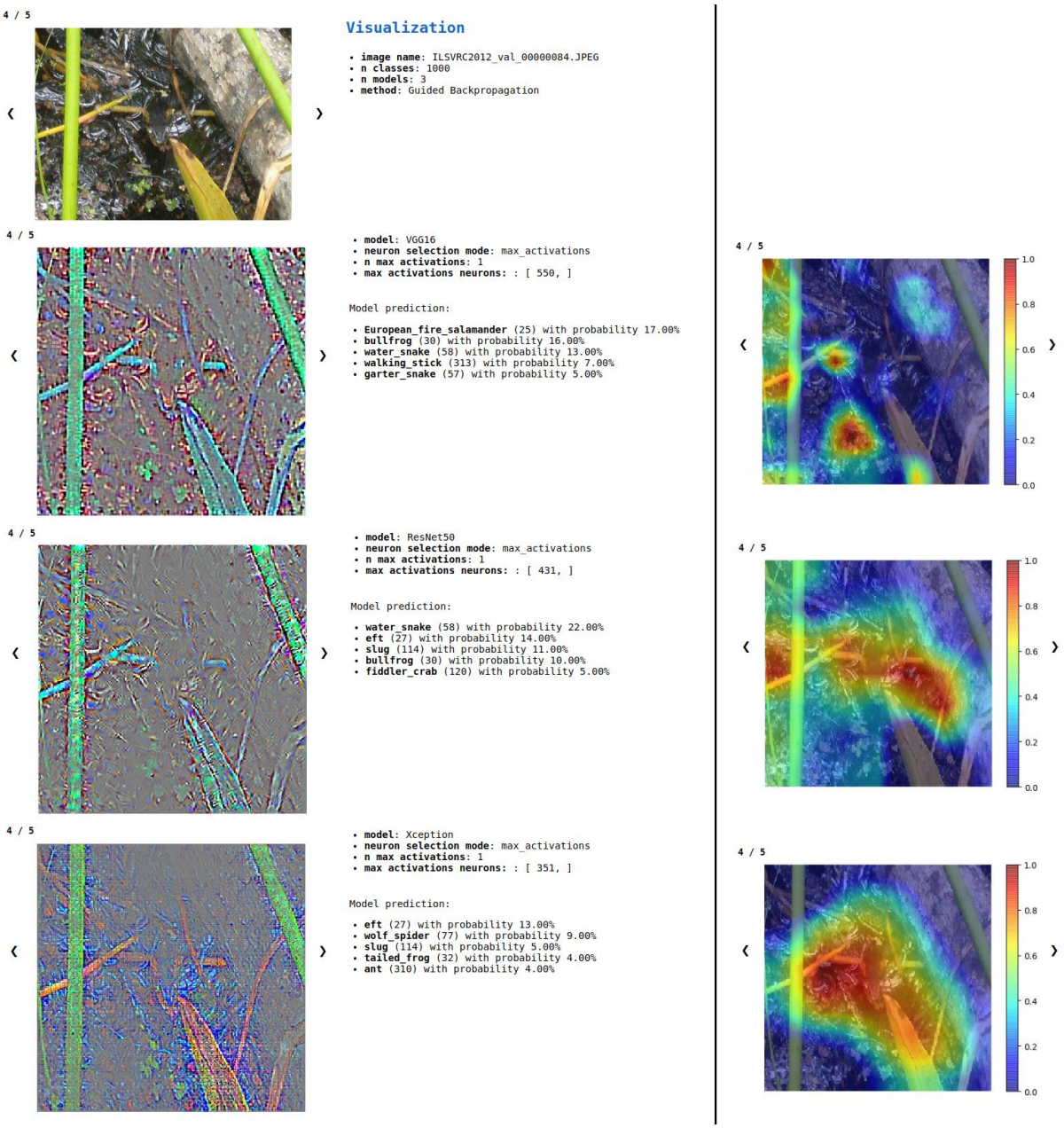

Figura 48 – Rappresentazioni prodotte dai metodi Guided Backpropagation (a sinistra) e GradCAM (a destra) per l'immagine identificata dalla maggior parte dei partecipanti per rispondere alla domanda 4 del questionario (vedi appendice 1). I risultati delle classificazioni sono visibili al centro (per le visualizzazioni GradCAM è mostrato solo l'output del metodo, e sono state tagliate le informazioni aggiuntive della visualizzazione fornite da iNNvestigate-GUI per motivi di spazio).

i modelli?", i partecipanti che hanno individuato l'immagine dell'esempio hanno risposto 4, su una scala da 1, corrispondente alla risposta "*No*", a 5, corrispondente alla risposta "*Si*", che risulta in linea con quanto ci si aspettava. La maggior parte dei partecipanti ha utilizzato proprio GradCAM per rispondere correttamente a questa domanda.

Analogamente, si può considerare l'immagine identificata dalla maggior parte dei partecipanti per rispondere alla domanda 4 (vedi appendice 1), che richiedeva di identificarne almeno una per la quale i modelli risultavano poco confidenti e incoerenti nelle classificazioni risultanti, ovvero la ILSVRC2012_val_00000084, che raffigura un serpente d'acqua (difficile da distinguere anche da un umano). Anche in questo caso la visualizzazione GradCAM aiuta maggiormente, come è possibile vedere in figura 48 (che mostra anche i risultati della classificazione, che confermano che l'immagine individuata è corretta per la domanda 4 del questionario), dove viene messa a confronto con la rappresentazione Guided Backpropagation. In particolare, la risposta alla domanda "*le caratteristiche visive utilizzate dai modelli per la classificazione sono state le stesse per tutti i modelli?*" relativa alle immagini identificate per la domanda 4, è stata 3 (nella stessa scala descritta sopra, da 1 a 5, da "*No*" a "*Si*") per chi ha usato il metodo GradCAM, ed in effetti le rappresentazioni mostrano che, soprattutto per il modello VGG16, le caratteristiche visive non combaciano del tutto tra i diversi modelli, mentre è stata più discordante per i partecipanti che hanno usato Guided Backpropagation, con risposte quasi "casuali" (le risposte sono state 2, 3 e 5, nella stessa scala da 1 a 5 di cui sopra). Questo evidenzia ancora una volta come le rappresentazioni GradCAM siano le più adatte per questo tipo di task e, come anticipato sono state appunto le più utilizzate dai partecipanti per rispondere a queste domande, come si vede in figura 46.

Sempre per il primo task era previsto che i partecipanti segnalassero un'ulteriore immagine ritenuta interessante da analizzare: per rispondere a questa domanda la strada seguita da tutti è stata quella di continuare ad utilizzare il grafico a dispersione proposto da "*Suggest input images*" in combinazione, per quasi tutti i partecipanti, con il metodo di visualizzazione GradCAM, come si vede in figura 46. Questo ha portato tutti ad individuare immagini con caratteristiche

visive che costituiscono in effetti casi interessanti sui quali approfondire l'analisi, segnalando in particolare:

- immagini sulle quali i modelli erano in forte disaccordo, risultando in un modello molto confidente sulla previsione effettuata e gli altri molto insicuri, come ad esempio l'immagine ILSVRC2012_val_00000088, analizzata in figura 49;
- immagini sulle quali i modelli hanno effettuato la stessa classificazione, focalizzandosi però su caratteristiche visive molto diverse da modello a modello, come ad esempio l'immagine ILSVRC2012_val_00000001, analizzata in figura 50;
- immagini sulle quali i modelli erano in disaccordo su classificazioni a confidenza bassa che si sono rivelate errate per la mancanza di una classe adatta nei modelli in uso, come ad esempio l'immagine ILSVRC2012_val_00000008 analizzata in figura 51, oppure per la mancanza nell'immagine, o per l'incapacità di riconoscimento da parte dei modelli, di sufficienti caratteristiche visive per far propendere la scelta verso una sola classe, come l'immagine ILSVRC2012_val_00000048 analizzata in figura 52.

L'importanza del processo di identificazione di queste casistiche risulta ancora più evidente analizzando le risposte ad un'altra domanda riguardante questo primo task, che chiedeva di dare un giudizio generale sull'efficacia dei modelli in uso con il dataset proposto, sulla base dei risultati ottenuti. Quasi tutti i partecipanti hanno segnalato che i modelli sono risultati piuttosto confidenti e coerenti tra di loro nella maggior parte dei casi e che il dataset è risultato essere adeguato all'utilizzo con questi modelli. L'andamento del grafico a dispersione, mostrato in figura 45, e i risultati delle classificazioni effettuate dai modelli sulle immagini proposte, confermano quanto affermato dai partecipanti. In questo scenario, la facilità di individuazione di queste casistiche, grazie all'utilizzo delle funzionalità proposte da iNNvestigate-GUI, è evidentemente un grande valore aggiunto. La maggior parte dei partecipanti inoltre, ha confermato quello che si è già detto in precedenza in questo lavoro in merito al target principale di iNNvestigate-GUI, ovvero corrispondente all'utenza meno esperta, affermando infatti di non ritenersi in grado di giungere ai risultati richiesti per rispondere alle domande del questionario e qui esposti, senza l'utilizzo del tool, come mostrato in figura 53.

