UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN BIOINGEGNERIA

GENERAZIONE DI UN DATASET SINTETICO DI IMMAGINI DI SOGGETTI TATUATI PER APPLICAZIONI DI INTELLIGENZA ARTIFICIALE IN AMBITO FORENSE

Relatore: Prof.ssa Maria Francesca Uccheddu

Laureanda: Irene De Zotti
Matricola: 2039795

ANNO ACCADEMICO 2022 – 2023

Data di laurea: 14/12/2023

Desidero ringraziare tutti coloro
che mi hanno sostenuto lungo questo percorso.
Senza il vostro prezioso contributo e incoraggiamento,
questo successo non sarebbe stato possibile.
Grazie di cuore per essere stati al mio fianco
in questo significativo capitolo della mia vita.

Abstract

Nella società contemporanea, i tatuaggi, diffusi come forma di espressione, assumono rilevanza anche nell'ambito della biometria, in quanto rappresentano dei segni distintivi per il riconoscimento degli individui. La creazione di un vasto dataset di tatuaggi, con varietà di disegni e posizioni corporee, potrebbe accelerare lo sviluppo di sistemi di intelligenza artificiale per l'identificazione individuale. Le attuali reti neurali, sebbene efficaci in altri contesti, falliscono nel riconoscere tatuaggi a causa della loro tridimensionalità sul corpo umano. È essenziale, pertanto, addestrare tali reti su nuovi dataset per migliorare la precisione e l'affidabilità del riconoscimento dei tatuaggi. Tuttavia, la disponibilità di una banca dati costituita da immagini reali è limitata, spesso a causa di questioni legate alla privacy e all'etica.

Questo studio si propone di affrontare questa sfida attraverso lo sviluppo di un dataset sintetico di immagini contenenti soggetti tatuati, coprendo una vasta gamma di varianti con l'obiettivo di aumentare la diversità e la complessità del dataset stesso.

Per realizzare il dataset sintetico è stato sviluppato un *tool* nell'ambiente di realtà virtuale Unity. Il sistema proposto prevede la proiezione automatica di immagini su modelli sintetici 3D di corpi umani, acquisendo dati da varie angolazioni e in contesti diversi, inclusi cambiamenti di illuminazione e sfondo. Il sistema è stato testato in un ipotetico contesto forense per identificare membri di una gang. Questo limita la varietà dei tatuaggi considerati, poiché, nonostante la diversità, i membri della stessa mafia tendono ad avere tatuaggi simili tra loro.

La validazione dell'efficacia del dataset è stata condotta mediante l'impiego di una rete neurale per il riconoscimento dei tatuaggi. Diverse architetture sono state testate, identificando quella che ha dimostrato prestazioni superiori. L'analisi ha coinvolto sia immagini sintetiche che reali, garantendo così una completa comprensione delle prestazioni. Nel caso della rete InceptionV3, l'accuratezza nel riconoscimento di immagini reali ha raggiunto valori dell'87.17%. Di conseguenza, è stata adottata come modello di base per implementare una rete Fast R-CNN, consentendo anche la localizzazione precisa dei tatuaggi nelle immagini. La valutazione della sovrapposizione tra tatuaggi identificati manualmente e quelli individuati dalla rete ha generato valori medi nei range di riferimento per gli indici IoU (0.5805) e DICE (0.6916). Questi risultati evidenziano che l'impiego di un dataset sintetico può rappresentare un approccio innovativo per l'addestramento delle reti neurali.

Abstract

In contemporary society, tattoos, which are widespread as a form of expression, also assume relevance in the field of biometrics, as they represent distinctive signs for the recognition of individuals. The creation of a vast dataset of tattoos, with a variety of designs and body positions, could accelerate the development of artificial intelligence systems for individual identification. Current neural networks, although effective in other contexts, fail to recognise tattoos due to their three-dimensionality on the human body. It is therefore essential to train such networks on new datasets to improve the accuracy and reliability of tattoo recognition. However, the availability of a database consisting of real images is limited, often due to privacy and ethical issues.

This study aims to address this challenge through the development of a synthetic dataset of images containing tattooed subjects, covering a wide range of variants with the aim of increasing the diversity and complexity of the dataset.

The proposed system involves the automatic projection of images onto 3D synthetic models of human bodies, acquiring data from various angles and in different contexts, including changes in lighting and background. The system was tested in a hypothetical forensic context to identify gang members. This limits the variety of tattoos considered, as, despite the diversity, members of the same mafia tend to have similar tattoos.

Validation of the effectiveness of the dataset was conducted using a neural network for tattoo recognition. Several architectures were tested, identifying the one that demonstrated superior performance. The analysis involved both synthetic and real images, thus ensuring a complete understanding of performance. In the case of the InceptionV3 network, the accuracy in the recognition of real images reached values of 87.17%. Consequently, it was adopted as a basic model to implement a Fast R-CNN network, also enabling the precise localisation of tattoos in images. The evaluation of the overlap between manually identified tattoos and those identified by the network generated mean values in the reference ranges for the IoU (0.5805) and DICE (0.6916) indices. These results show that the use of a synthetic dataset may represent an innovative approach for training neural networks.

Sommario

Introduzione

Negli ultimi anni, la biometria ha assunto un ruolo fondamentale, in modo particolare nel contesto della sicurezza e dell'identificazione personale. Questa tecnologia sfrutta le caratteristiche biologiche o comportamentali di un individuo, al fine di identificarlo in maniera univoca e affidabile.

Oltre ai tratti biometrici storicamente usati, come le impronte digitali ed il timbro vocale, stanno acquisendo sempre più importanza i tratti biometrici definiti *soft*, come ad esempio i tatuaggi e le cicatrici. I tatuaggi, essendo delle marcature permanenti sulla pelle, possono fornire informazioni uniche e potenzialmente preziose.

La biometria è utilizzata in svariati settori, tra cui la sicurezza informatica, la verifica dell'identità nei sistemi digitali e, in particolare, nel campo investigativo. Infatti, nell'ambito delle scienze forensi, l'identificazione accurata dei soggetti risulta essenziale per garantire sicurezza e giustizia. A questo scopo le caratteristiche uniche dei tatuaggi possono aiutare nell'identificazione di individui, facilitando le indagini criminali, la risoluzione di casi e la conferma dell'identità di vittime o sospettati.

Attualmente, i dataset di immagini contenenti tatuaggi derivano prevalentemente da collezioni delle forze dell'ordine e delle agenzie investigative. Questi, spesso limitati in dimensioni e varietà, riflettono le informazioni raccolte durante indagini criminali e operazioni di arresto. Le immagini dei tatuaggi vengono comunemente conservate come prove fisiche o fotografiche, permettendo agli investigatori di identificare o associare individui a casi specifici.

L'integrazione delle reti neurali nel contesto della biometria riveste una notevole rilevanza, in quanto esse consentono di riconoscere in modo accurato e affidabile determinate caratteristiche, contribuendo significativamente all'evoluzione delle tecnologie. Tuttavia, l'importanza di avere a disposizione dataset specifici diventa evidente considerando le limitazioni delle attuali reti neurali nel riconoscere tatuaggi, soprattutto a causa della loro tridimensionalità sul corpo umano. Sebbene esse siano efficaci in altri contesti, la complessità dei tatuaggi richiede un approccio dedicato. È cruciale addestrare queste reti utilizzando nuovi dataset appositamente creati al fine di affinare la loro capacità di riconoscere con precisione e affidabilità queste caratteristiche uniche.

L'obbiettivo principale di questo studio consiste nella creazione di un dataset, costituito da immagini di natura sintetica, destinato all'addestramento di reti neurali. Tali reti, una volta

sottoposte ad un processo di apprendimento mediante le suddette immagini sintetiche, saranno testate utilizzando immagini reali, al fine di valutare l'idoneità del dataset nell'applicazione pratica.

Nella prima fase, lo scopo è quello di sviluppare un dataset sintetico di immagini contenenti soggetti tatuati, attraverso l'implementazione di un *tool* in Unity, un ambiente di realtà virtuale. Questa soluzione permette di risolvere importanti questioni legate alla privacy, poiché il dataset verrà utilizzato per addestrare reti neurali e modelli di apprendimento automatico. Utilizzando soggetti umani 3D sintetici, il *tool* consente la generazione controllata di immagini di tatuaggi senza coinvolgere dati personali sensibili o informazioni identificative di individui reali. Questo approccio non solo tutela la privacy degli individui, ma offre anche la flessibilità necessaria per raccogliere un vasto insieme di dati rappresentativi, consentendo così un addestramento efficace dei modelli di riconoscimento dei tatuaggi.

Nella seconda fase, il dataset appositamente creato sarà impiegato nell'addestramento di reti neurali con l'obiettivo preciso di sviluppare un modello capace di identificare immagini contenenti soggetti tatuati. Inizialmente, la rete verrà allenata per riconoscere la presenza di tatuaggi nelle immagini, affinando la sua capacità di discriminare tra pelle tatuata e non tatuata. Poi, il focus si sposterà sulla localizzazione precisa del tatuaggio, mirando a ottenere una rete in grado di indicare con precisione la posizione del tatuaggio all’interno dell’immagine.

Capitolo 1

Tatuaggi come tratti biometrici

Nel vasto panorama delle scienze forensi, l'identificazione e la profilazione dei sospettati rappresentano da sempre sfide cruciali e fondamentali. La continua evoluzione delle tecnologie e delle metodologie ha portato a progressi significativi in questo settore, compresa l'adozione di nuove fonti di dati e l'innovazione nei metodi di analisi.

Tra i vari sviluppi, in questo contesto, l'utilizzo dei tatuaggi come traccia biometrica rappresenta un punto cruciale. Sebbene i tatuaggi abbiano una storia millenaria come forma d'arte e di espressione personale, la loro utilità nell'ambito delle scienze forensi è stata presa in considerazione in tempi recenti, quando si è capito che essi possono essere utilizzati non solo per l'identificazione dei colpevoli, ma anche delle vittime.

In questo capitolo verranno, inoltre, presentate le informazioni, ricavate dalla letteratura, per quanto riguarda i due elementi caratterizzati l'elaborato; in primis, verrà esposta l'importanza di avere un dataset di tatuaggi per poter raccogliere dati ed effettuare valutazioni in ambito forense. Successivamente verrà esposta una panoramica delle metodologie attualmente implementate, per quanto riguarda il recupero delle immagini contenenti tatuaggi.

LA BIOMETRIA

Il termine “riconoscimento biometrico”, detto anche più semplicemente “biometria”, fa riferimento al riconoscimento automatico degli individui, basandosi sulle loro caratteristiche anatomiche o comportamentali [1]. L’abilità di identificare gli individui in maniera pressoché istantanea trova, al giorno d’oggi, vari ambiti di applicazione: dall’impiego nelle scienze forensi, ai sistemi di accesso a smartphone e computer.

La biometria svolge un ruolo significativo in applicazioni inerenti alle forze dell’ordine sia come uno strumento investigativo per restringere una lista di sospettati, sia come evidenza scientifica in tribunale. Un ulteriore esempio, riportato dagli autori in [2], illustra come il riconoscimento biometrico possa essere impiegato nei Paesi in via di sviluppo per fornire un primo strumento di identificazione, qual ora venissero a mancare documenti d’identità ufficiali.

I sistemi di riconoscimento biometrico devono risultare accurati e devono permettere di identificare in maniera corretta l’individuo in questione: un loro fallimento può comportare ripercussioni molto gravi. Per questo motivo risulta importante creare dei sistemi che presentino buone performance anche quando le condizioni del campione non sono ottimali.

Saini et al. [3] riportano che un sistema biometrico, in generale, è composto da quattro moduli.

Il primo modulo permette di acquisire i dati biometrici grezzi mediante un sensore che permette di scannerizzare e leggere il tratto, come un sensore ottico nel caso delle impronte digitali. La qualità dei dati raccolti dipende strettamente dalla dispositivo usato.

Ad esso segue un modulo di valutazione delle qualità e di estrazione delle caratteristiche. Mediante degli algoritmi di miglioramento del segnale, la qualità dei dati raccolti viene aumentata. In seguito, i dati vengono processati e si estrae un set di caratteristiche, rappresentative del tratto biometrico stesso. Dopo di che, questo dato viene conservato.

A questo step, segue il modulo di *matching* e di decisione, in cui il tratto raccolto viene confrontato con un dataset già disponibile e si valutano dei punteggi di similarità, sulla base dei quali l’identità della persona viene validata o meno.

Infine, esiste il modulo rappresentante il database; in esso, vengono associati i dati del tratto biometrico insieme ad altre informazioni biografiche dell’individuo, come nome ed indirizzo.

Classificazione dei tratti biometrici

Come viene affermato dagli autori in [4], qualsiasi tratto fisico o caratteriale può essere candidato a tratto biometrico, purché esso soddisfi determinati criteri, quali: universalità, capacità di essere discriminativo tra la popolazione, invarianza, capacità di essere acquisito facilmente; inoltre, deve essere facilmente accettato come tale dalla popolazione e deve essere robusto in termini di resistenza ad attacchi contro i sistemi di riconoscimento stessi.

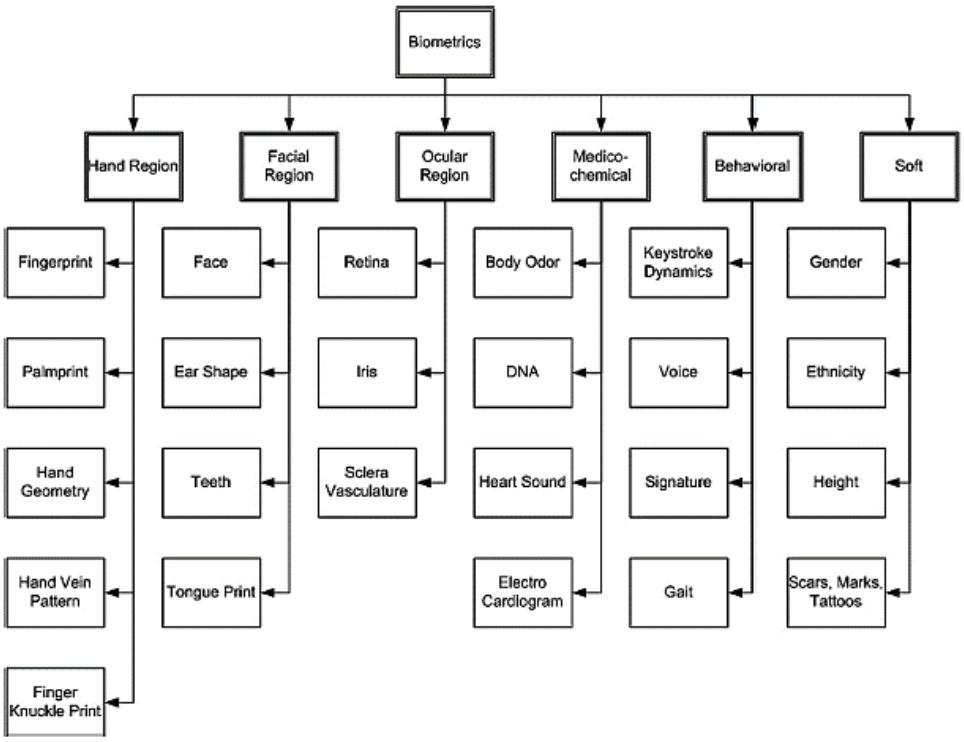

***Figura 1**. Classificazione dei vari tratti biometrici.* [5]

Una delle classificazioni possibili per quanto riguarda i tratti biometrici è quella che li suddivide in tratti di tipo *hard* o *soft*, a seconda della loro capacità discriminativa.

Le caratteristiche biometriche *hard* sono quelle che vengono normalmente utilizzate per l'identificazione degli individui facendo riferimento alle loro caratteristiche fisiche (DNA, orecchio, iride, retina, faccia, impronta digitale, palmo, vene, odore e corporatura) o comportamentali (andatura del passo, firma, voce, …).

Diverse sono invece le possibili definizioni per le caratteristiche biometriche di tipo *soft*. Una delle possibili viene riportata dagli autori in [1] come segue: "i tratti biometrici *soft* sono le caratteristiche che forniscono informazioni identificative su un individuo, ma mancano della distintività e permanenza tali da differenziare in modo sufficiente tra due individui."

Anche se i tratti biometrici di tipo *soft* non hanno la capacità di fornire una sufficiente informazione discriminativa per svolgere il compito di riconoscimento, essi possono essere utilizzati per una classificazione dei dati, che permette di velocizzare il tempo di riconoscimento e l'accuratezza, andando a restringere il campo di ricerca. Inoltre, questa tipologia può svolgere un ruolo molto importante nel momento in cui i tratti biometrici di tipo *hard*, risultano essere compromessi.

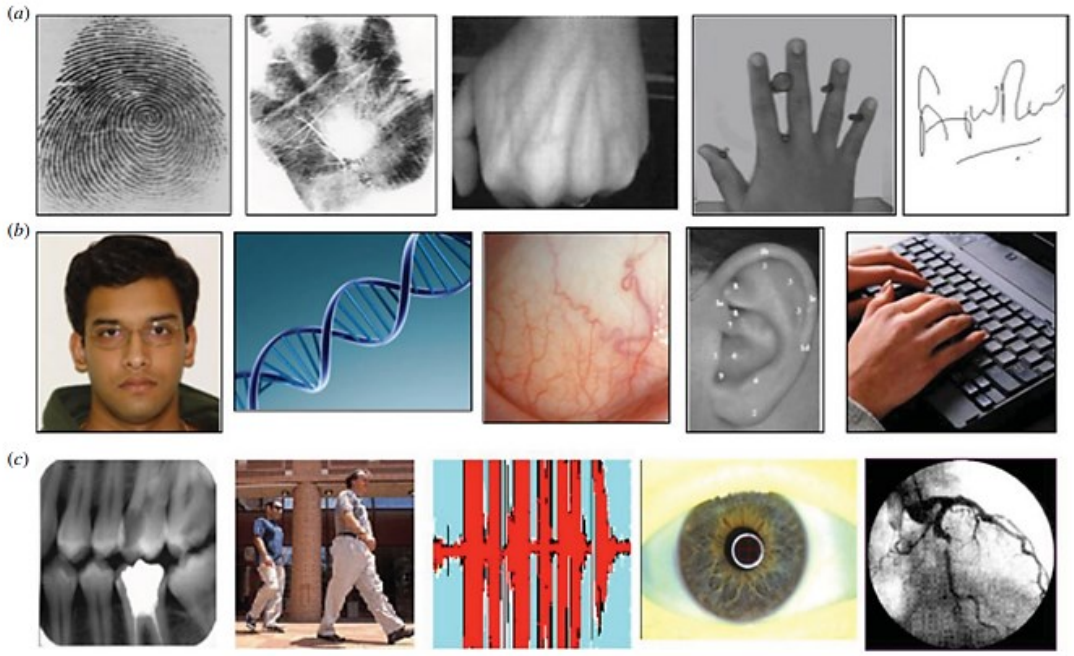

***Figura 2**. Esempi di tratti biometrici. (a) Impronte digitali, impronte del palmo, vascolarizzazione della mano, forma della mano e firma. (b) Viso, DNA, sclera (sul bulbo oculare), forma dell'orecchio e modelli di battitura (dinamica della battitura). (c) Denti (odontoiatria forense), andatura, voce o linguaggio, iride e retina. Alcuni di questi tratti, vale a dire le impronte digitali, le impronte del palmo, il viso, la voce, i denti, la forma dell'orecchio e il DNA, sono usati anche in medicina legale.* [2]

Confronto tra biometria e scienze forensi

Mentre la definizione di biometria è già stata analizzata in precedenza, con il termine "scienze forensi" ci si riferisce ai principi scientifici per poter analizzare le prove in una scena del crimine, allo scopo di ricostruire e descrivere gli eventi passati in un contesto legale. Questa disciplina è stata largamente influenzata dal principio di scambio di Locard, il quale stabilisce che ogni criminale lascia sul luogo del delitto una traccia e porta via con sé qualcosa [6]. Quindi, quando nell'ambito di un crimine una persona entra in contatto con un oggetto o con un'altra persona, si può sempre registrare uno scambio di prove fisiche.