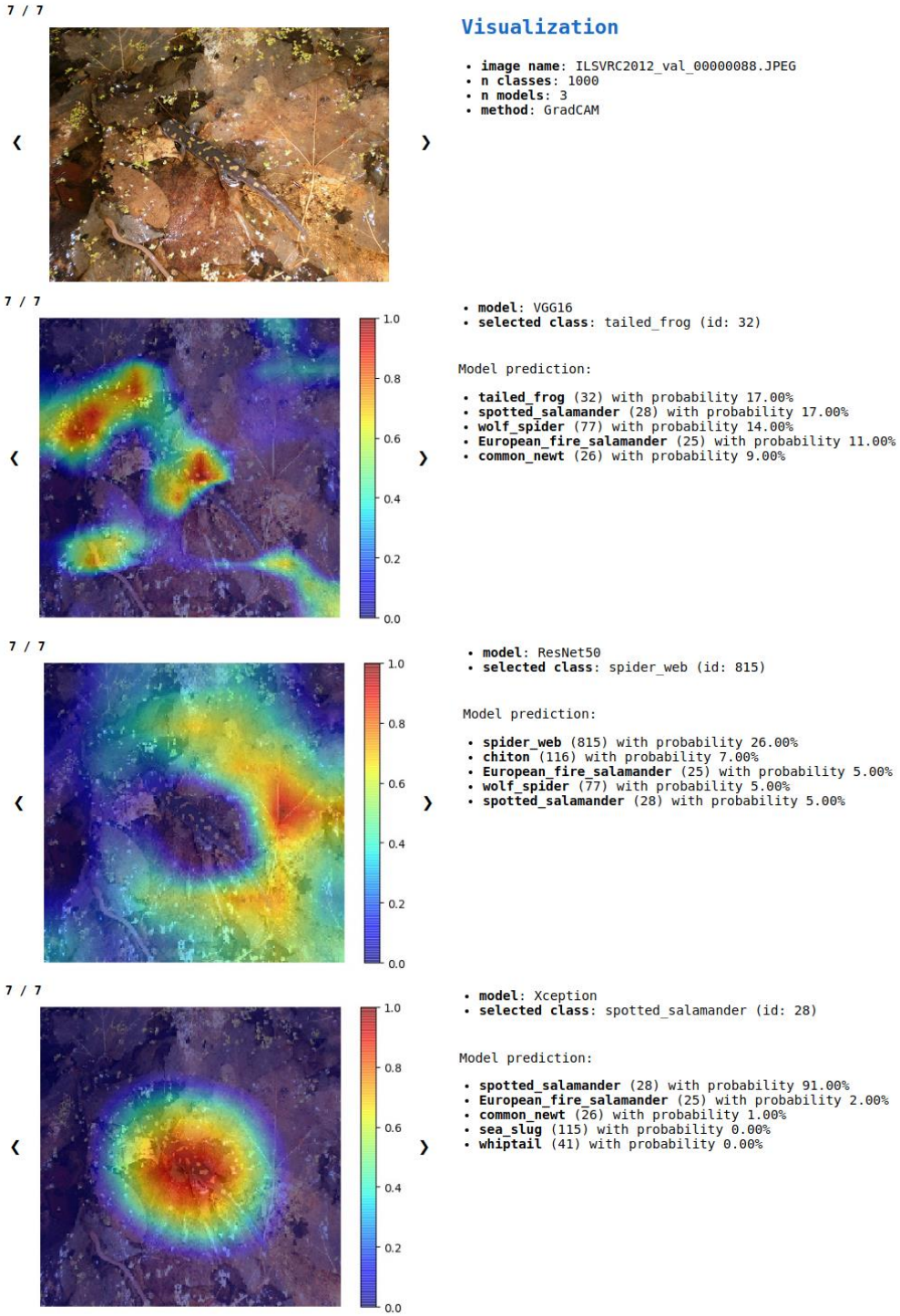

Figura 49 – Rappresentazioni GradCAM di una delle immagini identificata come caso interessante durante il test per i tre modelli in uso: in particolare è facile notare come il modello Xception si sia focalizzato correttamente sulla parte dell'immagine più appropriata per effettuare la classificazione, ottenendo in effetti una confidenza molto alta nella classe predetta, che risulta corretta, mentre gli altri modelli predicono classi sbagliate, oltretutto con uno score molto basso, a causa della mancata individuazione delle corrette feature nell'immagine di input.

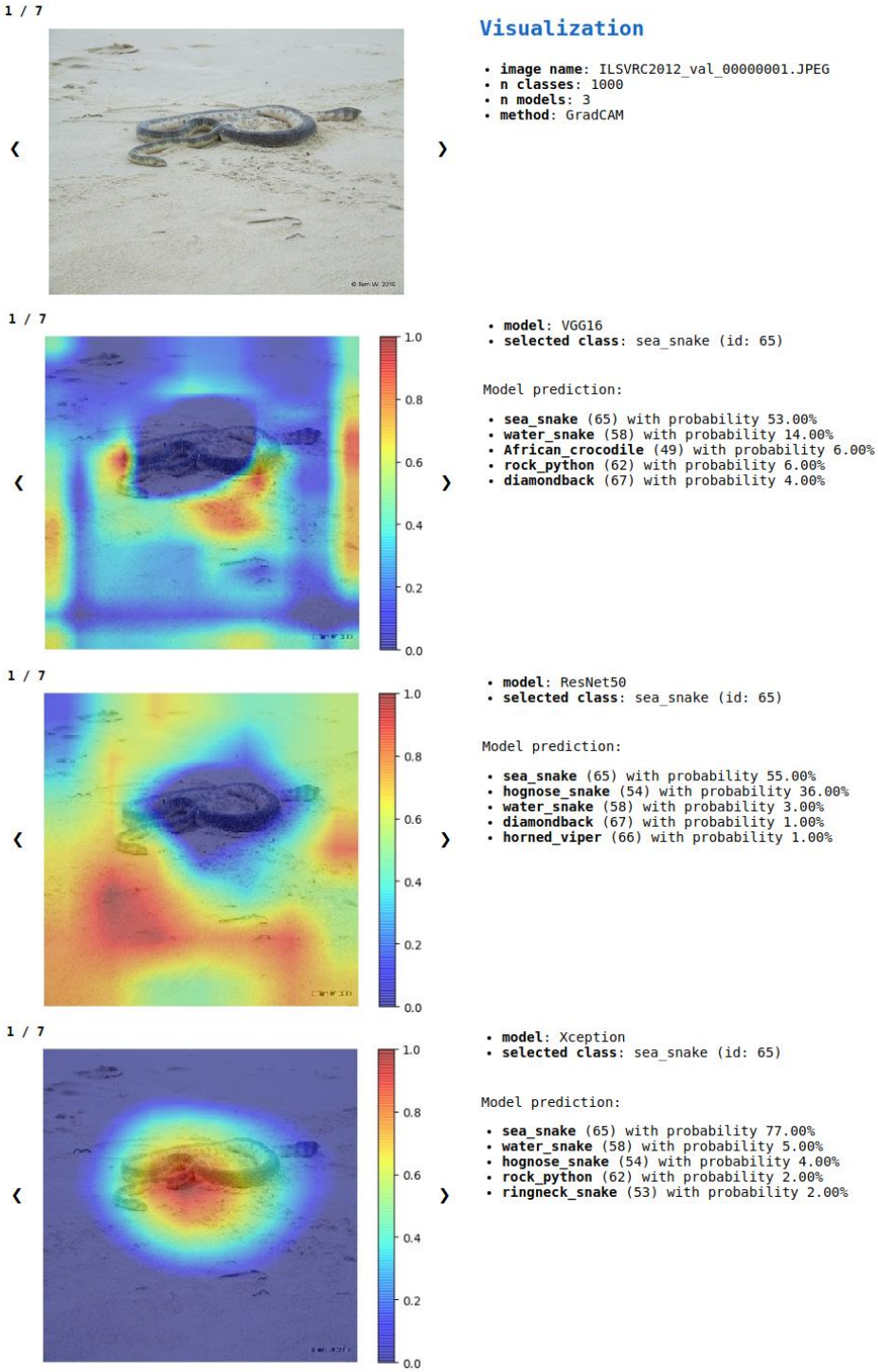

Figura 50 – Rappresentazioni GradCAM di una delle immagini identificata come caso interessante durante il test per i tre modelli in uso: in particolare si nota come i modelli hanno effettuato la stessa classificazione corretta, ma si sono focalizzati su caratteristiche dell'immagine molto differenti fra di loro. La differenza nelle caratteristiche evidenziate porta i modelli ad essere più o meno confidenti: ad esempio il modello Xception predice sea_snake con uno score più alto rispetto agli altri modelli, ed in effetti le caratteristiche dell'immagine su cui si focalizza sono più appropriate rispetto agli altri modelli.