Per questo motivo, un gran numero di tratti biometrici può essere utilizzato nelle investigazioni forensi. Uno dei principali obiettivi di quest'ultime è quello di associare una prova (es. impronta digitale) ad una sorgente (es. un individuo). Per raggiungere questo scopo sono possibili due diversi approcci [7].

Il primo approccio viene definito tradizionale. In questa tipologia di approccio sono possibili tre diversi risultati in seguito all'analisi della prova:

I. Individuazione: nessun altro individuo sulla terra può essere sorgente dell'impronta in oggetto;
II. Inconclusività: non è possibile stabilire con certezza se l'impronta appartiene ad un soggetto noto;
III. Esclusione: l'impronta sicuramente non è associato ad alcun individuo noto.

Oltre ad esso, è stato sviluppato anche un approccio contemporaneo. In questo caso ci si focalizza sulla forza dell'evidenza in favore di un paio di proposizioni:

H1: l'impronta digitale in oggetto ha origine da un donatore sospettato nel caso;
H2: l'impronta digitale proviene da un altro donatore.

Da questo punto di vista allora, sia la biometria che le scienze forensi cercano di collegare dati biologici ad un particolare individuo. Tra le due discipline però sono presenti un gran numero di differenze [2]; le scienze forensi sono invocate in seguito ad un evento per ricostruire l'accaduto, mentre il riconoscimento biometrico solitamente viene utilizzato prima dell'evento stesso (es. accedere ad un computer). Nell'investigazione forense non è possibile stabilire a priori il tipo di tratto che verrà utilizzato, mentre nei sistemi biometrici si conoscono in anticipo i tratti biologici utilizzati per riconoscere l'individuo. Le scienze forensi prevedono una raccolta e un'analisi manuale delle prove, mentre un riconoscimento biometrico è per definizione automatizzato. Le decisioni di riconoscimento nel caso delle scienze forensi non devono essere comunicate in tempo reale, mentre quelle della biometria richiedono questa caratteristica. Nelle scienze forensi un falso negativo può comportare l'esclusione di un soggetto della lista dei sospettati, mentre nel caso di un sistema biometrico può avere varie ripercussioni. Una decisione inconclusiva nelle scienze forensi significa che la prova del crimine non può essere associata con certezza ad un individuo, ma un sistema biometrico può acquisire campioni aggiuntivi di un tratto biometrico per stabilire o meno la corrispondenza. La qualità dei dati ottenuti nel caso delle scienze forensi è solitamente più bassa rispetto a quella della biometria; il risultato di un processo di investigazione forense deve essere comunicato ad una giuria, mentre quello di un riconoscimento biometrico è un valore numerico che viene utilizzato da un sistema automatico per dichiarare un eventuale riscontro.

Una tra le applicazioni della biometria nelle scienze forensi consiste nel riconoscimento facciale a partire da un disegno [2]. In questo caso, infatti, i disegni ed altri *sketch* compositi vengono utilizzati ormai di routine dalle forze dell'ordine per identificare sospetti coinvolti in crimini, qual ora non risulti disponibile un'immagine chiara del volto del sospettato. Dopo aver ricostruito un volto, le forze dell'ordine inviano il composito facciale a varie fonti con la speranza che qualcuno possa riconoscere l'individuo e fornire informazioni per portare ad un arresto.

Figura 3. *Esempi di compositi facciali utilizzati nei casi in cui il sospetto è stato arrestato con successo dopo che la polizia ha ricevuto una soffiata dal pubblico. Vengono mostrati esempi di compositi disegnati da un artista forense e le corrispondenti foto segnaletiche per (a) David Berkowitz (Son of Sam), (b) Timothy McVeigh (l'attentatore di Oklahoma City) e (c) Ted Kaczynski (l'Unabomber).* [2]

I TATUAGGI E IL LORO USO IN BIOMETRIA

Un tatuaggio è una deformazione artificiale permanente dei tessuti cutanei, ottenuta mediante segni indelebili prodotti per puntura dall'inserzione sotto la cute di sostanze coloranti senza alterare la superficie epidermica [8]. Se ciò accade involontariamente, per esempio, in seguito ad infortuni stradali, assume il nome di tatuaggio traumatico [9]. In ogni caso, l'origine dei tatuaggi più comuni è di tipo decorativo, ed essi sono legati alla moda del momento o ad un significato simbolico ad essi associato.

Uno dei più antichi tatuaggi rilevati risale al 3000 a.C. ed appartiene alla mummia di "Ötzi, l'uomo dei ghiacci", rinvenuta nel confine italo-austriaco nel 1991. Un esame radiologico delle ossa ha mostrato segni di osteocondrosi nelle aree in cui erano presenti i tatuaggi. Per questo motivo è stato ipotizzato che questi tatuaggi potrebbero essere stati correlati a trattamenti antidolorifici simili alla moderna agopuntura. Se così fosse, questa pratica potrebbe essere esistita almeno 2000 anni prima del suo primo utilizzo precedentemente noto, in Cina [10].

Criteri di scelta per la posizione di un tatuaggio

Nel vasto mondo dei tatuaggi, la scelta della parte del corpo su cui lasciare un'opera d'arte permanente è una decisione che richiede riflessione e cura. Ogni regione del corpo offre un'ampia tela per esprimere la personalità, i significati profondi e la creatività di chi decide di adornarla con inchiostro. Ogni parte del corpo ha il suo fascino e la sua unicità, e la posizione scelta può influenzare significativamente l'impatto visivo e simbolico del tatuaggio.

Nel mondo dei tatuaggi, le scelte delle parti del corpo su cui incidere l'arte della pelle possono variare significativamente tra uomini e donne. Ogni genere può avere preferenze distinte riguardo alle zone del corpo da tatuare, influenzate da motivazioni culturali, estetiche e personali.

Birngruber et al. [11] riportano una panoramica delle zone del corpo in cui, a seconda del sesso, sono presenti i tatuaggi. Secondo le loro indicazioni, le parti del corpo più tatuate, in ordine crescente, sono: genitali e glutei, testa e collo, gambe e piedi, tronco, avambracci e mani, spalle e braccia superiori. Inoltre, riportano come nella popolazione maschile le tre regioni del corpo che presentano una maggiore frequenza sono spalle e braccia superiori, avambracci e mani, tronco. Nei soggetti femminili, invece, sono stati riscontrati più tatuaggi su avambracci e mani, tronco, seguite dalle gambe e dai piedi.

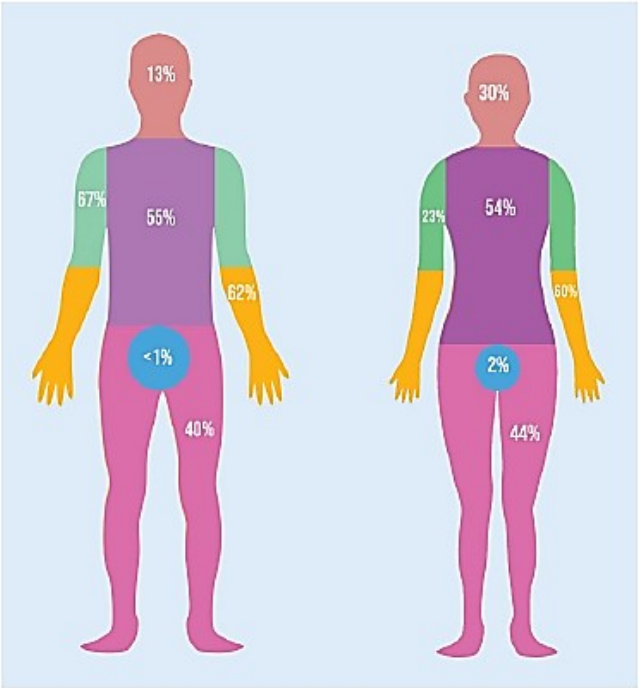

Figura 4. *Frequenza dei tatuaggi sulle diverse localizzazioni del corpo rispettivamente per soggetti maschili e femminili.*[11]

Ovviamente, queste differenze nelle preferenze non sono assolute e vanno sempre considerate le scelte individuali di ogni persona. Le preferenze possono variare notevolmente a seconda della personalità, dello stile di vita e dei significati personali che si vogliono esprimere attraverso il tatuaggio.

La scelta delle parti del corpo su cui farsi tatuare può dipendere da una serie di criteri, sia soggettivi che oggettivi, che variano da persona a persona. Alcuni dei criteri comuni [12] che spesso influenzano la decisione includono significato personale, estetica e design del tatuaggio stesso, visibilità del tatuaggio o possibilità di coprirlo facilmente, dolore associato alla parte del corpo tatuata, tendenze, mode o tradizione.

A seconda del sesso del soggetto, dunque, e tenendo in considerazione i vari criteri sopra citati, è possibile individuare 12 parti del corpo, che si presentano con frequenza diversa tra individui maschili e femminili [13].

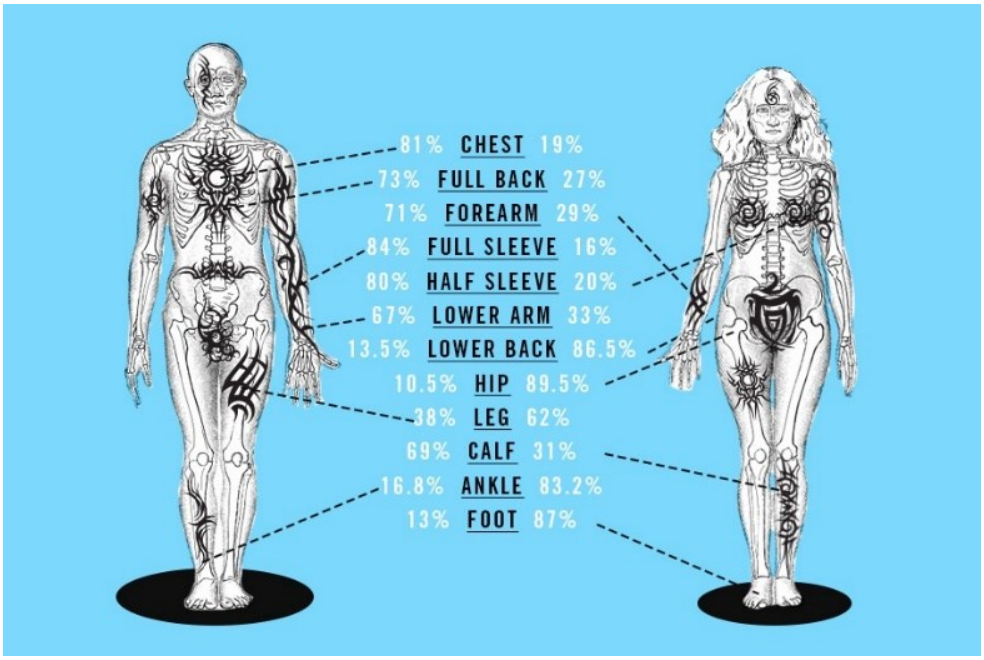

***Figura 4**. Rappresentazione delle varie parti del corpo tatuate, in soggetti maschili e femminili, con associata la frequenza di scelta.* [13]

Tatuaggi come tratti biometrici

Tra i vari tratti biometrici di tipo *soft*, i tatuaggi sono stati considerati una delle tipologie più importanti. Essi forniscono più informazioni discriminative per individuare un soggetto rispetto agli indicatori demografici classici come età, altezza, etnia e genere. Inoltre, molto spesso i tatuaggi vengono realizzati allo scopo di rappresentare specifici tratti della personalità oppure denotare l'appartenenza a determinati gruppi [1]. Per questo, l'analisi dei tatuaggi spesso porta a capire meglio il *background* di un individuo e la sua eventuale appartenenza ad un'organizzazione. Tuttavia, mentre per i tratti biometrici di tipo *hard* esiste una rappresentazione unica, per i tatuaggi non è possibile individuare una classificazione universale in quanto possono variare le forme, i colori, le textures, rendendo difficile il loro utilizzo.

***Figura 4**. Immagini del tatuaggio catturate dai corpi dei sospetti al momento della prenotazione. Per gentile concessione della Polizia di Stato del Michigan.* [2]

Lee et al. in [14], riportano l'utilità della biometria nel caso dell'identificazione delle vittime nei casi dell'attentato dell'11 settembre 2001 e del tsunami asiatico nel 2004. In questo ambito, l'utilizzo dei tatuaggi come tratti biometrici, ha permesso di riconoscere le vittime anche in presenza di gravi ustioni, in quanto i pigmenti si trovano in profondità e non in uno strato superficiale della pelle.

Come riportato anche in [11], le informazioni raccolte riguardo ai tatuaggi delle vittime devono essere il più accurate e oggettive possibile. Per le informazioni *ante-mortem* devono essere eseguite delle interviste con i parenti in cerca dei dispersi per poter raccogliere immagini allo scopo di generare un gran numero di dataset. *Post-mortem* i tatuaggi vengono solitamente documentati durante l'autopsia. Nel caso di corpi decomposti, alcuni metodi semplici, veloci ed economici, possono essere la fotografia infrarossa oppure il perossido di idrogeno; mediante questi strumenti è possibile ottenere informazioni adeguate sulla presenza di tatuaggi e sui motivi ricorrenti.

Rambhatla et al. [15] riportano come i tatuaggi possano essere visti come tratti distintivi nel processo di riconoscimento di vittime e sopravvissuti ai trafficanti di esseri umani. Uno dei segni distintivi di questi tatuaggi è sicuramente la bassa qualità di esecuzione, dovuta probabilmente alla natura *homemade* del tatuaggio stesso, come viene sottolineato dagli autori.

Inoltre, un altro grande ambito di applicazione dei tatuaggi come tratti biometrici è rappresentato dall'identificazione dei colpevoli, in quanto tramite i tatuaggi è possibile ricostruire il passato dell'individuo, la sua appartenenza a determinate organizzazioni criminali o gli anni trascorsi in prigione. Infatti, un gran numero di tatuaggi racchiude al loro interno significati nascosti e che permettono alle forze dell'ordine di identificare più velocemente i sospettati. Questa strategia investigativa, tuttavia, deve essere applicata con cautela, tenendo conto del contesto culturale e della libertà individuale di scelta dei tatuaggi, poiché non tutti i tatuaggi rappresentano necessariamente un coinvolgimento in attività criminali o illegali.

Processo di riconoscimento di un tatuaggio

Il processo di riconoscimento di un tatuaggio viene considerato un processo complesso che prevede diversi passaggi che possono essere suddivisi in due grandi fasi: pre-processing e riconoscimento [16].

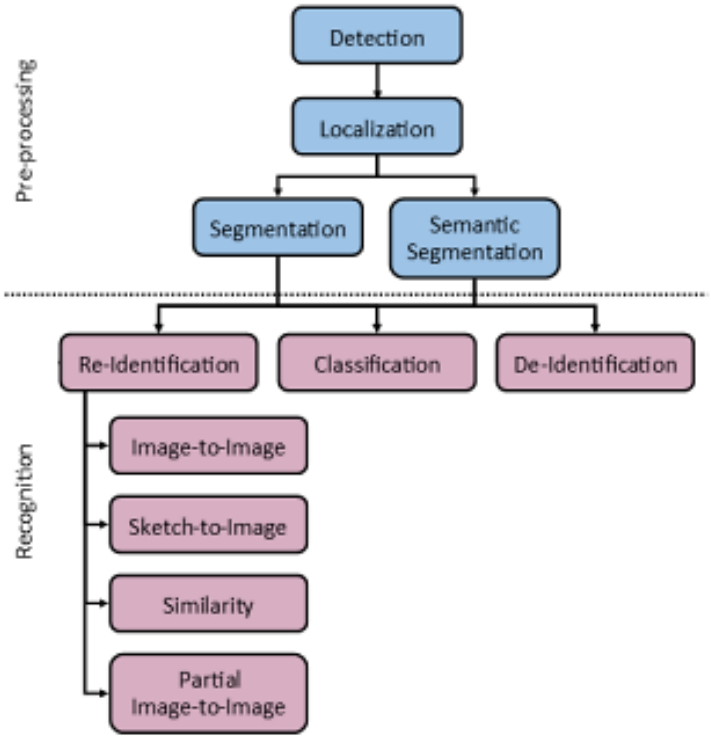

***Figura 5**. Step da effettuare per il processo di riconoscimento di un tatuaggio.* [16]

Lo step iniziale è quello di pre-processing ed ha lo scopo di preparare l'immagine in modo ottimale per il passaggio successivo; infatti, a partire da un immagine grezza si cerca di ottenerne una che sia filtrata dal rumore dello sfondo. In questo passaggio si possono identificare le seguenti problematiche:

- Rilevamento: determinare se una data immagine contiene o meno al suo interno un tatuaggio;
- Localizzazione: individuare il luogo preciso all'interno dell'immagine in cui il tatuaggio è presente e creare una ROI (*Region of Interest*) che lo circondi;
- Segmentazione: identificare il contorno reale del tatuaggio e restituire un'immagine contenente solamente il tatuaggio, eliminando lo sfondo;
- Segmentazione semantica: individuare eventuali sotto-immagini all'interno del tatuaggio principale allo scopo di isolare ogni singolo oggetto.

Il secondo step consiste nel vero e proprio riconoscimento del tatuaggio. A seconda della qualità del pre-processing sarà possibile ottenere un risultato migliore o meno. Anche in questo caso sono presenti diverse problematiche che devono essere investigate:

- Re-identificazione: prevede di ricercare l'immagine oggetto di studio all'interno di un dataset, restituendo in output la corrispondenza migliore. Questo processo può essere ulteriormente suddiviso in:
    - Immagine-a-immagine: data un immagine, fornire l'immagine più simile;

    - Disegno-a-immagine: dato un disegno ricreato a mano, restituire l'immagine che corrisponde al meglio;
    - Gruppi simili: identificare tatuaggi che presentano tratti simili a quello fornito in input, ma che non si tratti di una precisa corrispondenza;
    - Immagine parziale-a-immagine: ricavare, a partire da una sola porzione del tatuaggio, l'immagine più estesa che abbia la miglior corrispondenza possibile.
- Classificazione: data un'immagine, restituire una descrizione degli oggetti che la caratterizzano, allo scopo di creare un'etichetta per il tatuaggio stesso.
- De-identificazione: a partire dall'immagine contenente il tatuaggio, rimpiazzare il colore dei pixel corrispondenti al tatuaggio, in modo da rimuovere il tratto biometrico in essi contenuto.