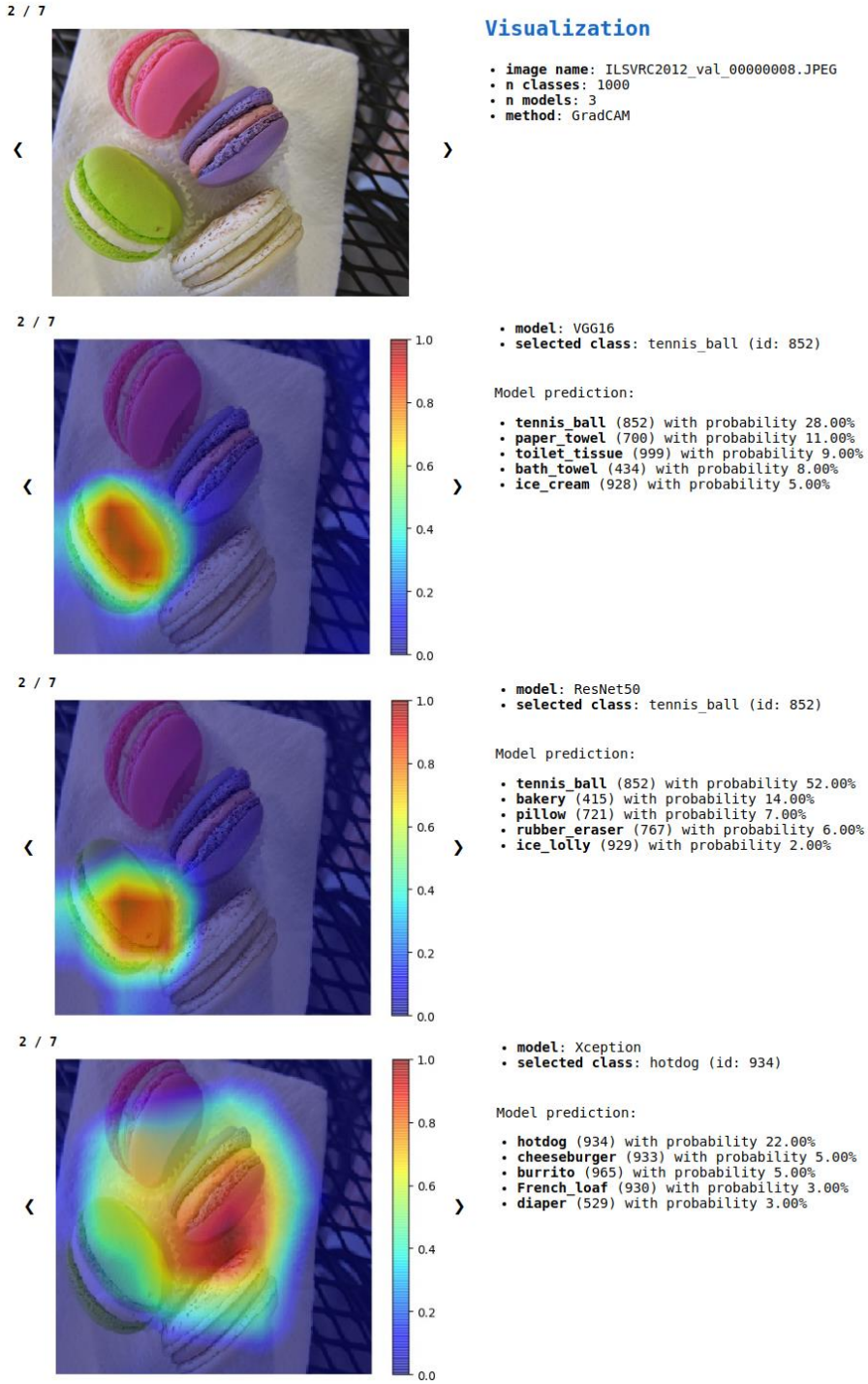

Figura 51 – Rappresentazioni GradCAM di una delle immagini identificata come caso interessante durante il test per i tre modelli in uso: in particolare si nota come i modelli abbiano effettuato classificazioni diverse, tutte sbagliate e caratterizzate da una confidenza piuttosto medio-bassa. L'insicurezza e incoerenza tra i modelli può essere ricondotta al fatto che l'immagine in questione rappresenti dei macarons, ma tra le classi ImageNet questa non è presente, dunque, sulla base delle caratteristiche riconosciute dai modelli, questi predicono la classe più appropriata, e vedendo le rappresentazioni GradCAM dei tre modelli, non stupiscono le classi prodotte in output.

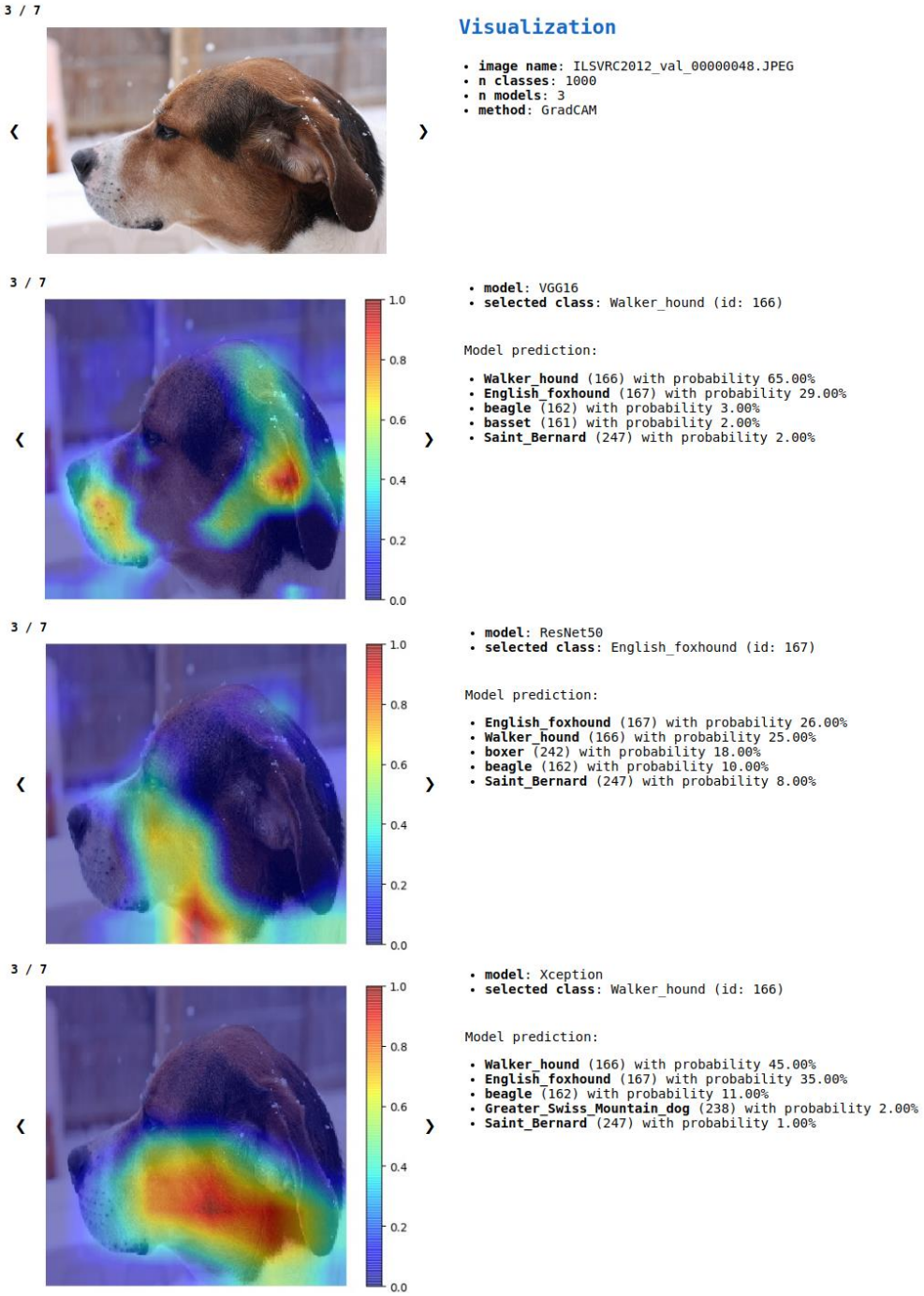

Figura 52 – Rappresentazioni GradCAM di una delle immagini identificata come caso interessante durante il test per i tre modelli in uso: in particolare si nota come i tre modelli producano classificazioni diverse tra loro, con una confidenza medio-bassa e anche se ResNet50 sembra aver dato la classificazione corretta, lo score associato risulta piuttosto basso. Utilizzando le rappresentazioni GradCAM si può verificare in effetti che questo comportamento può essere dovuto al fatto che nessuno dei tre modelli sembra focalizzarsi su una ben precisa caratteristica dell'immagine che possa far propendere la scelta verso una razza di cane piuttosto che un'altra con più sicurezza, e questo può essere dovuto al fatto che i modelli fanno fatica a riconoscere le caratteristiche più appropriate, ma anche che l'immagine potrebbe non contenere sufficienti informazioni (ad esempio in questo caso risulta un primo piano piuttosto ravvicinato).