Nel corso di questo elaborato, l'attenzione viene posta nella fase del pre-processing, in modo particolare nella fase del rilevamento e della localizzazione del tatuaggio all'interno dell'immagine. Questi passaggi iniziali sono cruciali per facilitare il successivo riconoscimento del tatuaggio, garantendo un processo fluido e accurato nella sua identificazione.

DATASET DI TATUAGGI

La presenza di dataset accessibili pubblicamente o privatamente svolge un ruolo cruciale in una vasta gamma di settori e contesti. Riferendoci alla creazione di dataset contenenti immagini di soggetti tatuati, possiamo osservare come essi costituiscono una risorsa preziosa per vari scopi.

In primo luogo, i dataset di tatuaggi sono importanti per l'addestramento di algoritmi di riconoscimento, come quelli utilizzati in applicazioni di sicurezza, identificazione personale o analisi di immagini online. Questi algoritmi imparano dai dati forniti nel dataset, migliorando progressivamente la loro capacità di identificare e categorizzare tatuaggi in diverse situazioni.

In secondo luogo, questi dataset costituiscono una risorsa essenziale per la creazione di una vasta banca dati di immagini di tatuaggi. Questa banca dati può essere utilizzata per scopi di comparazione e riconoscimento, consentendo a persone e organizzazioni di identificare tatuaggi specifici, analizzarne le tendenze e valutarne aspetti come l'originalità o l'autenticità.

Dataset attualmente implementati

Uno dei primi dataset, contenenti un gran numero di immagini, ad essere stato realizzato e reso disponibile è il dataset Tatt-C (*Tattoo Recognition Technology - Challenge*) [17], pubblicato nel 2015. Le immagini contenute sono state raccolte dalle agenzie governative durante varie azioni investigative e prevedono, pertanto, un'alta variabilità per quanto riguarda l'ambiente di cattura e il contenuto e la qualità dei tatuaggi. Insieme al dataset sono state proposte cinque diverse sfide verso cui i candidati potevano rivolgere la loro attenzione:

1. Similarità tra tatuaggi: data un'immagine di un tatuaggio, identificare all'interno del database un'immagine visivamente simile o contenente tatuaggi ad essa correlati;
2. Identificazione di tatuaggi: data un'immagine di un tatuaggio, identificare diverse istanze dello stesso soggetto e dello stesso tatuaggio all'interno del database;
3. Regione di interesse: data una porzione ristretta di un immagine di un tatuaggio, identificare il tatuaggio maggiormente esteso, che la contiene;
4. Media misti: date altre tipologie di immagini (disegni, graffiti, immagini digitali), identificare immagini visivamente simili o tatuaggi correlati presenti all'interno del dataset;
5. Rilevamento di tatuaggi: data un'immagine, identificare se essa contiene o meno un tatuaggio.

Nella Tabella 1 è presente un riassunto dei vari ambiti di applicazione, con riportati anche i numeri di immagini disponibili per ognuno di essi.

	Tattoo Similarity	Tattoo Identification	Region of Interest	Mixed Media	Tattoo Detection
Use Case	Match visually similar or related tattoos from different subjects	Match different instances of the same tattoo from the same subject over time	Match small region of interest contained in a larger tattoo	Match visually similar or related tattoos across different mediums	Detect whether an image contains a tattoo
Utility Example	Group Affiliation	Person identification	Person identification	Intelligence gathering	Database construction and maintenance
Task	One-to-many search	One-to-many search	One-to-many search	One-to-many search	Classification
Types of images	Tattoos	Tattoos	Tattoos	Tattoos, sketches, graffiti, computer graphics	Tattoos, faces
Number of images	2212	372	454	454	2349
Number of probes	851	157	297	181	2349

***Tabella 1**. Riassunto dei casi d'impiego del Tatt-C* [17]

Per ognuno di questi ambiti di applicazione, nelle figure successive, vengono riportate alcune immagini presenti all'interno del dataset.

***Figura 6**. Esempi di insiemi di immagini dal caso d'uso "Similarità tra tatuaggi".* [17]

***Figura 7**. Esempi di insiemi di immagini dal caso d'uso "Identificazione di tatuaggi".* [17]

***Figura 8**. Esempi di insiemi di immagini dal caso d'uso "Regione di Interesse".* [17]

***Figura 9.** Esempi di insiemi di immagini dal caso d'uso "Media Misti".* [17]

***Figura 10**. Esempi di immagini dal caso d'uso "Rilevamento di tatuaggi".* [17]

Altri articoli precedenti allo sviluppo di questo dataset impiegavano immagini raccolte da internet [14], [18], [19] ricavando quindi rispettivamente 4323, 100 e 2157 immagini.

Per aumentare il numero delle immagini disponibili gli autori di [19] hanno applicato una serie di trasformazioni alle immagini raccolte, in modo tale da aumentare la variabilità dei dati, ottenendo così un totale di 43140 immagini trasformate. Esse sono state modificate come segue: due diverse intensità di sfocatura, rumore aggiuntivo e illuminazione; sei diverse proporzioni, quattro diverse rotazioni e quattro diverse modifiche di colore.

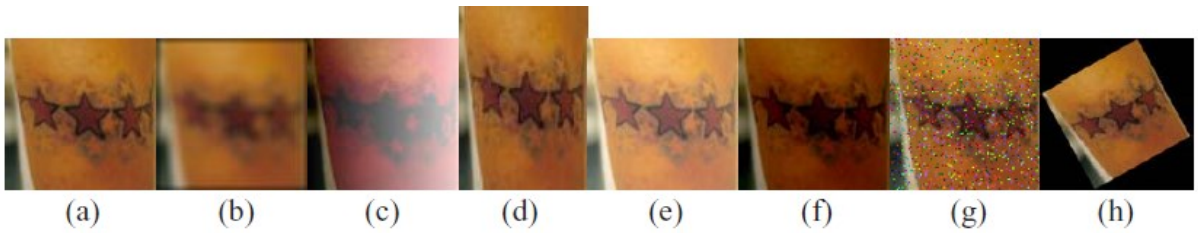

***Figura 11**. Esempi di trasformazioni delle immagini di tatuaggi: (a) originale, variazioni dovute a (b) sfocatura, (c) illuminazione, (d) proporzione, (e) e (f) colore, (g) rumore aggiuntivo e (h) rotazione.* [19]

Anche Heflin et al. [20] hanno valutato il processo di riconoscimento di cicatrici, tatuaggi ed altre tipologie di segni, utilizzando 18922 immagini ricavate da internet.

In [16] sono stati compilati due dataset denominati *TattDetectB* e *TattDetectF*, impiegando immagini estratte dalla rete attraverso Bing e Flickr, rispettivamente. Ciascuno di questi dataset ha consistito di 2.000 immagini raffiguranti individui, distribuite equamente tra due categorie: immagini con presenza di tatuaggi e immagini prive di tatuaggi. L'acquisizione delle immagini destinate a questo dataset è stata ottenuta mediante la scansione di pagine web, l'identificazione di immagini pertinenti e la loro successiva cattura.

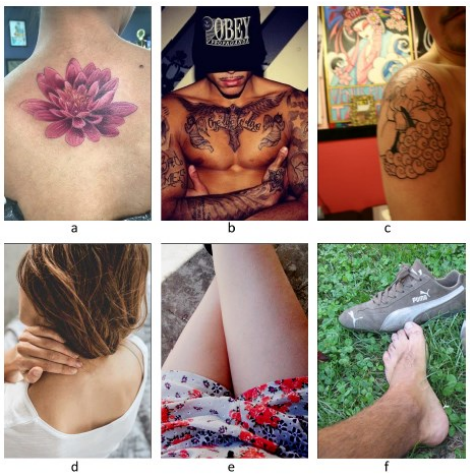

***Figura 12.** Esempio di immagini utilizzate per il compito di rilevamento, raffiguranti persone con e senza tatuaggi.* [16]

Oltre alle immagini scaricate dal web, lo studio condotto da Lee et al. [14] prevede anche l'utilizzo di 69507 immagini ricavate dall'archivio di stato della Polizia del Michigan, il quale non risulta disponibile ad uso pubblico. Anche nell'articolo pubblicato nel 2012 da Manger et al. [21] sono state utilizzate 327049 fotografie raccolte dalla polizia tedesca.

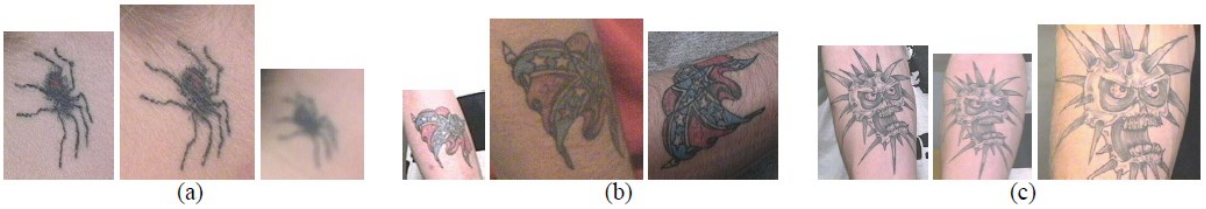

Figura 13. *Esempi di immagini di copie di tatuaggi nel database della Polizia del Michigan. Tre tatuaggi ciascuno in (a), (b) e (c) sono copie dello stesso tatuaggio. Queste immagini di copie di tatuaggi sono state acquisite in momenti diversi.* [14]

Han et al. [23] e Hrkać et al. [24] hanno condotto i loro studi anche mediante l'utilizzo di 890 immagini ricavate utilizzando il dataset DeMSI, il quale corrisponde ad una porzione del ben più ampio dataset ImageNet, contenente oltre 15 milioni di immagini appartenenti a oltre 22.000 categorie diverse [25].

Figura 14. *Esempi di immagini di tatuaggi provenienti dai quattro database di immagini di tatuaggi utilizzati negli esperimenti: (a) Tatt-C, (b) Flickr, (c) DeMSI e (d) il nostro dataset WebTattoo.* [23]

Nell'attuale stato dell'arte, allora, non esistono dataset pubblici e contenenti un numero rilevante di immagini di soggetti tatuati da poter utilizzare nei vari campi di applicazione. La tendenza generale è quella di ottenere l'accesso a dataset privati, rispettando i vincoli imposti dagli autori, oppure quella di creare una propria banca dati, scaricando le immagini dalla rete.

Scopi e utilità di un dataset

Nel contesto forense, un registro di tatuaggi assume un ruolo di rilevanza significativa, fornendo una serie di funzionalità di importanza cruciale. In primo luogo, esso agevola l'identificazione individuale attraverso l'analisi dei tatuaggi, poiché questi possono costituire attributi distintivi fondamentali per il riconoscimento di soggetti. Pertanto, consente alle autorità di polizia di catalogare e confrontare i disegni tatuati rinvenuti su vittime, sospetti o individui non identificati, dimostrandosi particolarmente efficace nelle inchieste relative a individui implicati in reati o nelle ricerche di persone scomparse.

Inoltre, il registro di tatuaggi può essere impiegato nell'analisi di gruppi o associazioni criminali. Tale utilizzo è giustificato dal fatto che i tatuaggi possono fungere da segni distintivi o simboli di affiliazione a organizzazioni illecite. Un archivio di tatuaggi consente alle autorità competenti di individuare e ricostruire legami tra individui o gruppi coinvolti in attività illegali, contribuendo in maniera significativa alle investigazioni relative a crimini organizzati, traffico di stupefacenti, estorsioni e altri reati connessi.

Un registro di tatuaggi può anche risultare prezioso nel rilevare e analizzare tendenze e schemi ricorrenti nei disegni dei tatuaggi. Ad esempio, potrebbe emergere che un determinato simbolo o stile di tatuaggio è ampiamente diffuso tra membri di una specifica organizzazione criminale. Queste informazioni possono costituire indicazioni cruciali per le indagini e l'identificazione di gruppi delinquenziali.

Infine, un archivio di tatuaggi agevola la condivisione di informazioni e la collaborazione tra diverse agenzie di polizia e organizzazioni internazionali. La standardizzazione nell'archiviazione dei disegni dei tatuaggi, oltre alla loro accessibilità da parte di giurisdizioni diverse, può agevolare lo scambio di dati rilevanti per inchieste transnazionali e rafforzare la cooperazione tra le competenti autorità.

Implicazioni sociali ed etiche

È importante notare che l'utilizzo di un database di tatuaggi in ambito forense richiede la considerazione di questioni legate alla privacy e alla protezione dei dati personali. È fondamentale garantire che le informazioni siano trattate in conformità alle leggi e alle normative sulla privacy e che siano adottate misure di sicurezza adeguate a prevenire un accesso non autorizzato.

Le normative sulla privacy possono variare da Paese a Paese, ma ci sono alcune leggi e regolamenti comuni che spesso vengono considerati quando si tratta di gestire un database contenente informazioni personali, come un database di tatuaggi.

Uno di questi è il Regolamento generale sulla protezione dei dati (GDPR) [26]. Il GDPR è un regolamento dell'Unione Europea che riguarda la protezione dei dati personali. Esso stabilisce una serie di principi e requisiti che devono essere rispettati quando si tratta di raccogliere, archiviare e gestire dati personali, incluso l'archivio dei tatuaggi. Ad esempio, richiede il consenso informato degli individui per la raccolta e l'elaborazione dei loro dati personali, fornisce diritti agli interessati come il diritto all'accesso e alla cancellazione dei dati, e richiede misure di sicurezza adeguate a proteggere i dati personali.

Indipendentemente dalle normative specifiche, è importante ottenere il consenso informato degli individui prima di raccogliere e archiviare informazioni sui loro tatuaggi. Il consenso dovrebbe essere volontario, specifico, informato e inequivocabile, e gli individui dovrebbero essere consapevoli di come saranno utilizzate le informazioni e dei loro diritti relativi alla privacy.

Qualsiasi database contenente informazioni personali, compresi i disegni dei tatuaggi, dovrebbe adottare misure di sicurezza adeguate per proteggere i dati da accessi non autorizzati, perdita o divulgazione accidentale. Ciò potrebbe includere l'adozione di procedure di sicurezza tecniche e organizzative, come l'uso di crittografia, il controllo degli accessi, la conservazione dei dati solo per il tempo necessario e la formazione del personale sulla gestione sicura dei dati.

METODOLOGIE DI RECUPERO DEI TATUAGGI

Le prime metodologie implementate per la ricerca di immagini di tatuaggi si basavano sulla corrispondenza basata su parole chiave o metadati. Ad esempio, le agenzie governative solitamente seguono lo standard ANSI/NIST-ITL 1-2000 [3] per assegnare una singola parola chiave a ciascuna immagine presente nel dataset.

Tuttavia, un sistema di ricerca che si basa sull'utilizzo di parole chiave può comportare diverse limitazioni nella pratica [1]:

- Le classi definite da ANSI/NIST-ITL offrono un vocabolario limitato che è insufficiente per descrivere vari modelli di tatuaggi;
- potrebbero essere necessarie più parole chiave per descrivere adeguatamente un'immagine di tatuaggio;
- l'annotazione umana è soggettiva e diversi soggetti possono dare etichette molto diverse alla stessa immagine di tatuaggio.

A causa di queste carenze nei sistemi di ricerca basati sulle *keywords*, si è iniziato allora a sviluppare tecniche di recupero di immagini basate su contenuti (*content-based image retrieval*, CBIR) per migliorare l'efficienza e l'accuratezza della ricerca di tatuaggi [1]. Il CBIR mira a estrarre determinate caratteristiche, ad esempio bordi, colore e texture, che possono riflettere il contenuto di un'immagine ed usarle poi per identificare immagini con alta similarità visiva. Ad esempio, nell'operazione di *matching* dei tatuaggi sono stati utilizzati l'istogramma del colore e il correlogramma, i momenti della forma e le caratteristiche di coerenza della direzione dei bordi [19]. Analogamente, sono state utilizzate caratteristiche globali e locali di bordi e colore [27], e la distanza euclidea tra vettori viene calcolata per misurare la similarità tra due immagini.

Uno dei metodi più popolari tra i primi sistemi che utilizzano CBIR per la ricerca dei tatuaggi è il modello "*bag-of-words*" (BoW), il quale utilizza le caratteristiche SIFT [28] come viene implementato in [1] e in [21].

Con l'avvento e con il successo del *deep learning* in molte attività di visione artificiale, l'attenzione dei metodi di riconoscimento dei tatuaggi si sta spostando dalle caratteristiche e dai modelli creati manualmente ai metodi basati sul *deep learning* stesso.

La tecnica del *transfer learning*, che verrà illustrata all'interno del capitolo successivo, ha dimostrato risultati eccellenti in molti problemi di classificazione, specialmente nel campo dell'elaborazione delle immagini, e può essere applicata a questo specifico problema.

Le reti utilizzate per l'applicazione di questo approccio possono essere varie. Ad esempio, l'architettura AlexNet [25], vincitrice della sfida ImageNet del 2012, è stata utilizzata con successo per classificare immagini contenenti tatuaggi rispetto a quelle che non ne contengono in diversi studi, come in [24], [29], [30], [31]. Gli autori di [24] propongono, inoltre, un'architettura ispirata al modello VGGNet proposto da Simonyan e Zisserman [32]. Un'altra rete neurale, denominata Faster RCNN, è stata valutata sia per la classificazione di immagini di tatuaggi rispetto a immagini senza tatuaggi che per la localizzazione di tatuaggi in [23] e [33].

In [16] viene pubblicata una tabella (Figura 15) riportante i risultati ottenuti in letteratura, al momento della pubblicazione dell'articolo. La colonna "Best Result 1" fa riferimento al caso in cui sia il *training* della rete, che il *testing*, siano stati eseguiti con lo stesso dataset. La colonna "Best Result 2" fa riferimento, invece, al caso in cui siano stati usati dataset diversi.

Table I
TATTOO DETECTION PUBLISHED RESULTS

Ref.	Year	Method	Best Result 1	Best Result 2
[4]	2015	not cited	96.30% acc.	-
[2]	2016	CNN	98.80% acc.	93.78%
[5]	2016	AlexNet + 2-Class SVM	99.83% acc.	-
[6]	2017	AlexNet + 2-Class SVM	99.83% acc.	-
[7]	2016	Decision tree	52.38% acc.	-
[8]	2016	Faster R-CNN	98.25% acc.	80.66%
[9]	2019	Faster R-CNN	87.10% recall (WebTattoo) 61.70% recall (Tatt-C)	80.00%

Figura 15. *Risultati pubblicati in merito al problema di rilevamento dei tatuaggi.* [16]

Inoltre, gli autori di [23] rendono disponibile una tabella riassuntiva in cui vengono elencati i principali articoli, disponibili al 2019 per quanto riguarda i metodi pubblicati per l'identificazione e il recupero dei tatuaggi all'interno delle immagini.