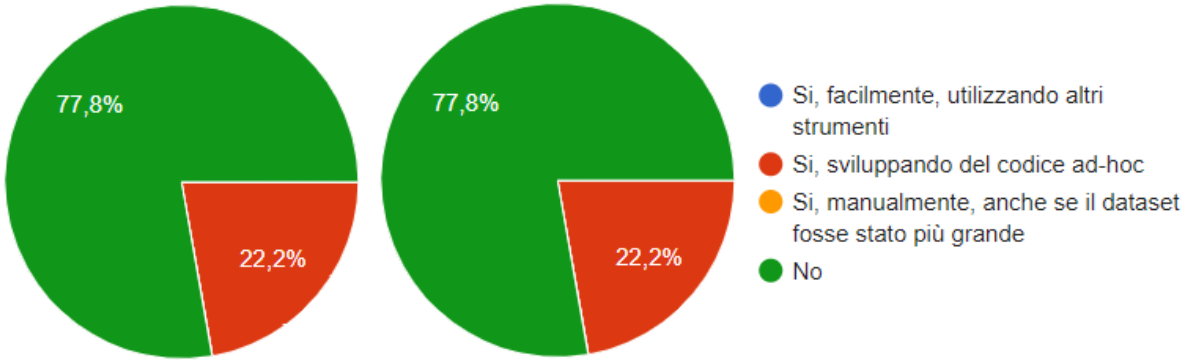

Figura 53 – Diagrammi a torta rappresentativi delle risposte date dai partecipanti del test relativamente al fatto di ritenersi in grado di giungere ai risultati richiesti durante il task 1 (a sinistra) e durante il task 2 (a destra) senza l'utilizzo di iNNvestigate-GUI.

5.2.2 Secondo task

Per quanto riguarda il secondo task, riguardante un dataset di immagini tutte appartenenti alla stessa classe ImageNet "*207 – golden retriever*", per cominciare è stato chiesto ai partecipanti come consideravano le classificazioni effettuate dai modelli in input rispetto all'effettiva classe di appartenenza. Come si vede dalla figura 54, i risultati dei modelli sono stati considerati piuttosto soddisfacenti utilizzando per la maggior parte, ovvero il 77,8% dei partecipanti, l'istogramma offerto dalla funzionalità di "*Suggest input images*", che in effetti mostra che la distanza media degli score ottenuti dai modelli rispetto alla classe in input è vicina allo 0 per la maggior parte delle immagini del dataset. Il 22,2% ha invece utilizzato solo il grafico a dispersione generato contemporaneamente dal tool, ed in effetti l'andamento di tale grafico conferma la risposta data (vedi figura 55).

Dunque, è stato chiesto di concentrarsi sulle immagini per le quali i modelli hanno funzionato molto bene, ovvero quelle presenti nel bin più a sinistra dell'istogramma a disposizione (visibile in figura 55). I partecipanti sono stati in grado di individuare correttamente questi casi per la maggior parte utilizzando proprio l'istogramma, mentre l'11,1% ha usato la vista fornita dal grafico a dispersione, che si è rivelata dunque utile anche in questo caso. Per l'individuazione delle caratteristiche visive utilizzate dai modelli per la classificazione di tali immagini, in questo

caso la maggior parte dei partecipanti ha utilizzato Guided Backpropagation, come si vede in figura 56: comunque, valgono le stesse osservazioni fatte per quanto riguarda il task 1, riguardo alla maggiore capacità del metodo GradCAM di aiutare l'utente nell'individuazione delle caratteristiche dell'immagine utilizzate dai modelli per la classificazione. Questo si è verificato

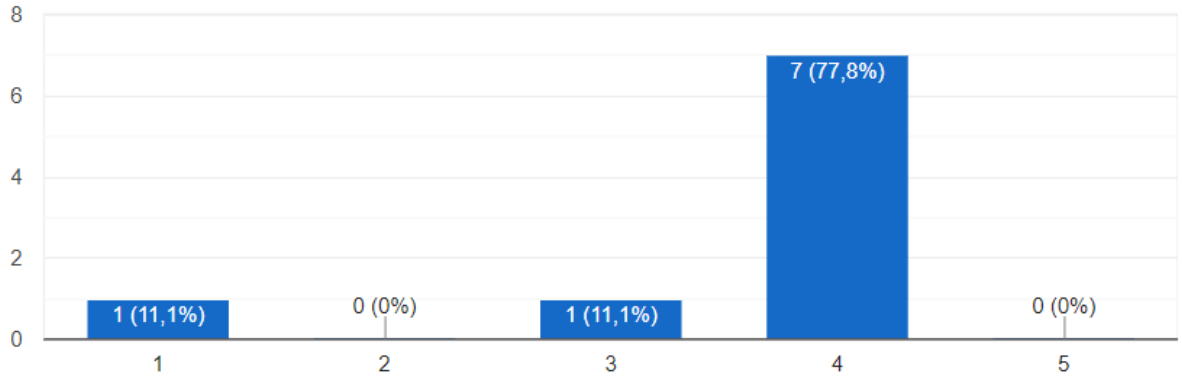

Figura 54 – Percentuale di risposte alla domanda "In generale, i modelli selezionati classificano in maniera soddisfacente le immagini presenti nel dataset in input?" relativa al task 2. I possibili valori per la risposta andavano da "No", assegnato al punteggio 1, a "Si", corrispondente al punteggio 5. Come si vede, grazie all'aiuto fornito dal tool, la maggior parte dei partecipanti ha risposto correttamente visualizzando l'andamento dei grafici (visibili in figura 55).

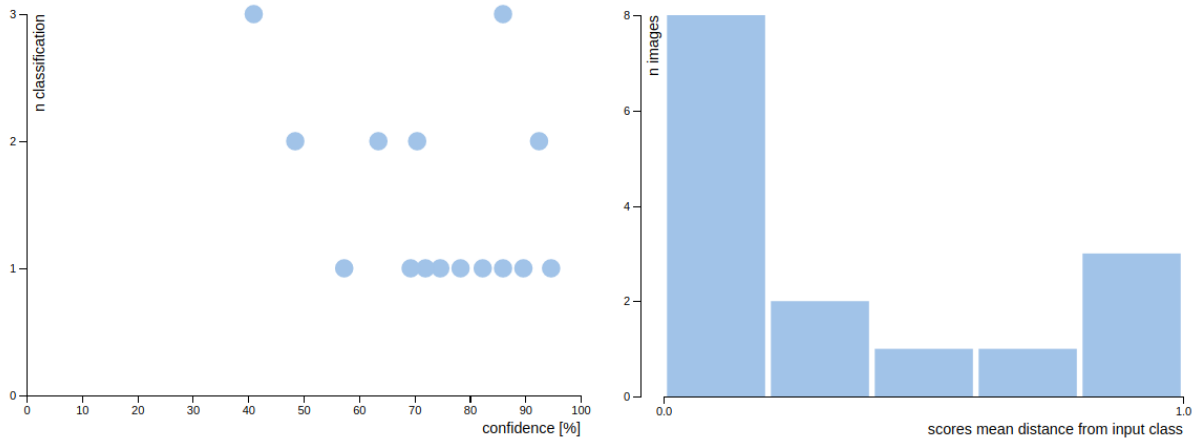

Figura 55 – Grafici generati da "Suggest input images" durante lo svolgimento del task 2. A sinistra è visualizzato il grafico a dispersione che mostra la relazione tra confidenza e coerenza delle classificazioni effettuate dai modelli, mentre a destra è visibile l'istogramma relativo alla distanza media degli score ottenuti dai modelli rispetto alla classe in input, "207 – golden retriever". L'andamento di entrambi i grafici risulta in accordo con quanto affermato dalla maggior parte dei partecipanti al test relativamente alla domanda riguardante la soddisfazione generale sui risultati prodotti dai modelli selezionati sul dataset in input (figura 54). L'istogramma rappresenta il modo migliore per individuare le immagini richieste dal task 2, utilizzando il primo bin a sinistra per individuare le immagini sulle quali i modelli funzionavano meglio, e l'ultimo bin a destra per quelle sulle quali i modelli funzionavano male. Le stesse immagini possono comunque essere individuate utilizzando il grafico a dispersione a sinistra, che mostra però una vista d'insieme e dunque non legata alla classe in input (come descritto nel paragrafo 4.2.2).