TABLE 1
A Summary of Published Methods on Tattoo Identification and Retrieval

Publication	Detection model	Feature and retrieval model	Tattoo database #images (query; target)	Results
Jain et al. [1], [4] (2007,2012)	Gradient thresholding	Color histogram and correlogram; shape moments; edge direction coherence; Fusion of per feature similarities	Tattoos from web (2,157; 43,140)[1]	46% prec.@60% recall
Acton and Rossi [8] (2008)	Active contour segmentation and skin detection	Global and local features of edge and color; Vector-wise euclidean distance	Recreational (30; rv 4;000)[2]; Gang (39; rv 4;000)[2]	Recreational: 94.7% acc.@rank-1; Gang: 82.2% acc. @rank-1
Jain et al. [16] (2009)	Pre-cropped tattoos	SIFT features with geometric constraint; indexing with location and keyword; Keypoint-wise matching	MSP (1,000; 63,592)	85.9% acc.@rank-1
Li et al. [17] (2009)	n/a	SIFT features; Bag-of-words; Re-ranking	MSP and ESP (995; 101,754)[3]	67% acc.@rank-1
D. Manger [9] (2012)	n/a	SIFT features; Bag-of-words, hamming embedding, and weak geometry consistency	German police (417; 327,049)	78% acc.@rank-1
Heflin et al. [18] (2012)	Automatic GrabCut and quasi connected components	LBP-like features, SVM	Tattoo classification (50; 500)[4]	85% acc.@10% FAR on average, for 15 classes
Han and Jain [10] (2013)	Pre-cropped tattoos	SIFT features; Sparse representation classification	MSU Sketch Tattoo (100; 10,100)	48% acc.@rank-100
Wilber et al. [19] (2014)	Pre-cropped tattoos	Exemplar code using HoG features; Random forest classifier	238 tattoos of 5 classes	63.8% avg. acc. for 5 classes
Xu et al. [20] (2016)	Skin segmentation and block based decision tree	Boundary features; Shape matching via coherent point drift	Full body tattoo sketch (547; 1,641)	52.38% acc.@rank-50
Kim et al. [21] (2016)	Graphcut	n/a	Tatt-C (Detection): 6,308 images; Evil (Detection): 1,105 images	Tatt-C: 70.5% acc. @41%recall; Evil: 69.9% acc. @67.0%recall
Xu et al. [11] (2016)	Modified AlexNet (tattoo vs. non-tattoo)	n/a	Tatt-C (tattoo vs. non-tattoo) (1,349; 1000)[4]; Flickr (tattoo vs. non-tattoo) (5,740; 4,260)[4]	Tatt-C (tattoo vs. non-tattoo): 98.8%; Flickr (tattoo vs. non-tattoo): 78.2%
Sun et al. [22] (2016)	Faster R-CNN	n/a	Tatt-C: tattoo vs. non-tattoo (1,349; 1000)[4]; Flickr: tattoo vs. non-tattoo (5,740; 4,260)[4]	Tatt-C (tattoo vs. non-tattoo): 98.25% Tatt-C (localization): 45%@0.1FPPI; Flickr (tattoo vs. non-tattoo): 80.66%
Di and Patel [23], [24] (2016)	AlexNet and SVM (tattoo vs. non-tattoo)	Siamese network with triplet or contrastive loss	Tatt-C: tattoo vs. non-tattoo (1,349; 1,000)[4]; mixed media (181; 55)	Tattoo vs. non-tattoo: 99.83%; Mixed media: 56.9% acc.@rank-10

TABLE 1
(Continued)

Publication	Detection model	Feature and retrieval model	Tattoo database #images (query; target)	Results
Proposed approach	Deep end-to-end learning for joint detection and compact representation learning		Tatt-C: detection: 7,526 images identification (157; 4,375); Flickr (detection): 5,740 images; DeMSI (identification): 890 images; WebTattoo (500, rv300K)	Detection (localization) Tatt-C: 61:7% recall@0.1FPPI WebTattoo: 87:1% recall@0.1FPPI Tattoo search WebTattoo (photo): 60:1% mAP (w/o background) 25:3% mAP (300K background) WebTattoo (sketch): 37:2% mAP (w/o background) Tattoo identification WebTattoo: 63:5% acc.@rank-1 (w/o background) 28:0% acc.@rank-1 (300K background) Tatt-C: 99:2% acc. @rank-1

[1] Twenty different image transformations were applied to 2,157 tattoo images to generate 43,140 synthetic tattoo images. [2] Forty different image transformations were applied to 100 tattoo images to generate 40,000 synthetic tattoo images. [3] 40; 000 images were randomly selected from the ESP game dataset to populate the tattoo dataset. [4](a, b) denotes the number of positive and negative tattoo images per class. [5](a, b) denotes the number of tattoo and non-tattoo images.

Figura 16. *Riassunto dei metodi pubblicati in merito all'identificazione e recupero di tatuaggi.* [23]

Capitolo 2
Background tecnologico

Nell'epoca contemporanea, l'universo della tecnologia è stato trasformato in modo radicale grazie al progresso senza precedenti nel campo del *Deep Learning*. Questo capitolo ha l'obiettivo di introdurre questo mondo di innovazione e sperimentazione, in cui le reti neurali si ergono come i pilastri fondamentali.

Il *Deep Learning* è una disciplina che ha rivoluzionato il modo in cui le macchine comprendono e interpretano dati complessi. Questa metodologia si basa su reti neurali artificiali, le quali mirano a estrarre rappresentazioni informative da dati grezzi, consentendo di affrontare compiti sofisticati come il riconoscimento di oggetti, la classificazione e la generazione di contenuti. In questo contesto, approfondiremo le reti neurali convoluzionali (CNN), che risultano particolarmente efficaci nel trattare dati visivi.

Un elemento chiave della nostra indagine sarà anche l'approccio del *Transfer Learning*, una strategia intelligente che consente di sfruttare il potenziale di reti neurali pre-addestrate, come AlexNet, ResNet-50, VGG-16 e Inception-V3, per compiti specifici. Questo metodo di apprendimento si basa sulla capacità delle reti neurali di trasferire conoscenze da un dominio all'altro, accelerando notevolmente il processo di addestramento e migliorando le prestazioni dei modelli.

Inoltre, verrà posto l'accento sull'importanza degli strumenti utilizzati nella sperimentazione. Unity Game Engine, rinomato per la sua versatilità, permette di sviluppare simulazioni interattive e ambienti di apprendimento per reti neurali. Parallelamente, MATLAB, con i suoi strumenti specializzati come *Deep Network Designer* e *Image Labeler*, consente di progettare, formare e valutare le reti neurali, semplificando il processo sperimentale e l'analisi dei dati.

DEEP LEARNING E RETI NEURALI

Il *Deep Learning* è un tipo di apprendimento automatico (*machine learning*) modellato secondo il funzionamento del cervello umano, che apprende attraverso molteplici livelli di rappresentazione e richiede meno ingegneria manuale, ottenendo previsioni più accurate con grandi quantità di dati. [34]

Esso permette quindi di addestrare i computer a svolgere delle attività che risultano naturali per l'uomo. Può risultare una tecnologia fondamentale in molti ambiti come, ad esempio, nella progettazione di auto a guida autonoma, in quanto consente di riconoscere segnali di stop oppure di distinguere pedoni da altri elementi caratteristici della strada [35]. Un ulteriore esempio è dato dalla capacità di apprendere attività di classificazione di immagini, testi oppure suoni. I modelli possono raggiungere livelli di precisioni superiori a quelli ottenuti dall'uomo, andando così a rappresentare un utilissimo strumento a servizio dell'umanità. È in questo contesto, allora, che il *deep learning* può essere applicato in ambito forense.

Queste tecniche risultano basate su reti neurali artificiali profonde, con molteplici strati di neuroni, che sono ispirate alla struttura del cervello umano. Le reti neurali profonde sono costruite impilando numerosi strati, noti come *layer*. Ogni *layer* è formato da un numero variabile di neuroni, o nodi, che sono collegati agli strati precedenti e successivi. Durante l'allenamento, i pesi di queste connessioni vengono regolati in modo da minimizzare l'errore tra le previsioni del modello e i dati di addestramento.

Il *Deep Learning* è una disciplina che, sebbene concepita negli anni '80, ha recentemente guadagnato notevole rilevanza per due ragioni fondamentali [35]. In primo luogo, richiede l'accesso a un vasto *corpus* di dati etichettati per l'addestramento efficace dei modelli. Ad esempio, per applicazioni come lo sviluppo di veicoli autonomi, è necessario disporre di milioni di immagini e migliaia di ore di video opportunamente annotati. Questa necessità di dati di addestramento abbondanti è una delle barriere principali per l'adozione diffusa di questa tecnica. In secondo luogo, il *Deep Learning* richiede una considerevole capacità computazionale per addestrare reti neurali profonde con successo. Questa potenza di calcolo è spesso fornita da unità di elaborazione grafica (GPU) ad alte prestazioni, che sono dotate di architetture parallele altamente efficienti per l'esecuzione di operazioni intensive tipiche del *Deep Learning*. In combinazione con sistemi di calcolo distribuito come cluster di server o il *cloud computing*, le squadre di sviluppo possono notevolmente ridurre i tempi necessari per l'addestramento di reti di *Deep Learning*, passando da diverse settimane a poche ore. Questo avanzamento tecnologico ha contribuito in modo significativo all'adozione su larga scala di questo approccio in varie applicazioni.

Reti Neurali Artificiali (ANN)

Una Rete Neurale Artificiale (*Artificial Neural Network*, ANN) rappresenta un modello matematico computazionale non-lineare per l'elaborazione delle informazioni, con architetture ispirate alla biologia organizzativa dei neuroni [36].

Storicamente, l'introduzione del concetto di ANN è attribuita a Frank Rosenblatt [37], il quale, nel 1958, sfruttò le conoscenze contemporanee sulla neurofisiologia per sviluppare un modello matematico atto a simulare il processo di elaborazione delle informazioni nei neuroni umani. Questo primo design fu denominato "*perceptron*" ed è ancora alla base di alcuni dei sistemi ANN avanzati attualmente in uso.

Questa tipologia di modelli si caratterizza per una struttura costituita da elementi di elaborazione computazionale di base interconnessi, noti come neuroni artificiali, che operano in parallelo eseguendo compiti computazionali semplici e similari. Nella Figura 17 viene illustrato il funzionamento interno di un singolo neurone.

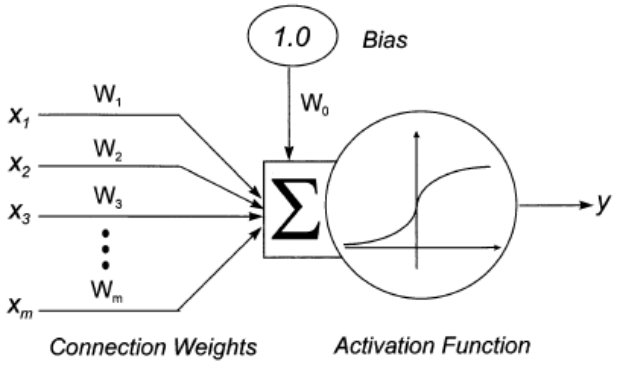

***Figura 17**. Funzionamento interno di un singolo neurone.* [36]

Gli elementi caratterizzanti questo modello sono:

- x_1, x_2, …, x_m, i quali rappresentano gli input per il neurone. Per i neuroni presenti nei livelli nascosti o di output, questi input derivano dagli output dei neuroni nel livello precedente;
- w_1, w_2, . . ., w_m, ovvero i pesi delle connessioni corrispondenti;
- w_0, il quale rappresenta un termine di *bias* che viene aggiunto alla somma ponderata totale degli input per agire come una soglia per spostare il "punto di attivazione" del neurone;
- Una funzione di attivazione sigmoide, che agisce come un "interruttore soft" per produrre un output, y, maggiore o minore, a seconda della somma ponderata totale degli input e del termine di *bias.*

Prima di venire indirizzato verso il neurone, ciascun input è soggetto a una moltiplicazione per un coefficiente di peso che riflette la forza eccitatoria o inibitoria della connessione dalla sorgente di *input* al neurone. Successivamente, il neurone aggrega i vari input mediante un'operazione di sommatoria. La somma degli input ponderati, oltre a un termine di *bias*, viene quindi sottoposta a una funzione di attivazione che determina se il neurone genererà un *output* apprezzabile.

Tra le opzioni comuni per la funzione di attivazione viene utilizzata la funzione logistica [38], spesso indicata come "curva a S" o "funzione sigmoide", espressa come

$$f(x) = \frac{1}{(1 + e^{-x})}$$

dove "f(x)" rappresenta il valore di *output* della funzione logistica per un dato input "x".

Il termine di *bias*, oltre alle sue interpretazioni matematiche e computazionali, può essere visto come una soglia da superare affinché la somma ponderata degli input superi il valore di 0.5, consentendo al neurone di emettere un segnale di output.

La capacità di elaborazione delle informazioni di una ANN dipende principalmente dal tipo e dal numero di neuroni nella rete, nonché dall'organizzazione topologica delle connessioni tra di essi [36]. La conoscenza o la capacità di elaborazione di una ANN è definita in maniera distributiva in base ai valori effettivi dei pesi delle connessioni tra i neuroni nella rete stessa. Tale conoscenza è acquisita attraverso una fase di apprendimento, durante la quale esempi di dati da elaborare vengono ripetutamente presentati al modello ANN, e i pesi delle connessioni all'interno della ANN sono adattati in modo dinamico per soddisfare specifici obiettivi prestabiliti di performance.

Nel corso degli anni, sono stati sviluppati e documentati vari tipi di reti neurali artificiali (ANN), ciascuno caratterizzato da proprietà specifiche che li rendono più idonei a particolari compiti rispetto ad altri.

Oltre al modello "*perceptron*", impiegato ancora in problemi di classificazione binaria, alcuni esempi sono rappresentati da *Multi-Layer Perceptron* (MLP), reti neurali ricorrenti (RNN), *Long Short-Term Memory* (LSTM) e reti neurali convoluzionali (CNN).

Queste ultime sono quelle utilizzate nel corso di questo elaborato.

Addestramento di un modello di deep learning e la tecnica del transfer *learning*

Dopo aver determinato la struttura (numero di livelli, neuroni, connessioni), l'addestramento di una rete neurale consiste nel determinare il valore dei pesi *w* che permettono di ottenere la miglior corrispondenza tra output prodotto dalla rete e l'output desiderato.

Nei contesti come quello dell'elaborazione delle immagini, emergono due diversi approcci di apprendimento: l'apprendimento supervisionato e l'apprendimento non supervisionato [39].

L'apprendimento supervisionato si caratterizza per l'uso di input precedentemente etichettati, i quali agiscono come obiettivi predeterminati. In ogni esempio di addestramento, si presentano un insieme di valori di input (vettori) e uno o più valori di output specificamente designati. L'obiettivo principale di questa metodologia di addestramento è la riduzione dell'errore di classificazione complessivo dei modelli, ottenuta attraverso il calcolo accurato del valore di output corrispondente a ciascun esempio di addestramento.

Dall'altro lato, l'apprendimento non supervisionato si distingue per il fatto che l'insieme di addestramento non include etichette predefinite. Il successo in questo contesto è solitamente definito in base alla capacità della rete di ridurre o aumentare una specifica funzione di costo associata.

Tuttavia, risulta importante sottolineare che la maggior parte delle attività di riconoscimento di modelli incentrate sull'immagine si basa prevalentemente sulla classificazione ottenuta mediante l'uso dell'apprendimento supervisionato.

Focalizzando poi l'attenzione sul riconoscimento di oggetti, ovvero il problema a cui ci si è ricondotti per lo sviluppo di questo elaborato, è possibile osservare come ci siano tre diversi approcci per eseguire il *training* di un modello di *deep learning*.

La prima possibilità è quella di addestrare un modello da zero [35]. Per poterlo fare è necessario, innanzitutto, raccogliere un set di dati etichettati di grandi dimensioni. Dopo di che, risulta fondamentale progettare un'architettura di rete ottimale, in modo che possa apprendere le feature e il modello. Questa tecnica non viene spesso applicata, in quanto l'addestramento richiede tempi molto lunghi, riconducibili a giorni o settimane, a causa della grande quantità di dati necessaria.

Un approccio di *Deep Learning* meno comune ma altamente specializzato coinvolge l'utilizzo della rete neurale come estrattore di feature [40]. In questa metodologia, ciascun *layer* della rete neurale è concepito per apprendere specifiche caratteristiche dalle immagini o dai dati di input.

Ciò consente l'estrazione di tali *feature* dalla rete in qualsiasi fase del processo di addestramento. Le caratteristiche così estratte possono poi essere impiegate come input in modelli di *Machine Learning* tradizionali, come le *Support Vector Machines* (SVM) o altri classificatori. Questo approccio offre un vantaggio notevole poiché sfrutta la capacità della rete neurale di catturare rappresentazioni complesse e informative dei dati, consentendo l'utilizzo di tali rappresentazioni per il successivo processo di classificazione o previsione all'interno del contesto del *Machine Learning*.

La metodologia più utilizzata per poter addestrare una rete neurale è quella del *transfer learning*. Come riportato anche dagli autori in [41] il concetto di *transfer learning* può inizialmente avere origini nella psicologia dell'istruzione. Infatti, secondo la teoria generale del trasferimento, proposta dallo psicologo C. H. Judd, questo concetto sembra derivare dalla generalizzazione dell'esperienza. È possibile effettuare il trasferimento da una situazione all'altra, a condizione che una persona generalizzi la propria esperienza. Secondo questa teoria, il prerequisito del trasferimento è che debba esistere una connessione tra due attività di apprendimento.

In [41] viene riportato inoltre un semplice esempio intuitivo secondo cui una persona che ha imparato a suonare il violino può apprendere il pianoforte molto più rapidamente rispetto agli altri, poiché entrambi sono strumenti musicali e condividono alcune conoscenze comuni, come banalmente quella delle note o del tempo musicale.

Nella Figura 18 vengono riportati alcuni esempi intuitivi di problemi riguardanti il *transfer learning*.

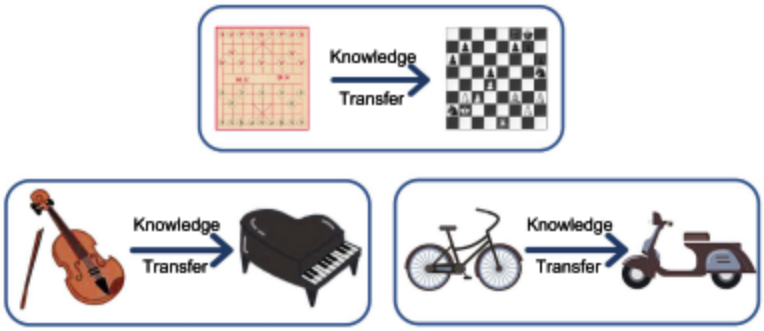

***Figura 18**. Esempi intuitivi di transfer learning.* [41]

Ci sono diverse possibilità per categorizzare le tecniche di *transfer learning*. Zhuang et al. [41] riportano una classificazione che distingue inizialmente tra la categorizzazione del problema oppure della soluzione.