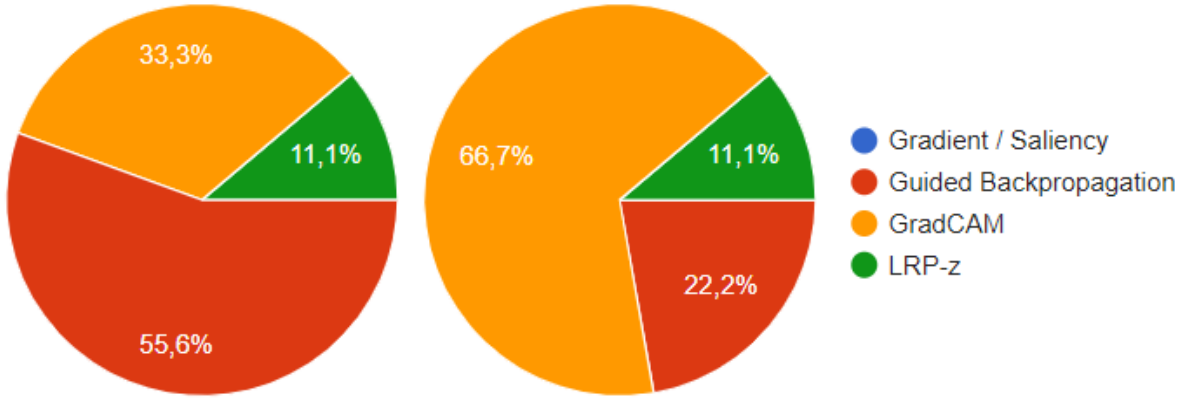

Figura 56 – Diagrammi a torta che mostrano le percentuali dei metodi di visualizzazione preferiti dai partecipanti, tra i quattro suggeriti, durante lo svolgimento del task 2 per l'individuazione delle caratteristiche visive delle immagini sulle quali i modelli hanno funzionato bene (a sinistra) e male (a destra). Il metodo preferito rimane complessivamente GradCAM, che restituisce rappresentazioni più precise per il task in questione, ma nel primo caso i partecipanti hanno utilizzato maggiormente Guided Backpropagation. Come per il task 1, anche durante il task 2 il metodo Gradient / Saliency si è confermato il piu inefficace.

in particolare nelle risposte alle domande specifiche per ogni modello, dove si chiedeva se le caratteristiche visive usate dal modello in questione potevano ritenersi non appropriate ai fini della classificazione. Le risposte di chi ha usato Guided Backpropagation risultano quasi sempre fortemente in disaccordo con l'affermazione precedente, mentre quelle di chi ha usato GradCAM risultano talvolta d'accordo, ed in effetti GradCAM offre una rappresentazione che permette di identificare più facilmente le aree dove il modello si è focalizzato maggiormente per la classificazione. Comunque, come si vede in figura 57, le risposte supportano quanto visto in precedenza nella domanda riguardante l'andamento generale delle classificazioni effettuate dai modelli sul dataset in input, che era stato considerato soddisfacente. Infatti, per i tre modelli in uso, i partecipanti si sono ritrovati complessivamente in disaccordo riguardo al ritenere non appropriate le caratteristiche visive utilizzate da ciascun modello per la classificazione, confermando il buon andamento generale dei modelli con il dataset proposto.

È stato poi chiesto ai partecipanti di concentrarsi sulle immagini per le quali i modelli hanno funzionato peggio, ed anche in questo caso la maggior parte ha individuato tali immagini correttamente utilizzando l'istogramma fornito da "*Suggest input images*", in particolare utilizzando stavolta l'ultimo bin a destra (figura 55). Il 22,2% dei partecipanti ha invece semplicemente utilizzato il grafico a dispersione generato contemporaneamente all'istogramma,

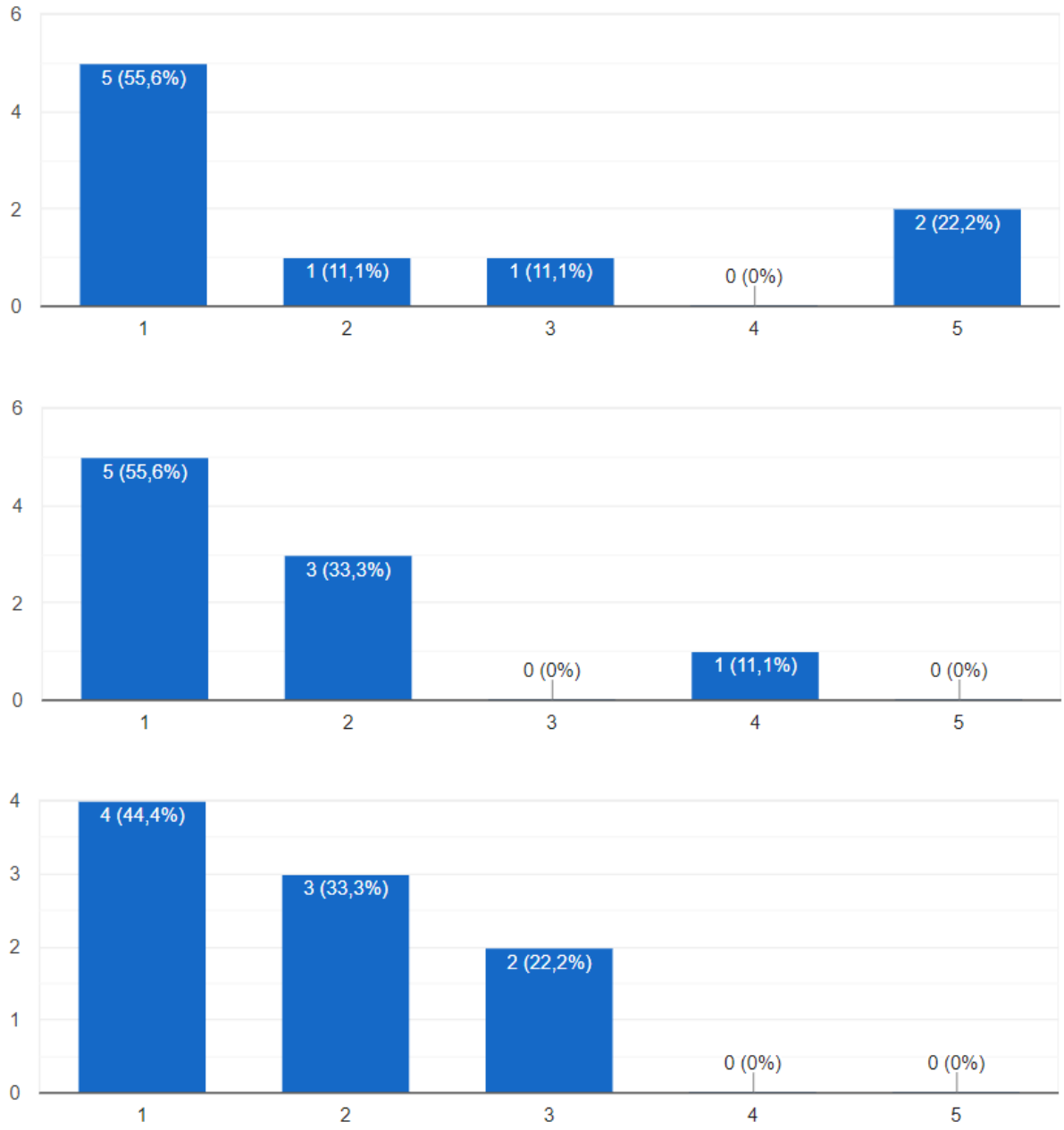

Figura 57 – Percentuali di risposte relative alle immagini sulle quali i modelli hanno funzionato bene durante il task 2, alla domanda "Ritieni che il modello abbia utilizzato caratteristiche visive che non considereresti appropriate per il riconoscimento della classe?", per i modelli VGG16 (primo grafico in alto), ResNet50 (al centro) e Xception (in basso). L'andamento complessivo supporta quanto si afferma relativamente alla figura 54, ovvero che i modelli risultano soddisfacenti nelle classificazioni effettuate, mostrando che la maggior parte dei partecipanti si è ritenuta in forte disaccordo (punteggio 1, corrispondente alla risposta "No") con quanto enunciato nella domanda, considerando in generale i modelli efficaci nell'individuare le giuste caratteristiche visive da utilizzare per una corretta classificazione.

confermandosi utile anche in questo scenario (figura 55). In questo caso, il metodo di visualizzazione più utilizzato per l'individuazione delle caratteristiche visive usate dai modelli per la classificazione è stato GradCAM, come si vede in figura 56, riconfermandosi ancora una volta il metodo più utile per questo tipo di analisi.

Infine, anche per il task 2, i partecipanti hanno per la maggior parte affermato di non ritenersi in grado di giungere ai risultati richiesti per rispondere alle domande del questionario senza l'utilizzo di un tool come iNNvestigate-GUI (figura 53).

5.2.3 System Usability Scale e domande finali

Come detto, oltre alle domande riguardanti i due task da svolgere, sono state integrate nel questionario le domande del SUS [50], in modo da ottenere una misura di usabilità di iNNvestigate-GUI. Il SUS è un sistema che prevede dieci domande alle quali rispondere con un punteggio da 1 a 5, rispettivamente se l'intervistato si trovi fortemente d'accordo o fortemente in disaccordo con l'affermazione contenuta nella domanda stessa. Al termine del questionario viene quindi calcolato un punteggio[25] sulla base delle risposte date, il punteggio SUS, che assegna un valore da 0 a 100 corrispondente alla misura dell'usabilità del sistema in questione.

Per ogni partecipante al test svolto è stato quindi calcolato il relativo punteggio SUS ottenuto, e il valore medio corrispondente risulta infine **73,34**. Questa è la misura complessiva di usabilità di iNNvestigate-GUI ricavata durante il test, che è considerabile "sopra la media" (in quanto il punteggio è maggiore di 68)[26] e in linea con le aspettative, soprattutto considerando che, come descritto in precedenza, i partecipanti erano tutti utenti non esperti.

Infine, sono state poste ai partecipanti alcune domande generali sul tool. In particolare, è stato chiesto loro se le funzionalità trovate potevano ritenersi utili per l'identificazione di eventuali problemi dei modelli o dei dataset in uso, e per tutti la risposta è stata positiva. Nello specifico

[25] Per maggiori informazioni sul calcolo del punteggio SUS si veda [50]
[26] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

poi, il 22,2% le ha ritenute utili affermando però che, in assenza di iNNvestigate-GUI, sarebbe stato in grado di utilizzare altri strumenti per lo stesso scopo. Nessuno dei partecipanti al test ha però affermato di essere a conoscenza di altri strumenti simili o che ritenesse migliori di iNNvestigate-GUI. Infine, il 77,8% ha affermato che uno strumento simile lo aiuterebbe molto nel suo lavoro/studio, mentre i restanti hanno fermato che lo ritengono utile ma non stravolgente per il loro workflow abituale. Complessivamente, nessuno dei partecipanti ha ritenuto lo strumento inutile né troppo difficile da usare. L'ultima domanda richiedeva di inserire eventuali commenti o migliorie da apportare al tool: tutti i suggerimenti sono stati di natura stilistica o riguardanti ulteriori semplificazioni al workflow proposto, senza segnalare problemi o mancanze particolari nelle funzionalità offerte.

6 Conclusioni

In questo lavoro è stato analizzato nel dettaglio lo stato dell'arte delle tecniche di visualizzazione delle Deep Neural Network, concentrandosi in particolare sulle reti neurali convoluzionali, e dei principali strumenti e librerie software che le implementano. Nello specifico, si è visto come la visualizzazione di una rete neurale non corrisponde ad un singolo concetto, ma può riguardare caratteristiche differenti ed essere utile per obiettivi diversi a seconda dell'oggetto della visualizzazione. In ogni caso però, si è discusso di come la visualizzazione rappresenti un grande aiuto durante lo sviluppo o l'utilizzo di una rete neurale, per facilitarne la comprensione e dunque riuscire a migliorarla o ad usarla più efficacemente.

Sono state quindi approfondite nel dettaglio le principali tecniche di visualizzazione delle attivazioni, uno degli ambiti di maggior interesse per la ricerca, suddividendole in base all'obiettivo che permettono di raggiungere, ovvero a cosa corrisponde l'output della tecnica stessa. Per questo tipo di tecniche si è visto come, tra i software analizzati, pochi risultino gli strumenti che offrono un'implementazione standard e unificata, rendendo dunque difficile l'integrazione della visualizzazione nel workflow degli sviluppatori o utilizzatori delle reti, a meno di scrivere codice ad-hoc caso per caso. Nonostante la sua grande utilità, per questi motivi la visualizzazione delle reti viene spesso completamente evitata, soprattutto dagli utenti meno esperti, più limitati nello sviluppo di librerie apposite per risolvere il problema, che allo stesso tempo risulterebbero i più indicati per approfittarne, proprio per via della loro scarsa esperienza. Tra i pochi tool individuati che offrono una soluzione al problema, si è infine identificata come scelta migliore iNNvestigate, uno strumento a riga di comando che rende disponibile un vasto set di tecniche di visualizzazione delle attivazioni in un'implementazione unificata che risulta così molto semplice da utilizzare.

6 Conclusioni

Sulla base di iNNvestigate, per questa tesi è stato sviluppato iNNvestigate-GUI, uno strumento che offre un'interfaccia grafica moderna ed intuitiva in ambiente web per tutte le funzionalità offerte da iNNvestigate, estendendone le potenzialità ed affiancandolo a nuove funzioni. Si è visto come, con iNNvestigate-GUI, sia possibile sfruttare le tecniche di visualizzazione delle attivazioni proposte da iNNvestigate, ed altre ancora, per ottenere rappresentazioni di modelli di CNN per la classificazione di immagini, predefiniti o custom, ad uno specifico layer della rete, neurone di un layer o relative ad una certa classe tra quelle riconosciute dai modelli. Partendo dal presupposto che i dataset utilizzati per le operazioni da svolgere con le CNN contengono un gran numero di immagini, visualizzare le rappresentazioni ottenute dalle varie tecniche proposte, a partire da ciascuna di queste immagini, ne vanificherebbe l'utilità visto il gran numero di risultati che si dovrebbero analizzare. Per questo motivo, iNNvestigate-GUI propone una seconda importante macro-funzionalità: a partire da un dataset di immagini ed i risultati ottenuti su di esse da uno o più modelli in input, il tool è in grado di proporre viste con grafici interattivi che forniscono strumenti per l'identificazione dei casi più interessanti, come ad esempio le immagini che non stimolano un particolare layer o neurone, ovvero che producono un'attivazione bassa, analizzabili per comprendere quali feature non vengono riconosciute dal layer o neurone in esame, oppure le immagini per le quali i modelli producono classificazioni differenti l'uno dall'altro, analizzabili per identificare quali caratteristiche dell'immagine vengono utilizzate per la classificazione dai diversi modelli, ed eventualmente per riconoscere problemi nei modelli o nel dataset.

Con iNNvestigate-GUI si è proposto uno strumento in grado di supportare completamente lo sviluppatore o utilizzatore di reti neurali, offrendo un workflow da integrare nel proprio processo, a partire dal suggerimento delle immagini da utilizzare per approfondire l'analisi di una rete, fino all'analisi stessa attraverso le varie tecniche di visualizzazione fornite. Per risultare il più trasparente possibile agli utenti, il tool è stato sviluppato puntando a semplicità e velocità d'uso, riducendo al minimo le conoscenze aggiuntive da apprendere per poter cominciare ad utilizzarlo. Proprio per questo motivo, pur ritenendo lo strumento molto utile per qualsiasi tipo di utenza, il target principale è rappresentato sicuramente dagli utenti meno esperti, come confermato dai risultati ottenuti durante il test svolto per la valutazione di efficacia ed usabilità del

tool. Il punteggio SUS ottenuto e i risultati riguardanti i due task svolti dai partecipanti, chiamati a rappresentare appunto un'utenza non esperta, tramite l'utilizzo delle funzionalità di iNNvestigate-GUI, sono in linea con le aspettative auspicabili sulla base degli obiettivi seguiti per lo sviluppo del tool.

6.1 Sviluppi futuri

Il tool è stato sviluppato sulla base di iNNvestigate, richiedendo come requisito l'utilizzo di Keras come framework per il Deep Learning. Sarebbe interessante integrare uno strumento di conversione per permettere l'utilizzo di iNNvestigate-GUI anche con modelli sviluppati in altri framework, come ad esempio Caffe. Inoltre, lo sviluppo si è incentrato sulle CNN per la classificazione di immagini, ma le funzionalità messe a disposizione dal tool sarebbero sicuramente molto utili se adattate anche per altri tipi di reti, come ad esempio quelle per il riconoscimento di oggetti (Object Detection)[27] o per reti GANs (Generative Adversarials Networks)[28].