I problemi di *transfer learning* possono essere suddivisi in tre categorie: *transfer learning* trasduttivo, induttivo e non supervisionato. Queste tre categorie possono essere interpretate da un punto di vista di etichettatura.

Il *transfer learning* trasduttivo si riferisce alle situazioni in cui le informazioni sulle etichette provengono solo dal dominio di origine. Se, al contrario, sono disponibili le informazioni sulle etichette delle istanze del dominio di destinazione, lo scenario può essere categorizzato come *transfer learning* induttivo. Infine, se le informazioni sulle etichette sono sconosciute sia per il dominio di origine che per quello di destinazione, la situazione è nota come *transfer learning* non supervisionato.

Un'altra categorizzazione si basa sulla coerenza tra gli spazi delle *feature* e degli spazi delle etichette del dominio di origine e del dominio di destinazione.

Definiti:

X^S come spazio delle *feature* del dominio di origine;

X^T come spazio delle *feature* del dominio di destinazione;

Y^S come spazio delle etichette del dominio di origine;

Y^T come spazio delle etichette del dominio di destinazione;

Se $X^S = X^T$ e $Y^S = Y^T$, lo scenario viene definito *transfer learning* omogeneo.

Altrimenti, se $X^S \neq X^T$ e/o $Y^S \neq Y^T$, lo scenario è definito *transfer learning* eterogeneo.

Secondo l'indagine [42], gli approcci di transfer learning possono essere categorizzati in quattro gruppi: *instance-based*, *feature*-b*ased*, *parameter-based* e *relational-based*.

Gli approcci di transfer learning basati su istanze si basano principalmente sulla strategia di pesatura delle istanze. Gli approcci basati sulle caratteristiche trasformano le caratteristiche originali per creare una nuova rappresentazione delle caratteristiche; possono essere ulteriormente suddivisi in due sottocategorie, cioè asimmetrici e simmetrici. Gli approcci asimmetrici trasformano le caratteristiche del dominio di origine per farle corrispondere a quelle del dominio di destinazione. Al contrario, gli approcci simmetrici cercano di trovare uno spazio di caratteristiche latenti comune e quindi trasformano sia le caratteristiche del dominio di

origine che quelle del dominio di destinazione in una nuova rappresentazione delle caratteristiche.

Gli approcci di *transfer learning* basati sui parametri trasferiscono la conoscenza a livello di modello/parametri. Gli approcci di *transfer learning* basati sulle relazioni si concentrano principalmente sui problemi nei domini relazionali. Tali approcci trasferiscono le relazioni logiche o le regole apprese nel dominio di origine al dominio di destinazione.

Nella Figura 19 viene riportato uno schema riassuntivo di tutte le tipologie di *transfer learning*.

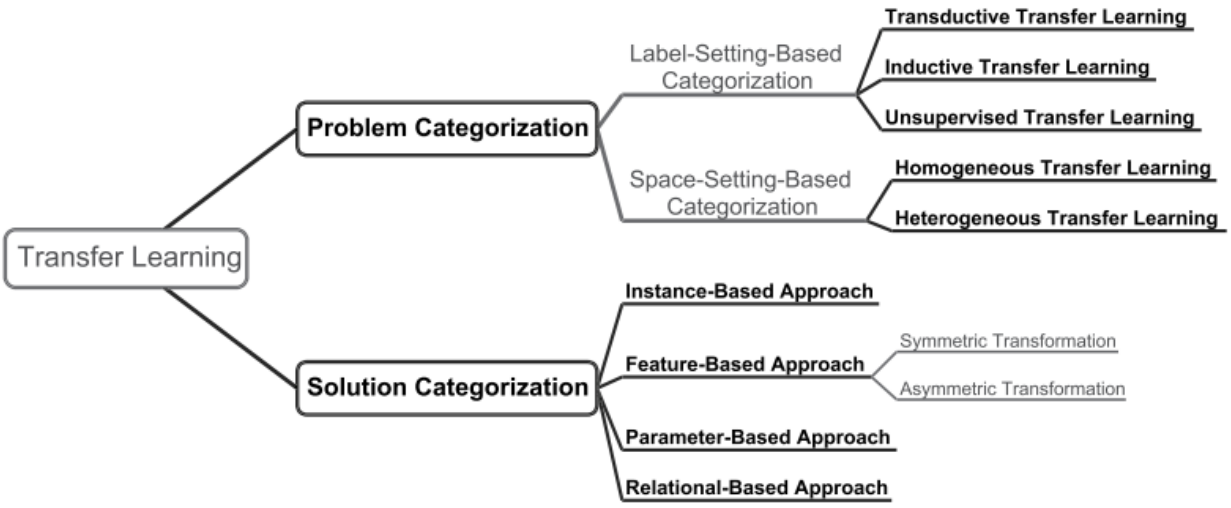

***Figura 19**. Categorie di transfer learning* [41]

RETI NEURALI CONVOLUZIONALI (CNN)

Le CNN, o *Convolutional Neural Networks*, sono un elemento fondamentale nell'ambito dell'apprendimento automatico e dell'elaborazione delle immagini. La loro storia affonda le radici negli anni '90, quando Yann LeCun e altri ricercatori iniziarono a sviluppare questa architettura per affrontare problemi di riconoscimento di immagini [43]. I successi iniziali portarono alla diffusione delle CNN in applicazioni come il riconoscimento facciale e la classificazione di oggetti.

L'etimologia del termine "convoluzione" si riferisce al processo matematico di combinazione di un'immagine con un *kernel* per estrarre caratteristiche rilevanti [44].

La CNN è un tipo di rete neurale *feedforward*, la quale possiede la capacità di estrarre caratteristiche dai dati utilizzando strutture di convoluzione. A differenza dei metodi tradizionali di *feature extraction*, l'architettura della CNN è ispirata alla percezione visiva [45]:

- un neurone biologico corrisponde ad un neurone artificiale;
- i *kernel* della CNN rappresentano diversi recettori capaci di rispondere a varie caratteristiche;
- le funzioni di attivazione simulano la funzione per cui solo i segnali elettrici neurali che superano una certa soglia possono essere trasmessi al neurone successivo;
- le funzioni di perdita e gli ottimizzatori sono, invece, qualcosa inventato dall'uomo per insegnare all'intero sistema CNN ciò che ci aspettiamo che impari.

Una CNN è composta da uno strato di *input* e uno strato di *output*, oltre a diversi strati nascosti. Solitamente, questi *layers* appartengono a tre categorie principali: convoluzione (CONV), pooling (POOL) e *fully-connected* (FC), ovvero completamente connessi [46].

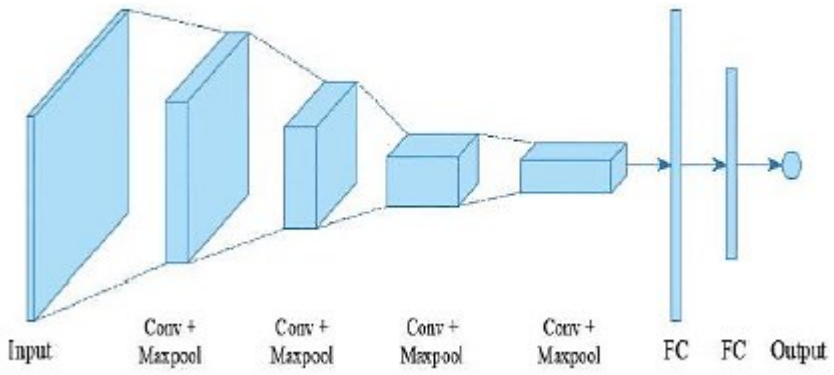

***Figura 20**. Struttura base di una CNN* [46]

Oltre a strati di convoluzione, strati di pooling e strati *fully-connected*, è possibile individuare anche ulteriori strutture, come funzioni di attivazione non lineare e una funzione di attivazione dell'ultimo livello.

Strato di convoluzione

Gli strati di convoluzione, noti anche come strati convoluzionali, sono una parte fondamentale delle CNN. Essi sono progettati per estrarre caratteristiche significative dalle immagini, riducendo la necessità di estrarre manualmente queste caratteristiche.

In uno strato di convoluzione, vengono utilizzati filtri, detti *kernel*, i quali possiedono dimensioni ridotte (ad esempio 3x3), ma si estendono lungo tutta la profondità dell'input [39].

Quando i dati raggiungono uno strato convoluzionale, lo strato convolve ciascun filtro lungo la dimensione spaziale dell'input per produrre una mappa di attivazione bidimensionale. Man mano che si scorre lungo l'input viene calcolato il prodotto scalare per ciascun valore in quel *kernel*. Da questo, la rete imparerà dei kernel che 'si attivano' quando individuano una specifica caratteristica in una posizione spaziale data dell'input. Queste sono comunemente note come attivazioni.

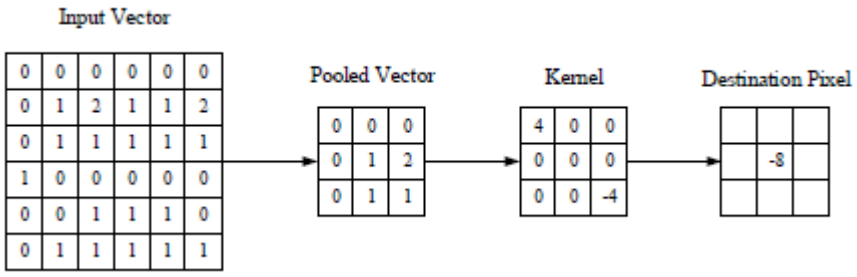

***Figura 21**. Rappresentazione visiva di uno strato di convoluzione. L'elemento centrale del kernel viene posizionato sull'input vettoriale, e quindi viene calcolato e sostituito con una somma pesata di se stesso e dei pixel circostanti.* [39]

Ogni *kernel* avrà una mappa di attivazione corrispondente, che verrà impilata lungo la dimensione della profondità per formare il volume di output completo dello strato convoluzionale. Questo significa che in un singolo strato convoluzionale, ci sono molte *feature map*, ognuna delle quali è specializzata nella rilevazione di una caratteristica specifica, come bordi, angoli o texture.

Le *feature map*, conosciute appunto anche come mappe delle caratteristiche o mappe di attivazione, rappresentano l'output di un filtro specifico applicato all'input. Sono visualizzate come immagini in scala di grigi, dove i pixel più chiari indicano una forte attivazione del filtro su una specifica caratteristica, mentre i pixel più scuri rappresentano aree meno rilevanti.

Gli strati convoluzionali sono anche in grado di ridurre significativamente la complessità del modello attraverso l'ottimizzazione del suo *output*. Questa ottimizzazione avviene mediante tre iperparametri: la profondità, lo stride e l'impostazione dello *zero-padding* [39].

La profondità in uno strato di convoluzione si riferisce al numero di filtri o *kernel* applicati durante l'operazione di convoluzione. Ogni filtro estrae un diverso insieme di caratteristiche dall'immagine di input. Un numero maggiore di filtri implica la capacità di estrarre una gamma più ampia di caratteristiche, ma possono anche aumentare il costo computazionale.

Lo *stride* è un parametro che controlla quanto lontano il filtro si sposta ogni volta che esegue un'operazione di convoluzione sull'immagine. Un valore di stride maggiore riduce la dimensione dell'output e può ridurre il costo computazionale. Uno stride più piccolo conserva più dettagli spaziali, ma può richiedere più risorse computazionali.

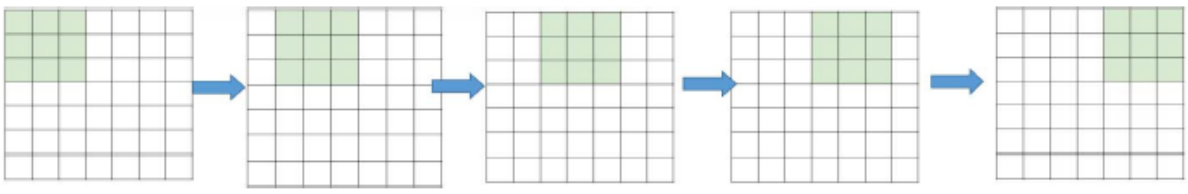

***Figura 22**. Esempio di stride pari a 1; in questo caso, la finestra del filtro si sposta solo una volta per ciascuna connessione* [44]

Il *zero-padding* è una tecnica utilizzata per gestire la dimensione dell'output dopo l'operazione di convoluzione. Aggiungendo zeri intorno ai bordi dell'immagine di input prima di eseguire la convoluzione, è possibile controllare la dimensione dell'output. Spesso viene utilizzato per preservare le dimensioni dell'immagine e per evitare la perdita di informazioni spaziali durante le convoluzioni.

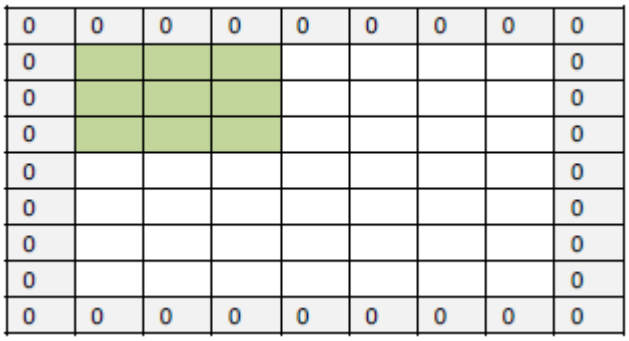

Figura 23. *Esempio di zero-padding* [44]

Per calcolare allora la dimensionalità spaziale dell'output degli strati convoluzionali è possibile applicare la seguente formula:

$$\frac{(V - R) + 2Z}{S + 1}$$

Dove V rappresenta la dimensione del volume di input (altezza x larghezza x profondità), R rappresenta la dimensione del campo recettivo, ovvero l'area dell'*input* che influisce sull'attivazione del filtro; Z rappresenta la quantità di *zero-padding* impostata e S si riferisce allo *stride* [39].

Strato di pooling: maxPooling

Oltre ai *layers* di convoluzione, sono presenti anche alcuni strati di *pooling*; spesso essi sono posizionati subito dopo uno strato di convoluzione e ciò suggerisce che gli *output* della convoluzione rappresentino gli *input* degli strati di pooling della rete [46].

L'obiettivo delle operazioni di *pooling* è quello di ridurre le dimensioni delle *feature map*, utilizzando alcune funzioni che permettano di riassumere le sotto-regioni, considerando il valore medio o massimo. Questi strati hanno allora lo scopo di ridurre gradualmente la dimensionalità dei dati, diminuendo il numero di parametri e la complessità della procedura del modello, controllando così il problema dell'*overfitting*.

Alcune delle comuni operazioni di pooling sono il *max pooling*, l'*average pooling*, lo *stochastic pooling* e lo *spectral pooling* [46].

Nella maggior parte delle CNN viene utilizzata l'operazione di *max pooling*, con *kernel* di dimensione 2 x 2 e con uno *stride* di 2 sulle dimensioni spaziali dell'input, riducendo la

dimensione della mappa di attivazione al 25% della dimensione originale, mantenendo comunque la profondità invariata [44].

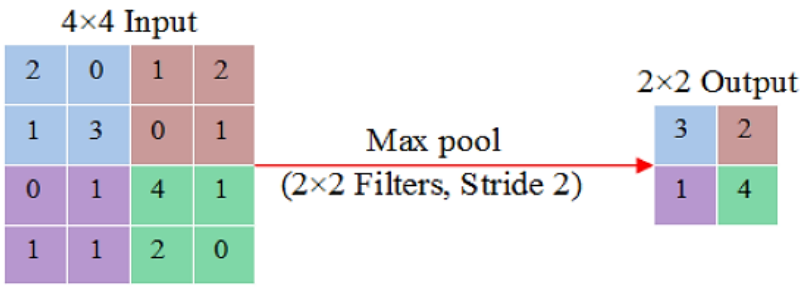

***Figura 24**. Esempio di operazione di max pooling* [46]

Funzione di attivazione non lineare

Successivamente alla convoluzione vengono applicate anche delle funzioni di attivazione non lineare. Esse hanno lo scopo di regolare o troncare l'uscita generata [44].

Le funzioni di attivazione comunemente utilizzate sono solitamente il sigmoide, la tangente iperbolica (tanh) e ReLU (*Rectified Linear Unit*).

Rispetto a diverse funzioni, le Unità Lineari Rettificate (ReLU) [46] sono preferite in quanto sono in grado di addestrare reti neurali molto più velocemente.

Questa tipologia di funzione di attivazione non lineare presenta definizioni più semplici sia per la funzione stessa, che per il gradiente. Esse possono essere formulate come [44]:

$$ReLU(x) = max(0, x)$$

$$\frac{d}{dx} ReLU(X) = \begin{cases} 1 \; se \; x > 0 \\ 0 \; altrimenti \end{cases}$$

Strato completamente connesso

Gli strati completamente connessi, noti anche come strati densamente connessi (*fully connected layers*) o strati FC, rappresentano una parte cruciale nelle reti neurali convoluzionali (CNN) e in altre reti neurali profonde [44].

Il loro obiettivo è quello di compiere operazioni di classificazione, producendo una previsione finale o una decisione basata sulle caratteristiche estratte nelle fasi precedenti della rete.

Come suggerito dal nome, ed anche come è possibile osservare anche nella Figura 25, negli strati completamente connessi ogni nodo è collegato in modo diretto ad ogni nodo sia nel livello precedente che in quello successivo.

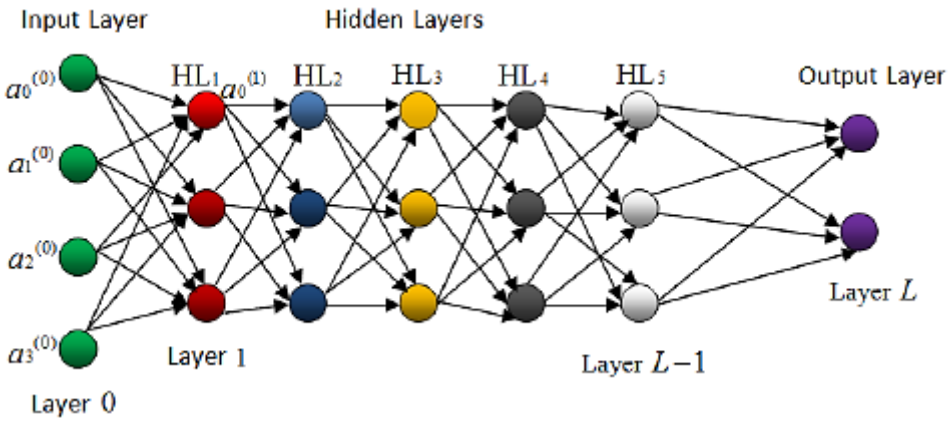

Figura 25. *Esempio di strato completamente connesso* [46]

Dopo l'estrazione di caratteristiche attraverso strati di convoluzione e pooling, gli strati completamente connessi effettuano un'operazione di *flattening*, ovvero appiattiscono l'*output* in un vettore unidimensionale e lo collegano a unità neurali completamente connesse. Queste unità ricevono *input* da tutte le *feature* estratte, e attraverso il processo di apprendimento dei pesi, la CNN regola i pesi per minimizzare l'errore tra le previsioni e i valori desiderati.