Alcuni aspetti del tool possono essere ulteriormente semplificati, ad esempio potrebbe essere introdotta la possibilità di ottenere direttamente le rappresentazioni prodotte dalle tecniche di visualizzazione sulle immagini suggerite con la funzionalità "*Suggest input images*", senza bisogno di dover procedere manualmente alla funzionalità "*Visualize*". Un altro esempio potrebbe essere l'introduzione di ulteriori accorgimenti stilistici come ad esempio la possibilità di rinominare e di spostare a piacere le tab che si aprono al click dei bottoni "*Visualize*" o "*Suggest input images*". Per quanto riguarda l'utilizzo del tool con modelli custom invece, potrebbe essere interessante aggiungere ulteriori modalità di pre-processing delle immagini in input. Inoltre, potrebbe essere molto utile integrare uno zoom nelle varie viste disponibili, o la possibilità di ricercare le immagini all'interno dei grafici proposti per l'analisi del dataset, in modo da poter riprendere facilmente un'analisi interrotta, o ancora una funzionalità di export dei grafici

[27] https://en.wikipedia.org/wiki/Object_detection
[28] https://en.wikipedia.org/wiki/Generative_adversarial_network

e delle viste prodotte dal tool. Sarebbe poi interessante trasformare il tool in uno strumento multiutente e persistente, dove ad esempio l'utente possa salvare determinate viste ottenute, senza bisogno di doverle rigenerare all'occorrenza, o che permetta la creazione di una coda di comandi, ad esempio la predisposizione di una serie di "*Visualize*" o "*Suggest input images*" che il tool possa eseguire uno dopo l'altro in autonomia, ad esempio inviando notifiche quando ogni vista è stata generata.

Infine, l'integrazione di ulteriori tecniche di visualizzazione è sicuramente un aspetto che rappresenta una possibilità di sviluppi futuri interessanti in ogni momento, soprattutto con l'introduzione di metodi di nuova generazione.

Riferimenti

[1] https://www.wikipedia.org/

[2] http://wiki.fast.ai/index.php/Main_Page

[3] http://neuralnetworksanddeeplearning.com/index.html

[4] https://www.kaggle.com/learn/deep-learning

[5] http://ruder.io/optimizing-gradient-descent/index.html#gradientdescentoptimizational-gorithms

[6] https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

[7] http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

[8] https://arxiv.org/pdf/1502.03167.pdf

[9] http://neuralnetworksanddeeplearning.com/chap6.html

[10] F. Hohman, M. Kahng, R. Pienta, D. Horng Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," arXiv:1801.06889v3, 2018.

[11] D. Smilkov, S. Carter, D. Sculley, F. B. Viegas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," in ICML Workshop on Vis for Deep Learning, 2016.

[12] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in ISVC, 2015, pp. 867–877.

[13] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," JMLR, vol. 9, no. Nov, 2008.

[14] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea, “Visualizing the hidden activity of artificial neural networks,” IEEE TVCG, vol. 23, no. 1, 2017.

[15] P. E. Rauber, A. X. Falcão, and A. C. Telea, “Visualizing timedependent data using dynamic t-sne,” EuroVis, vol. 2, no. 5, 2016.

[16] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in ECCV. Springer, 2014.

[17] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” arXiv:1312.6034, 2013.

[18] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, “SmoothGrad: removing noise by adding noise,” in ICML Workshop on Vis for Deep Learning, 2017.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, “Striving for Simplicity: The All Convolutional Net,” arXiv:1412.6806v3, 2015.

[20] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in CVPR, 2016.

[21] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, “Grad-CAM: Why did you say that? visual explanations from deep networks via gradient-based localization,” arXiv:1610.02391, 2016.

[22] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation,” PloS one 10 (7) (2015) e0130140.

[23] http://heatmapping.org/

[24] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, “Visualizing deep neural network decisions: Prediction difference analysis,” arXiv:1702.04595, 2017.

[25] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, “Visualbackprop: visualizing cnns for autonomous driving,” arXiv:1611.05418, 2016.

[26] H. Li, K. Mueller, and X. Chen, “Beyond saliency: understanding convolutional neural networks from saliency prediction on layerwise relevance propagation,” arXiv:1712.08268, 2017.

[27] A. Dosovitskiy and T. Brox, “Inverting visual representations with convolutional networks,” in CVPR, 2016.

[28] A. Mahendran and A. Vedaldi, “Understanding deep image representations by inverting them,” in CVPR, 2015.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., “TensorFlow: Large-scale machine learning on heterogeneous distributed systems,” arXiv:1603.04467, 2016.

[30] M. Kahng, P. Andrews, A. Kalro, and D. H. Chau, “ActiVis: Visual exploration of industry-scale deep neural network models,” IEEE TVCG, vol. 24, no. 1, 2018.

[31] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” in ICML Deep Learning Workshop, 2015.

[32] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, “Towards better analysis of deep convolutional neural networks,” IEEE TVCG, vol. 23, no. 1, 2017.

[33] S. Chung, C. Park, S. Suh, K. Kang, J. Choo, and B. C. Kwon, “ReVACNN: Steering convolutional neural network via realtime visual analytics,” in NIPS Workshop on Future of Interactive Learning Machines, 2016.

[34] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, “Embedding Projector: Interactive visualization and interpretation of embeddings,” in NIPS Workshop on Interpretable ML in Complex Systems, 2016.

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[36] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "DeepEyes: Progressive visual analytics for designing deep neural networks," IEEE TVCG, vol. 24, no. 1, 2018.

[37] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," Pattern Recognition, https://doi.org/10.1016/j.patcog.2016.11.008, vol. 65, 2017.

[38] A. Shrikumar, P. Greenside, A. Kundaje, "Learning Important Features Through Propagating Activation Differences," Proceedings of the 34th International Conference on Machine Learning, PMLR 70:3145-3153, 2017.

[39] M. Sundararajan, A.Taly, Q. Yan, "Axiomatic Attribution for Deep Networks," arXiv:1703.01365v2, 2017.

[40] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, S. Dähne, "Learning how to explain neural networks: PatternNet and PatternAttribution," arXiv:1705.05598v2, 2017.

[41] M. T. Ribeiro, S. Singh, C. Guestrin, ""Why Should I Trust You?" Explaining the Predictions of Any Classifier," arXiv:1602.04938v3, 2016.

[42] J. Castro, D. Gómez, J. Tejada, "Polynomial calculation of the Shapley value based on sampling," Computers & Operations Research, https://doi.org/10.1016/j.cor.2008.04.004, vol. 36, issue 5, 2009.

[43] M. Ancona, C. Öztireli, M. Gross, "Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Values Approximation," arXiv:1903.10992v4, 2019.

[44] L. S. Shapley, "A Value for n-Person Games," In: H. W. Kuhn and A. W. Tucker, Eds., Contributions to the Theory of Games II, Annals of Mathematics Studies, Princeton University Press, Princeton, vol. 28, 1953.

[45] M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, P.-J. Kindermans, “iNNvestigate neural networks!,” arXiv:1808.04260v1, 2018.

[46] https://github.com/albermax/innvestigate

[47] Z. Qin, F. Yu, C. Liu, X. Chen, “How convolutional neural network see the world - A survey of convolutional neural network visualization methods,” arXiv:1804.11191v2, 2018.

[48] https://www.quora.com/Why-is-ReLU-the-most-common-activation-function-used-in-neural-networks

[49] M. Lin, Q. Chen, S. Yan, “Network In Network,” arXiv:1312.4400v3, 2014.

[50] J. Brooke, “SUS - A quick and dirty usability scale,” Usability evaluation in industry (P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.)), 1996

Appendice 1

Questionario sul test di validazione per iNNvestigate-GUI

Task 1

Domande sullo svolgimento del task 1: valutare quali caratteristiche dell'immagine (ad esempio quali parti dell'oggetto o della scena) vengono utilizzate per la classificazione da parte di reti differenti.

*Campo obbligatorio

1. Indica il nome di almeno un'immagine per la quale i modelli predicono i risultati con sicurezza e risultano d'accordo sulla classe assegnata.

2. Per l'immagine/i indicata/e nella risposta precedente, le caratteristiche visive utilizzate dai modelli per la classificazione sono state le stesse per tutti i modelli? *

   Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

3. Quale metodo di visualizzazione hai ritenuto più utile per rispondere alla domanda precedente? *

   Contrassegna solo un ovale.

   - ◯ Gradient / Saliency
   - ◯ Guided Backpropagation
   - ◯ GradCAM
   - ◯ LRP-z

4. Indica il nome di almeno un'immagine per la quale i modelli predicono i risultati con insicurezza e non risultano d'accordo con le classi assegnate. *

   _____

   _____

   _____

   _____

   _____

5. Per l'immagine/i indicata/e nella risposta precedente, le caratteristiche visive utilizzate dai modelli per la classificazione sono state le stesse per tutti i modelli? *

   Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

6. Quale metodo di visualizzazione hai ritenuto più utile per rispondere alla domanda precedente? *

   Contrassegna solo un ovale.

   - ◯ Gradient / Saliency
   - ◯ Guided Backpropagation
   - ◯ GradCAM
   - ◯ LRP-z

7. Indica il nome di almeno un'immagine che ritieni possa rappresentare un altro caso interessante e spiega il perché. *

_____

_____

_____

_____

_____

8. Come hai effettuato la scelta dell'immagine/i da indicare come risposta per la domanda precedente? *

Contrassegna solo un ovale.

- ◯ Ho utilizzato il grafico proposto dalla funzionalità "Suggest input images" inclusa nel tool
- ◯ Ho analizzato le immagini del dataset una ad una e ho scelto secondo la mia esperienza
- ◯ Ho analizzato le immagini del dataset una ad una e ho scelto utilizzando le rappresentazioni fornite dai metodi di visualizzazione integrati nel tool

9. Per l'immagine/i identificata/e come caso interessante, le caratteristiche visive utilizzate dai modelli per la classificazione sono state le stesse per tutti i modelli? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

10. Quale metodo di visualizzazione hai ritenuto più utile per rispondere alla domanda precedente? *

Contrassegna solo un ovale.

- ◯ Gradient / Saliency
- ◯ Guided Backpropagation
- ◯ GradCAM
- ◯ LRP-z

11. Ritieni che saresti stato in grado di rilevare questi casi, ed eventualmente altri interessanti, senza utilizzare gli strumenti messi a disposizione dal tool? *

Contrassegna solo un ovale.

- ◯ Si, facilmente, utilizzando altri strumenti
- ◯ Si, sviluppando del codice ad-hoc
- ◯ Si, manualmente, anche se il dataset fosse stato più grande
- ◯ No

12. Sulla base dei risultati ottenuti, come valuteresti in generale il comportamento dei modelli rispetto al dataset fornito? *

Seleziona tutte le voci applicabili.

- ☐ I modelli sembrano comportarsi in maniera casuale
- ☐ Non saprei
- ☐ Il dataset contiene immagini adeguate per l'utilizzo con i modelli scelti
- ☐ Il dataset sembra non essere adatto ai modelli scelti
- ☐ I modelli risultano coerenti nella classificazione nella maggior parte dei casi
- ☐ I modelli risultano piuttosto confidenti sulle classificazioni effettuate, ma non sono coerenti tra loro
- ☐ I modelli risultano piuttosto confidenti sulle classificazioni effettuate nella maggior parte dei casi

Task 2

Domande sullo svolgimento del task 2: verificare se reti neurali differenti si basino su caratteristiche dell'immagine diverse per riconoscere scene o oggetti di una determinata classe.

13. In generale, i modelli selezionati classificano in maniera soddisfacente le immagini presenti nel dataset in input? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

14. Cosa hai utilizzato per rispondere alla precedente domanda? *

Contrassegna solo un ovale.

- ◯ Ho analizzato le immagini una ad una utilizzando le rappresentazioni e le informazioni ricavate utilizzando i metodi di visualizzazione integrati nel tool
- ◯ Ho utilizzato il grafico proposto dalla funzionalità "Suggest input images" inclusa nel tool, senza fornire ulteriori input
- ◯ Ho utilizzato l'istogramma proposto dalla funzionalità "Suggest input images", inserendo come input la classe di appartenenza delle immagini del dataset ("207 - golden retriever")

15. Per le immagini sulle quali i modelli hanno funzionato molto bene, le caratteristiche visive utilizzate per la classificazione sono state le stesse per tutti i modelli? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

16. Per le stesse immagini della domanda precedente, ritieni che il modello VGG16 abbia utilizzato caratteristiche visive che non considereresti appropriate per il riconoscimento della classe? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

17. Per le stesse immagini della domanda precedente, ritieni che il modello ResNet50 abbia utilizzato caratteristiche visive che non considereresti appropriate per il riconoscimento della classe? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

18. Per le stesse immagini della domanda precedente, ritieni che il modello Xception abbia utilizzato caratteristiche visive che non considereresti appropriate per il riconoscimento della classe? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

19. Quale metodo di visualizzazione hai ritenuto più utile per rispondere alle domande precedenti inerenti le immagini sulle quali i modelli hanno funzionato molto bene? *

Contrassegna solo un ovale.

- ◯ Gradient / Saliency
- ◯ Guided Backpropagation
- ◯ GradCAM
- ◯ LRP-z

20. Come hai scelto le immagini sulle quali effettuare la visualizzazione? *

Contrassegna solo un ovale.

- ◯ Ho analizzato le immagini una ad una utilizzando le rappresentazioni e le informazioni ricavate utilizzando i metodi di visualizzazione integrati nel tool
- ◯ Ho utilizzato il grafico proposto dalla funzionalità "Suggest input images" inclusa nel tool, senza fornire ulteriori input
- ◯ Ho utilizzato l'istogramma proposto dalla funzionalità "Suggest input images", inserendo come input la classe di appartenenza delle immagini del dataset ("207 - golden retriever")

21. Per le immagini sulle quali i modelli hanno funzionato male, le caratteristiche visive utilizzate per la classificazione sono state le stesse per tutti i modelli? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No	◯	◯	◯	◯	◯	Si

22. Quale metodo di visualizzazione hai ritenuto più utile per rispondere alla domanda precedente? *

Contrassegna solo un ovale.

- ◯ Gradient / Saliency
- ◯ Guided Backpropagation
- ◯ GradCAM
- ◯ LRP-z

23. Come hai scelto le immagini sulle quali effettuare la visualizzazione? *

Contrassegna solo un ovale.

- ◯ Ho analizzato le immagini una ad una utilizzando le rappresentazioni e le informazioni ricavate utilizzando i metodi di visualizzazione integrati nel tool
- ◯ Ho utilizzato il grafico proposto dalla funzionalità "Suggest input images" inclusa nel tool, senza fornire ulteriori input
- ◯ Ho utilizzato l'istogramma proposto dalla funzionalità "Suggest input images", inserendo come input la classe di appartenenza delle immagini del dataset ("207 - golden retriever)

24. Ritieni che saresti stato in grado di scegliere le immagini da analizzare per rispondere alle precedenti domande, senza utilizzare gli strumenti messi a disposizione dal tool? *

Contrassegna solo un ovale.

- ◯ Si, facilmente, utilizzando altri strumenti
- ◯ Si, sviluppando del codice ad-hoc
- ◯ Si, manualmente, anche se il dataset fosse stato più grande
- ◯ No

System Usability Scale

Questionario SUS per iNNvestigate-GUI.

25. Penso che mi piacerebbe utilizzare questo strumento frequentemente *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

26. Ho trovato lo strumento inutilmente complesso *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

27. Ho trovato lo strumento semplice da usare *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

28. Penso che avrei bisogno del supporto di un tecnico per essere in grado di utilizzare lo strumento *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

29. Ho trovato le varie funzionalità dello strumento ben integrate *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

30. Ho trovato troppe incoerenze tra le varie funzionalità dello strumento *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

31. Penso che la maggior parte delle persone possa imparare ad utilizzare lo strumento molto rapidamente *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

32. Ho trovato lo strumento molto difficile da utilizzare *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

33. Mi sono sentito a mio agio nell'utilizzare lo strumento *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

34. Ho avuto bisogno di imparare molte cose prima di riuscire ad utilizzare al meglio lo strumento *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fortemente in disaccordo	◯	◯	◯	◯	◯	Fortemente d'accordo

Domande finali

35. Ritieni che le funzionalità messe a disposizione dal tool possano essere utili nel processo di identificazione di eventuali debolezze nei modelli o nei dataset utilizzati? *

Contrassegna solo un ovale.

- ◯ Si, molto utili
- ◯ Si, ma in assenza di questo tool userei altri strumenti
- ◯ No, altri strumenti includono funzionalità più interessanti
- ◯ No, è semplice effettuare tale processo manualmente
- ◯ No, tale processo è inutile

36. Indica i nomi di altri strumenti di tua conoscenza simili ad iNNvestigate-GUI o che mettono a disposizione funzionalità che ritieni più interessanti.

37. Utilizzeresti iNNvestigate-GUI come parte integrante del tuo lavoro/studio? *

Contrassegna solo un ovale.

- ◯ Si, mi aiuterebbe molto
- ◯ Si, ma non cambierebbe di molto il mio workflow
- ◯ No, preferisco utilizzare altri strumenti
- ◯ No, preferisco sviluppare del codice ad-hoc all'occorrenza
- ◯ No, non lo trovo utile
- ◯ No, è troppo difficile da usare

38. Indica eventuali modifiche che ritieni possano migliorare il tool o altri consigli e commenti.