Questo processo di adattamento dei pesi permette alla rete di apprendere da dati di addestramento e di generalizzare le conoscenze acquisite per effettuare decisioni finali. Gli strati completamente connessi possono essere impiegati per la classificazione o la regressione, a seconda degli obiettivi, fornendo previsioni di classi o valori numerici. Infatti, l'ultimo livello completamente connesso solitamente presenta lo stesso numero di nodi di uscita del numero di classi.

Funzione di attivazione dell'ultimo livello

Solitamente la funzione di attivazione utilizzata nell'ultimo strato completamente connesso di una rete neurale è diversa da quelle utilizzate negli strati precedenti. La scelta di questa funzione dipende dal compito specifico che la rete deve svolgere. Ad esempio, nel caso di un compito di classificazione multiclasse, viene spesso utilizzata una funzione chiamata "Softmax" [46].

Questa funzione assegna a ciascun pixel un valore di probabilità per ciascuna classe, producendo N canali di probabilità, dove N è il numero di classi. La classe con la probabilità più alta assegnata a ciascun pixel corrisponde alla prevista segmentazione o classificazione.

CNN UTILIZZATE

Nel paragrafo corrente, si procederà con la presentazione delle CNN impiegate nell'ambito della sperimentazione condotta. Le CNN rappresentano una componente fondamentale del nostro approccio di ricerca e svolgono un ruolo cruciale nell'analisi e nell'elaborazione dei dati. Saranno illustrate le caratteristiche architetturali delle reti utilizzate, al fine di fornire una comprensione dettagliata dei metodi implementati nella fase sperimentale del nostro lavoro. L'analisi delle CNN riveste, quindi, un'importanza strategica per la comprensione dei risultati ottenuti e per valutare l'efficacia del nostro approccio.

AlexNet

La CNN AlexNet è stata descritta per la prima volta nel 2012 da Alex Krizhevsky e Ilya Sutskever, con la collaborazione di Geoffrey Hinton, uno dei principali pionieri nel campo del *Deep Learning* [47].

Questa architettura è stata introdotta come partecipante al concorso *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) ed ha rappresentato una svolta epocale nell'ambito del riconoscimento delle immagini.

Infatti, nel contesto dell'ILSVRC 2012, AlexNet ha ottenuto il primo posto, riducendo significativamente l'errore di classificazione rispetto alle reti neurali profonde precedentemente sviluppate. Questa vittoria ha segnato un punto di svolta nella ricerca sulle reti neurali profonde, dimostrando il loro enorme potenziale per il riconoscimento delle immagini su larga scala.

Una tra le innovazioni chiave di AlexNet è l'uso di tecniche di *data augmentation*, le quali hanno consentito al modello di apprendere da un set di dati relativamente piccolo riducendo i problemi di *overfitting*. Questa tecnica ha contribuito a migliorare notevolmente la generalizzazione del modello.

La rete è formata in totale da otto strati principali: cinque strati di convluzione e tre *layers* completamente connessi [48].

Analizzando più nel dettagli gli strati convolutivi:

1. Il primo strato è un livello di convoluzione con 96 filtri 11x11x3. Questo strato esegue la convoluzione iniziale dell'immagine di input a colori (RGB).
2. Il secondo strato è un livello di convoluzione con 256 filtri 5x5x48, che segue il primo strato convoluzionale.

3. Il terzo strato è un livello di convoluzione con 384 filtri 3x3x256.
4. Il quarto e il quinto strato convolutivo presentano rispettivamente 384 e 256 filtri da 3x3x192. Questi strati hanno l'obiettivo di estrarre *features* complesse dalle immagini.

Dopo ogni strato convoluzionale, segue uno strato di *MaxPooling*, il quale permette di ridurre la dimensione spaziale dell'immagine estraendo le feature più rilevanti.

Dopo i cinque strati convolutivi e i tre strati di pooling, ci sono tre strati completamente connessi. Il primo strato completamente connesso contiene 4096 neuroni, seguito da un secondo strato con la stessa dimensione. L'ultimo strato completamente connesso ha 1000 neuroni, che corrispondono alle classi dell'ILSVRC. Questo è l'output della rete, che determina la classe di oggetti riconosciuti nell'immagine.

Tra i vari strati, viene anche utilizzata una funzione di attivazione ReLU, la quale contribuisce a mitigare il problema del gradiente scomparso e accelera l'addestramento.

L'architettura della rete è illustrata in Figura 26.

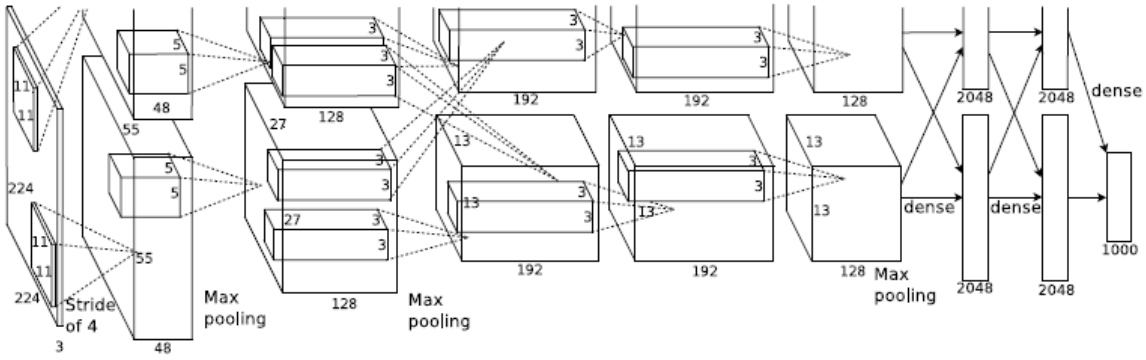

***Figura 26**. Architettura di AlexNet*. [48]

ResNet50

ResNet50 appartiene alla famiglia delle Reti Neurali Residuali (ResNet). La caratteristica distintiva è l'uso dei cosiddetti "blocchi residuali" per combattere il problema del degradamento del gradiente nelle reti neurali profonde.

ResNet50 rappresenta un altro importante traguardo nell'evoluzione delle architetture di reti neurali profonde, ed è stato sviluppato da Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun del Microsoft Research Asia. Questa innovativa architettura è emersa nel 2015, in un

contesto in cui la comunità scientifica cercava soluzioni per superare i problemi legati all'addestramento di reti neurali molto profonde.

L'anno 2015 è stato caratterizzato da significativi avanzamenti nell'ambito dell'elaborazione delle immagini, in particolare con il riconoscimento di oggetti e la segmentazione delle immagini. In questo contesto, ResNet50 ha portato un contributo fondamentale. La sua architettura è caratterizzata da una profondità eccezionale, rispetto alle architetture più tradizionali con meno strati.

ResNet50 è una variante del modello ResNet originario ed include 48 strati di convoluzione insieme a un livello di *MaxPooling* e un livello di *Average Pooling* [49].

Analizzando maggiormente la struttura della rete possiamo osservare come l'input di ResNet50 è un'immagine a colori con dimensioni tipiche di 224x224 pixel e tre canali (RGB).

Gli strati sono organizzati in blocchi residuali che permettono il passaggio delle informazioni direttamente da uno strato all'altro. Ciascun blocco residuale è composto da diversi strati convoluzionali, normalizzazione del batch e funzioni di attivazione ReLU. La connessione residuale rappresenta il cuore di questi blocchi. L'output di un blocco residuale, infatti, viene sommato all'input originale del blocco, permettendo al gradiente di propagarsi senza degradarsi. Questo rende possibile l'addestramento di reti neurali molto profonde.

Dopo ciascun blocco residuale, seguono strati di pooling (generalmente *MaxPooling*) per ridurre la dimensione spaziale dell'immagine estratta dai blocchi.

Alla fine della rete, ci sono uno o più strati completamente connessi che convergono nell'output della rete. In ResNet50, l'output è tipicamente un *layer* con 1000 neuroni, corrispondenti alle classi del dataset ImageNet, in cui ResNet50 è spesso addestrato.

Dopo i *layer* convoluzionali, al fine di garantire che l'output non diventi troppo grande o piccolo, viene applicata la normalizzazione del batch, andando così a facilitare l'addestramento.

L'architettura della rete è illustrata in Figura 27.

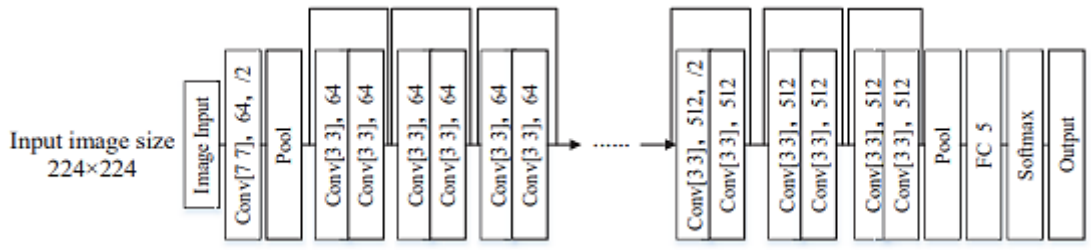

***Figura 27**. Architettura di ResNet.* [50]

VGG-16

La rete neurale convoluzionale VGG16, sviluppata da NK. Simonyan e A. Zisserman dell'Università di Oxford, è stata presentata per la prima volta nel documento intitolato "*Very Deep Convolutional Networks for Large-Scale Image Recognition*" [51].

Questo modello ha dimostrato un'elevata capacità di classificazione, raggiungendo un'accuratezza del 92,77% sui dati di test utilizzando il complesso dataset ImageNet.

Le immagini in *input*, di dimensioni 224x224x3, attraversano in sequenza 2 strati di convoluzione, seguiti da un livello di *MaxPooling*, e successivamente da ulteriori 2 strati di convoluzione e un secondo livello di *MaxPooling*.

L'architettura prosegue con l'implementazione di 3 strati di convoluzione, 1 strato di *Max Pooling*, altri 3 strati di convoluzione. Dopo di esso è presente un ulteriore strato di *Max Pooling*. L'intero processo è seguito da strati completamente connessi e da strati di attivazione ReLU.

Il numero di filtri varia al variare dei livelli. Nei livelli di convoluzione, viene utilizzato un filtro di dimensioni 3x3 con uno *stride* di 1, mentre nel livello di *Max Pooling*, il filtro ha dimensioni di 2x2 con uno *stride* di 2.

L'architettura della rete è illustrata in Figura 28.

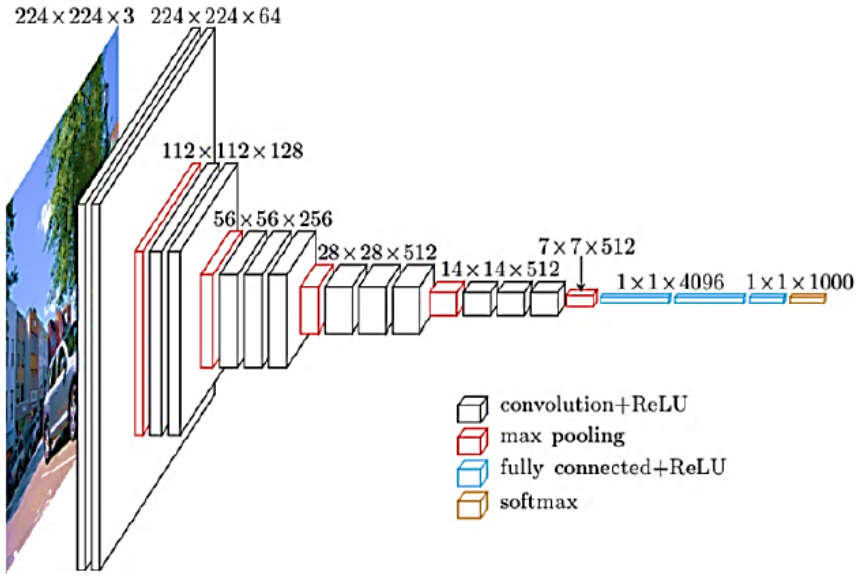

***Figura 28**. Architettura di VGG-16.* [52]

Inception-V3

La CNN Inception-V3 rappresenta un significativo contributo all'elaborazione delle immagini e al riconoscimento visivo. Questa architettura di *deep learning* è stata sviluppata da un team di ricercatori di Google, tra cui Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens e Zbigniew Wojna [53]. La sua introduzione nel 2015 ha segnato un punto di svolta nell'ambito del riconoscimento di oggetti e ha avuto un impatto notevole sulla comunità scientifica.

L'architettura Inception-V3 si distingue per l'utilizzo di moduli *Inception*, i quali vengono considerati una soluzione all'approccio tradizionale di convoluzione. Questi moduli sono in grado di catturare *features* da diverse scale spaziali all'interno di un'immagine. Ogni modulo *Inception* combina al suo interno convoluzioni di diverse dimensioni (1x1, 3x3, 5x5) e pooling al fine di estrarre informazioni a diverse scale. Questo approccio consente di acquisire dettagli fini e *features* di portata più ampia contemporaneamente.

Inception-V3 inizia con una sequenza di strati di convoluzione e pooling iniziali, finalizzati all'estrazione delle *features* di base. Questi strati riducono progressivamente le dimensioni dell'immagine.

I moduli *Inception* sono disposti in serie per estrarre features complesse da diverse scale spaziali. Ciascuno di essi impiega diverse convoluzioni di dimensioni differenti, permettendo all'architettura di catturare un'ampia gamma di informazioni dall'immagine.

Dopo di essi, Inception-V3 adotta uno strato di *pooling* globale che riduce la dimensione spaziale delle *features* in un vettore. Tale vettore è quindi sottoposto a uno o più strati completamente connessi per la classificazione finale.

Anche questa rete utilizza tecniche di normalizzazione del *batch* per migliorare la stabilità e l'addestramento. Inoltre, anche in questa rete vengono introdotte le funzioni di attivazione ReLU per introdurre la non linearità.

L’architettura della rete è illustrata in Figura 29.

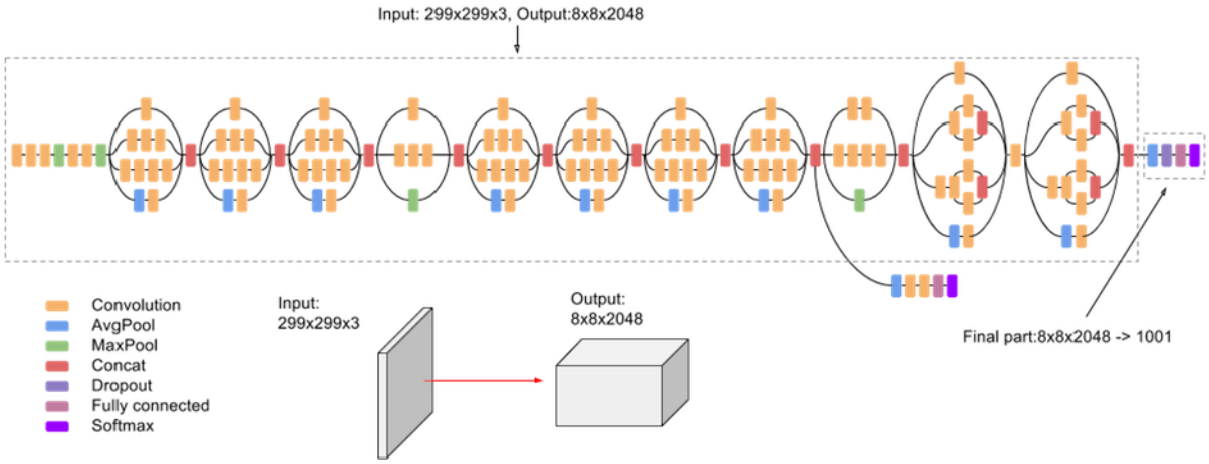

***Figura 29**. Architettura di Inception-V3.* [53]

Fast R-CNN

Fast R-CNN rappresenta un importante progresso nell'ambito del riconoscimento di oggetti e nella localizzazione degli stessi in immagini. Questa architettura è stata sviluppata da Ross Girshick [54], noto per i suoi contributi pionieristici nel campo delle reti neurali convoluzionali e del riconoscimento di oggetti. Questa tipologia di rete è stata introdotta nel 2015 come un notevole miglioramento rispetto agli approcci precedenti nel contesto dell'*object detection*, il quale consiste nell'identificare e localizzare oggetti specifici all'interno di immagini.

Nella fase di sviluppo dell'architettura, l'obiettivo principale era quello di aumentare l'efficienza e l'accuratezza del processo di *object detection*, superando le limitazioni di metodi precedenti.

Questa tecnica combina le reti neurali convoluzionali pre-addestrate con un algoritmo di *region proposal network* (RPN) per identificare oggetti di interesse in un'immagine. Il processo di funzionamento di Fast R-CNN può essere suddiviso in diverse fasi.

Innanzitutto, Fast R-CNN riceve un'immagine in input e la passa attraverso una CNN pre-addestrata (ad es. VGG-16 o Inception-V3) per estrarre le caratteristiche. L'utilizzo di una CNN pre-addestrata consente di sfruttare le conoscenze acquisite da modelli di grandi dimensioni su un ampio set di dati, migliorando l'accuratezza del rilevamento. Questa CNN convoluzionale converte l'immagine in un insieme di *feature maps*, ciascuna delle quali cattura dettagli di diversi livelli di astrazione.

Successivamente, Fast R-CNN utilizza il RPN per generare proposte di regioni candidate all'interno dell'immagine, basate su queste *feature maps*. Queste regioni candidate sono selezionate in modo che abbiano una probabilità significativa di contenere oggetti di interesse.

Il RPN è allenato per predire queste regioni utilizzando una combinazione di posizioni di ancore e punteggi di confidenza.

Una volta ottenute le regioni proposte, Fast R-CNN utilizza un *layer* di *RoI pooling* per adattare le dimensioni delle regioni proposte alle dimensioni fisse richieste dalla rete neurale *fully connected* successiva. Queste regioni adattate vengono quindi passate attraverso una serie di strati *fully connected* e *softmax* per classificare e raffinare la posizione degli oggetti all'interno di ciascuna regione.

Infine, Fast R-CNN fornisce le posizioni e le classificazioni degli oggetti all'interno dell'immagine, consentendo così di individuare e classificare gli oggetti di interesse in modo efficiente.

La metodologia di sviluppo di Fast R-CNN e l'integrazione del RPN hanno segnato un importante passo avanti nel campo del riconoscimento di oggetti, riducendo la complessità computazionale e migliorando la precisione del processo di *detection*.

L'architettura della rete è illustrata in Figura 30.

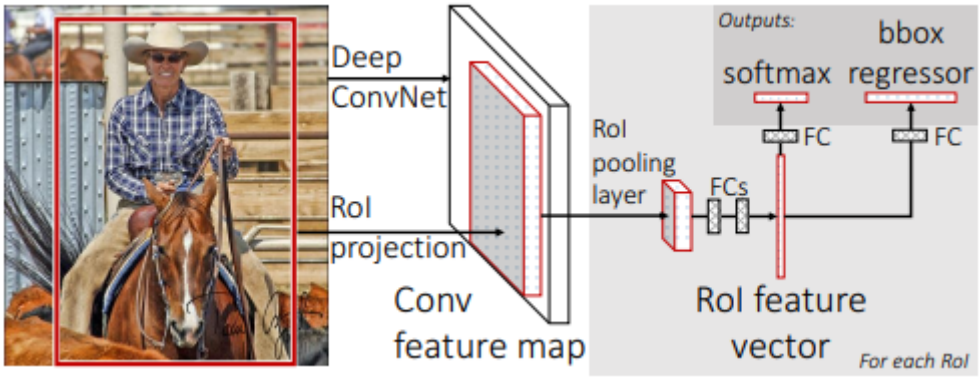

***Figura 30**. Architettura di Fast R-CNN.* [54]

UNITY (GAME ENGINE)

Unity è un motore di gioco multi-piattaforma molto popolare utilizzato per lo sviluppo di videogiochi e simulazioni interattive in 2D e in 3D, esperienze in realtà virtuale (VR), realtà aumentata (AR) e altri contenuti interattivi. È stato creato da Unity Technologies ed è ampiamente utilizzato sia da sviluppatori indipendenti che da grandi aziende nell'industria dei giochi e in altre industrie correlate. Consente l'esportazione per Android ed iOS, oltre che per le principali console in commercio [55].

Per implementare la logica del gioco, Unity utilizza il linguaggio di programmazione C#; inoltre, offre un potente editor visuale che consente agli sviluppatori di creare, modificare e organizzare facilmente gli elementi del gioco, come personaggi, ambienti, luci e oggetti, senza dover scrivere molto codice.

Interfaccia di Unity

L'interfaccia, con cui solitamente si presenta Unity, al suo avvio è composta da vari elementi [56]. Essi sono:

A. *Hierarchy*
B. *Scene*
C. *Game*
D. *Inspector*
E. *Toolbar*
F. *Console*
G. *Project*

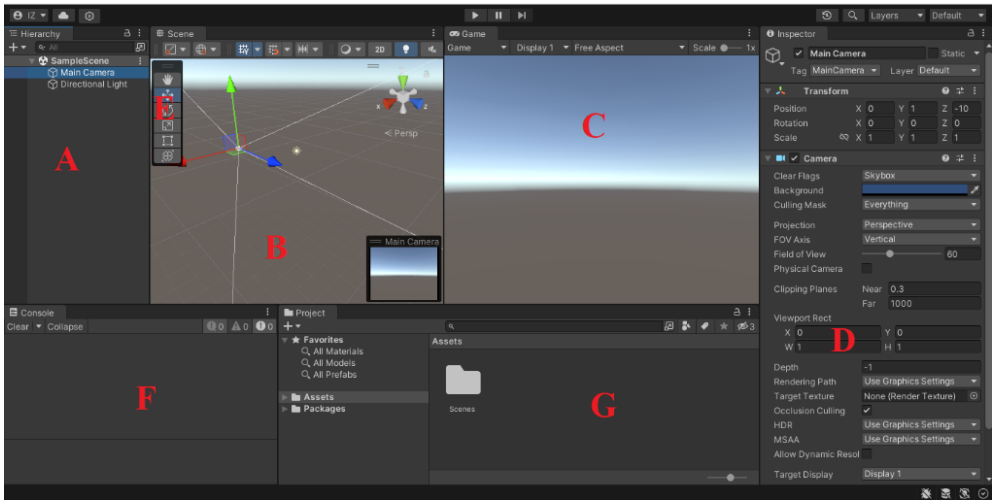

***Figura 31**. Layout di Unity*

La *Hierarchy* rappresenta la gerarchia degli oggetti presenti nella scena corrente. Al suo interno saranno visibili, dunque, tutti gli oggetti su cui sarà possibile lavorare. Gli oggetti presenti all'interno di una scena sono denominati *GameObjects* e verranno sempre elencati in questa finestra.

La *Scene* corrisponde alla finestra di lavoro in cui compare la scena attuale; in essa verranno posizionati, scalati e spostati gli oggetti.

All'interno della finestra *Game* verrà visualizzato il gioco come appare nella realtà, quando esso viene eseguito. Corrisponde ad una schermata di solo *output*, con la quale non sarà possibile interagire direttamente, ma sarà utilizzata per vedere il risultato del lavoro svolto.

L'*Inspector* è la finestra che permette di ispezionare l'oggetto selezionato. Verranno visualizzate tutte le informazioni riguardati il *GameObject* selezionato, mostrando gli scripts ad esso collegato e tutte le caratteristiche che possiede.

La *Toolbar* contiene alcune scorciatoie della *Transform* contenuta all'interno dell'*Inspector*, ovvero contiene le possibilità di ruotare, traslare o ridimensionare la componente della *Transform* associata ad uno specifico *GameObject*.

Tramite la *Console*, Unity permette di visualizzare eventuali errori o messaggi di debug.

Infine, mediante la finestra *Project* è possibile visualizzare tutte le cartelle contenenti i componenti presenti nell'applicazione. Tutto ciò che è presente, sarà copiato nella cartella del gioco e tutto ciò che si troverà nella cartella *Asset* sarà esportato per essere installato nei dispositivi dell'utente finale.

All'atto della creazione di una scena, come quella creata di default all'avvio di un nuovo progetto, denominata *SampleScene*, saranno presenti due *GameObject* fondamentali nella scena stessa, ovvero una *MainCamera* e una *DirectionalLight*. La telecamera corrisponde al punto di vista, ciò che cattura viene visualizzato nella finestra *Game*, mentre la luce corrisponde al sole, da cui dipendono ombre e illuminazioni visualizzate.

Metodi di Unity

Unity permette di associare uno o più script ad ogni *GameObject* presente in scena. Essi possono essere creati a partire dalla sezione *Project*, creando un file con estensione .CS (nel caso di C#) e successivamente editandolo con un editor come *Visual Studio*. Per assegnare lo script al *GameObject* d'interesse è possibile selezionare "*Add Component*" all'interno dell'*Inspector* del *GameObject* stesso.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class FirstScript : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
```

***Figura 32**. Struttura base di uno script .CS implementato in Unity*

Gli script creati di default da Unity implementano una classe derivante dalla classe integrata *Monobehaviour*. Inoltre, alla creazione dello script vengono inseriti due metodi, che sono i più rilevanti definiti all'interno della classe:

- *Start():* è un metodo che viene eseguito automaticamente al primo fotogramma della scena in cui si trova questo script e viene eseguito una sola volta. Al suo interno consente di inizializzare variabili, caricare un materiale ed eseguire tutto ciò che occorre sia fatto nel preciso istante dello *start* ed una sola volta.

- *Update():* questo metodo viene eseguito ad ogni fotogramma della scena. Finché la scena sarà in esecuzione, esso verrà ripetuto ininterrottamente decine di volte al secondo. Infatti, essendo aggiornato ad ogni frame, se il gioco lavora a 30fps, questo metodo verrà aggiornato 30 volte al secondo. Essendo poi il *frame rate* quasi mai costante, anche l'aggiornamento di questo metodo non sarà mai costante. In esso verranno implementati i movimenti degli oggetti, l'intercettazione dell'input del giocatore, ecc

Unity presenta poi altri metodi di default come *Awake(), OnEnable(), OnDisable(), OnCollisionEnter(), OnTriggerEnter()*.

MATLAB

Matlab, acronimo di "Matrix Laboratory," è un ambiente di calcolo numerico e linguaggio di programmazione ampliamente utilizzato in una vasta gamma di discipline scientifiche, ingegneristiche e industriali. Creato da MathWorks, Matlab è noto per la sua versatilità e il suo potente set di strumenti che consentono agli utenti di eseguire analisi complesse, elaborazioni dei dati e sviluppo di algoritmi avanzati [57].

Una delle caratteristiche distintive è la capacità di manipolare matrici e vettori in modo efficiente, il che lo rende ideale per risolvere problemi che coinvolgono dati multidimensionali e computazioni matematiche intensive. Grazie alla sua sintassi intuitiva e alla vasta gamma di funzioni integrate, Matlab è accessibile sia ai principianti che agli esperti, rendendo più agevole il processo di prototipazione e sviluppo di soluzioni avanzate.

Negli ultimi decenni, è diventato uno strumento essenziale per la ricerca accademica, l'innovazione industriale e lo sviluppo di prodotti avanzati. Le sue applicazioni spaziano dalla simulazione di sistemi complessi all'analisi dei dati in tempo reale, dal controllo automatico alla progettazione di algoritmi di intelligenza artificiale.

Inoltre, Matlab si integra perfettamente con altre lingue di programmazione come C, C++, Python e Java, consentendo agli sviluppatori di sfruttare le librerie esistenti e incorporare facilmente codice personalizzato nei loro progetti. Questa interoperabilità lo rende una scelta ideale per l'integrazione di soluzioni in applicazioni più ampie [57].

Offre un ricco ecosistema di toolboxes specializzati che coprono una vasta gamma di discipline, tra cui l'elaborazione del segnale, l'analisi statistica, il machine learning, la simulazione numerica, la grafica e molto altro [58].

Questi toolboxes estendono le capacità di base di Matlab, consentendo agli utenti di affrontare sfide specifiche del settore e risolvere problemi complessi in modo più efficiente.

Deep Network Designer

Il *Deep Network Designer* rappresenta un importante strumento messo a disposizione degli utenti, che permette di progettare, addestrare e sperimentare reti neurali profonde, semplificando notevolmente il processo di sviluppo di modelli di intelligenza artificiale, mediante un’interfaccia *user-friendly* [59].

Una delle sue principali funzionalità è la possibilità di progettare reti neurali personalizzate con una flessibilità straordinaria. Gli utenti possono creare topologie di rete più o meno complesse, andando così a definire gli strati di rete, configurare i parametri di addestramento e personalizzare la struttura del modello in modo rapido ed efficiente.

Inoltre, il toolbox *Deep Network Designer* semplifica notevolmente il processo di addestramento delle reti neurali. Gli utenti possono importare dati, suddividerli in set di allenamento, convalida e test, e configurare i parametri di addestramento in modo intuitivo [59].

Il toolbox offre anche una visualizzazione in tempo reale delle prestazioni del modello, consentendo agli utenti di monitorare e ottimizzare l'apprendimento.

Un'altra caratteristica di rilievo è la capacità di applicare approcci di *transfer learning*, utilizzando modelli pre-addestrati e personalizzandoli per specifiche applicazioni [59].

Questo approccio può notevolmente accelerare lo sviluppo di modelli di intelligenza artificiale in scenari in cui i dati di addestramento sono limitati.

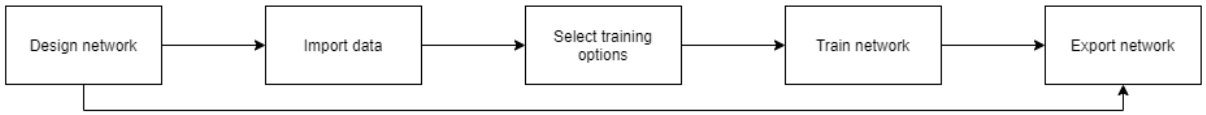

***Figura33**. Workflow* [60]

Il *toolbox* si compone di tre moduli principali:

- *Designer*: attraverso questo modulo gli utenti hanno la possibilità di personalizzare reti neurali profonde in modo semplice e rapido, definendo la topologia della rete, gli strati di connessione e i parametri specifici. Questa funzionalità è essenziale per esplorare diverse architetture di rete, adattandole alle esigenze specifiche. Inoltre, offre una visualizzazione in tempo reale della struttura della rete, semplificando l'ottimizzazione della configurazione [60].

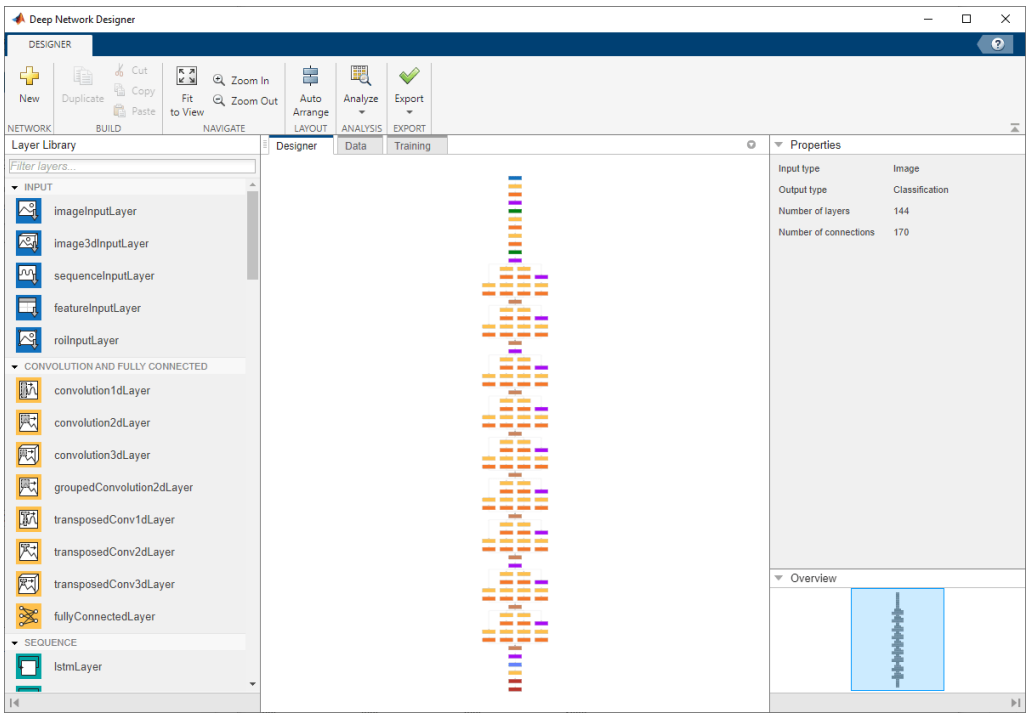

Figura 34. *Modulo "Designer" del toolbox Deep Network Designer* [60]

- *Data*: questo modulo semplifica notevolmente il processo di preparazione dei dati per l'addestramento delle reti neurali. Gli utenti possono facilmente importare dati da diverse fonti, suddividerli in set di allenamento e validazione, andando anche ad applicare operazioni di *data augmentation*. Questa funzionalità è essenziale per garantire che il modello di intelligenza artificiale abbia accesso a dati di alta qualità per l'addestramento, migliorando le prestazioni e la generalizzazione del modello [60].

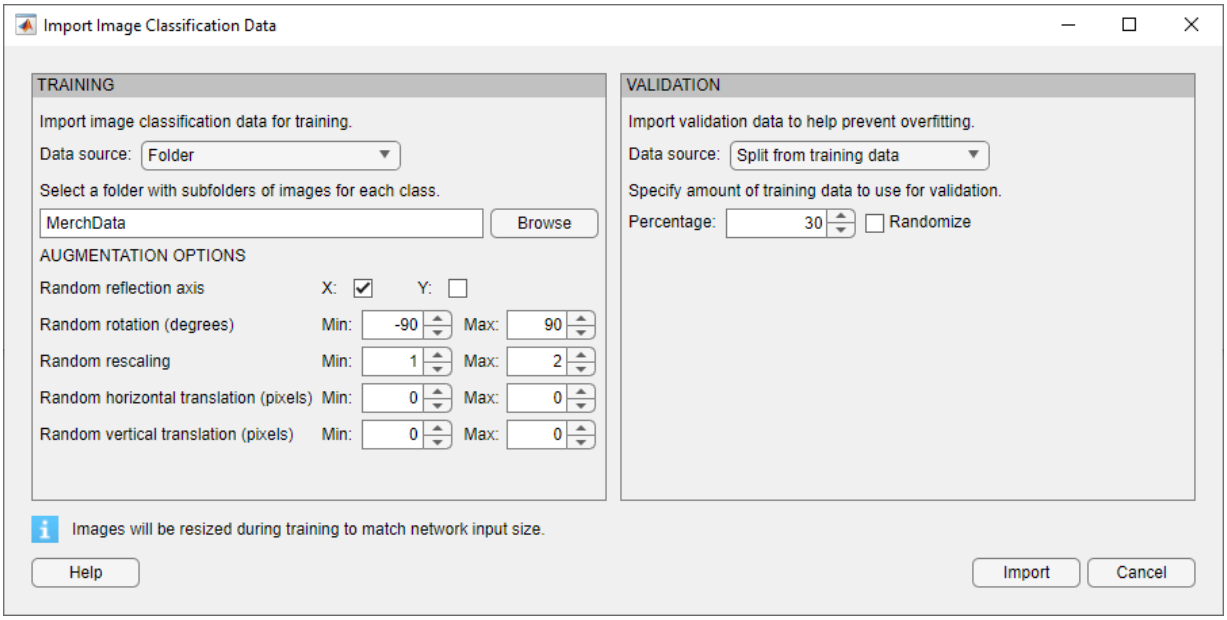

Figura 35. *"Import Image Classification Data" del toolbox Deep Network Designer* [60]

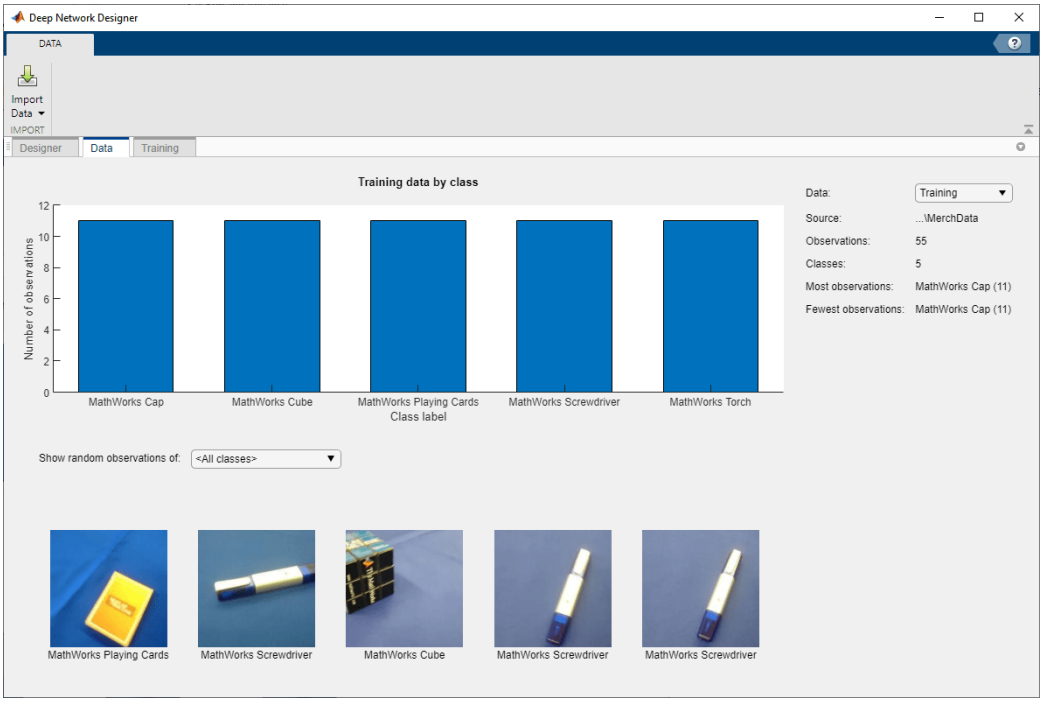

Figura 36. *Modulo "Data" del toolbox Deep Network Designer.* [60]

- *Training*: attraverso questa sezione è possibile configurare e gestire il processo di addestramento delle reti neurali in modo semplice ed efficace. Gli utenti possono definire parametri di addestramento, monitorare le metriche di prestazione, regolare l'apprendimento e valutare il modello in tempo reale. Questa funzionalità è cruciale per garantire che la rete raggiunga l'ottimizzazione e mantenga alte prestazioni. Inoltre, supporta il *transfer learning*, permettendo di utilizzare modelli pre-addestrati come punto di partenza, accelerando lo sviluppo di nuovi modelli personalizzati [60].

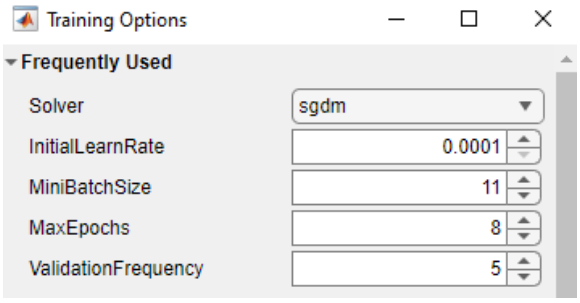

Figura 37. *"Training Options" del toolbox Deep Network Designer.* [60]

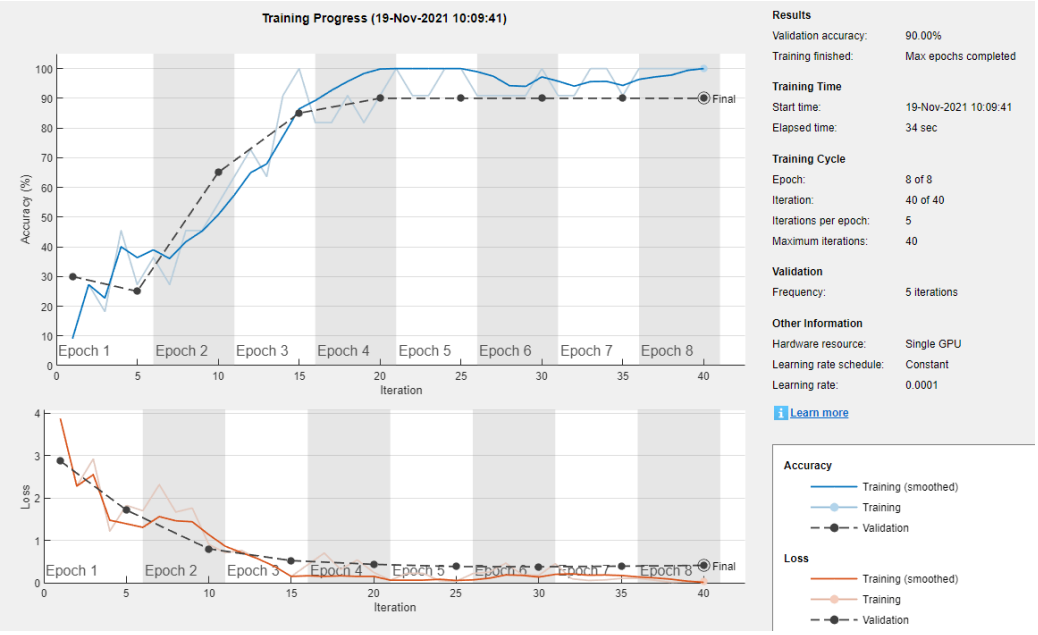

Figura 38. *Modulo "Training" del toolbox Deep Network Designer.* [60]

Image Labeler

Il toolbox *Image Labeler* è una risorsa essenziale per il campo della *computer vision* e dell'elaborazione delle immagini. Questa potente interfaccia fornisce agli utenti uno strumento completo per l'annotazione e l'etichettatura delle immagini, semplificando notevolmente il processo di preparazione dei dati per l'addestramento di modelli di intelligenza artificiale, come reti neurali convoluzionali (CNN) e algoritmi di *machine learning* [61].

Il *toolbox* permette agli utenti di importare facilmente immagini da varie fonti, comprese fotocamere, video e cartelle locali. Questo strumento risulta particolarmente utile in scenari in cui è necessario creare un set di dati etichettato per l'addestramento di algoritmi di riconoscimento, classificazione o segmentazione delle immagini [61].

Una delle sue principali funzionalità è la possibilità di annotare le immagini in modo efficiente. Gli utenti possono definire etichette personalizzate per oggetti, regioni d'interesse o categorie specifiche all'interno delle immagini. Questo è essenziale per insegnare ai modelli di intelligenza artificiale a riconoscere e comprendere il contenuto visivo.

L'interfaccia utente intuitiva del toolbox consente una facile navigazione tra le immagini e la gestione delle etichette, rendendo il processo di annotazione efficiente e preciso. Inoltre, *Image Labeler* offre un'ampia gamma di strumenti di disegno e di modifica per tracciare contorni, evidenziare regioni e definire i limiti degli oggetti di interesse. Questa funzionalità è

fondamentale per garantire che le etichette siano accurate e dettagliate, il che a sua volta migliora le prestazioni dei modelli di intelligenza artificiale [61].

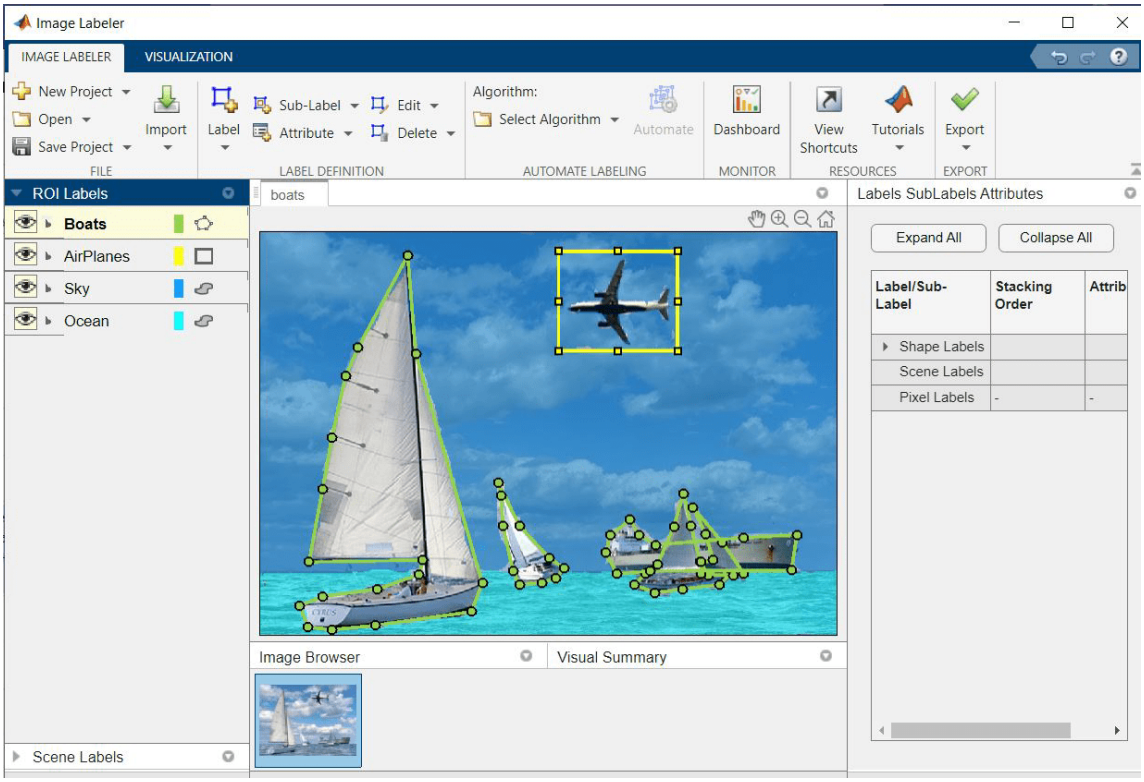

Figura 39. *Interfaccia del toolbox Image Labeler* [61]

Capitolo 3

Metodologia proposta

In questo capitolo verranno esposti i passaggi principali in cui questo progetto è stato articolato, andando a porre l'attenzione sui due grandi blocchi di cui si compone.

Nella prima parte del lavoro, infatti l'attenzione è stata posta sulla realizzazione del dataset. Tramite l'utilizzo della piattaforma Unity Engine, è stato realizzato un *tool* per poter acquisire delle istantanee contenenti un soggetto principale, sul quale sono stati applicati dei tatuaggi. La scelta dei tatuaggi in oggetto è stata fatta all'interno di uno dei campi di utilizzo della biometria nelle scienze forensi: il riconoscimento dei membri di gang. Infatti, le immagini selezionate sono riconducibili al simbolismo della mafia russa, al fine di limitare la variabilità dei soggetti raffigurati. Dopo aver terminato lo sviluppo del *tool*, le immagini sono state acquisite e memorizzate.

Nella seconda parte del lavoro, invece, ci si è concentrati nella realizzazione delle reti neurali, prendendo spunto dalla letteratura esistente, andando così ad applicare l'approccio del *transfer learning*, per poter modificare ed addestrare alcune CNN. L'obbiettivo di questa seconda parte è stato quello di validare il dataset realizzato.

Nello schema riportato in seguito, sono riportati inoltre i passaggi riassunti in precedenza.

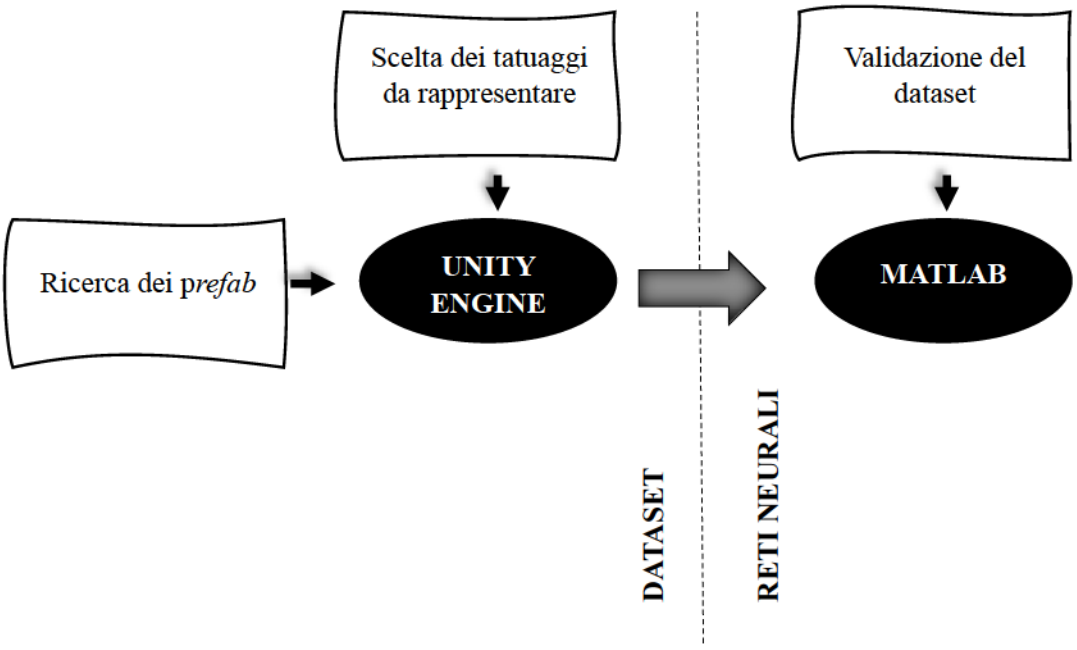

***Figura 40**. Schema dell'idea*

SVILUPPO DEL DATASET

Per poter creare un dataset di immagini contenenti soggetti tatuati è stato creato prima di tutto il *tool* mediante Unity Engine; in una fase successiva si è proceduto alla creazione del dataset vero e proprio mediante l'acquisizione delle immagini, come verrà illustrato nel Capitolo 4.

Sviluppo del *tool*

Il primo passo effettuato per la realizzazione del dataset, è stata la ricerca delle immagini. Inizialmente, sono stati valutati vari campi di applicazione del dataset, andando a selezionare poi immagini legate al simbolismo della mafia e della criminalità russa. Selezionando un campo di analisi ristretto, è possibile limitare la variabilità dei soggetti raffigurati. Infatti, coprire l'intera varietà di tatuaggi esistenti risulta pressoché impossibile. In questo ambito, allora, sono stati selezionate 10 diverse tipologie di tatuaggio, da poter rappresentare all'interno del dataset. Esse sono: cupole a cipolla, ragni, gatti, rose, coltelli, stelle, scarabei, sigle, croci e teschi.

La scelta dei tatuaggi si è svolta in parte mediante l'utilizzo del motore di ricerca Google, in parte attraverso la realizzazione di disegni originali. Le immagini sono state poi elaborate in Adobe Photoshop in modo tale da essere contenute in un quadrato di 12x12 cm, lasciando uno spazio vuoto adeguato attorno all'immagine, e successivamente salvate in formato .PNG.

Nella Figura 41 sono riportati alcuni esempi delle immagini utilizzate.

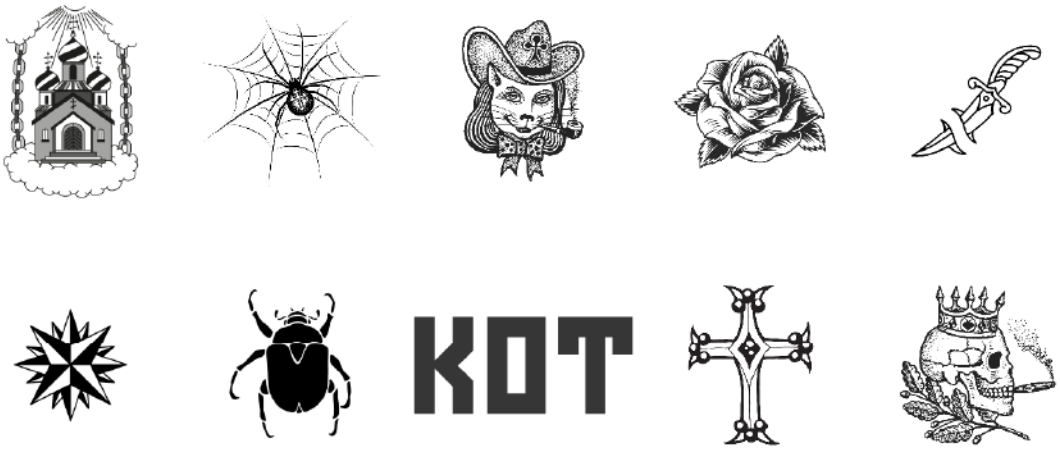

***Figura 41**. Esempi delle immagini utilizzate come tatuaggi.*

In totale sono state realizzate 144 immagini.

Dopo aver scelto il target dell'applicazione, sono stati ricercati gli elementi necessari per poter sviluppare quanto ipotizzato in Unity. Allora, è stata eseguita una ricerca all'interno dell'*Asset Store* di Unity ed in altri siti web dei principali *prefab* necessari. Sono stati selezionati un modello raffigurante una figura maschile ed un ambiente interno, che potesse agire come sfondo. Il criterio principale utilizzato in questa fase è il grado di realismo, in modo tale da ottenere poi un risultato finale quanto più simile alla realtà. Entrambi gli elementi necessari sono stati trovati nel sito web: https://www.cgtrader.com/free-3d-models.

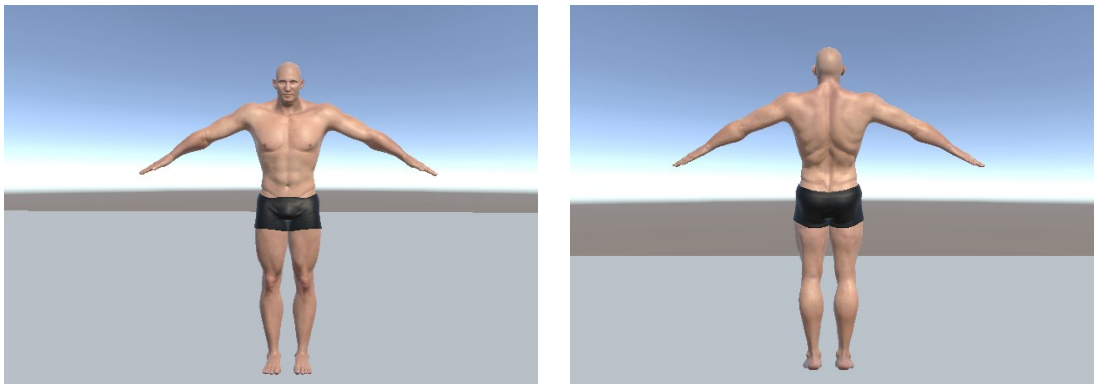

***Figura 42**. Prefab del soggetto scelto.*

***Figura 43**. Prefab dello sfondo scelto.*

All'interno della piattaforma Unity Engine, è stato creato un nuovo progetto mediante l'utilizzo dell'opzione 3D (URP), la quale permette al suo interno di implementare i *Decal projector*, ovvero i *GameObject* che hanno permesso di proiettare i tatuaggi sul corpo del soggetto.

Prima di poter iniziare con la realizzazione del *tool* è stato necessario aggiungere la *Renderer Feature* corrispondente ai *Decal*, all'interno del *URP-Performant-Renderer.asset*

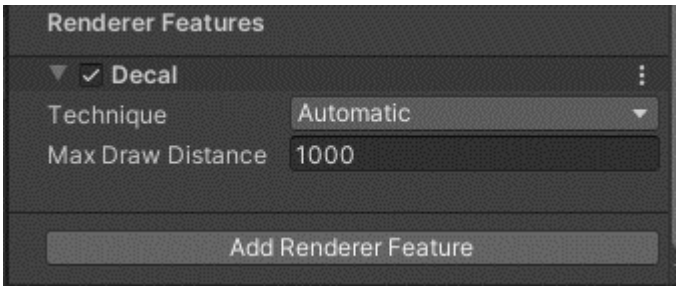

***Figura 44**. Aggiunta della Render Feature d'interesse.*

Questo ha permesso di aggiungere una nuova tipologia di *GameObject*, utilizzabile, denominata *URP Decal Projector*.

Il primo passaggio è stato quello di realizzazione della struttura della scena: sono stati importati i *prefab*, corrispondenti al soggetto e allo sfondo, e ne sono stati sistemati i rispettivi materiali.

Successivamente, sono stati posizionati i proiettori in corrispondenza delle zone del corpo su cui è stato deciso di posizionare i tatuaggi. Sono state scelte 11 zone corrispondenti a: petto, parte alta della schiena, parte bassa della schiena, fianco, braccio, avambraccio, polso, quadricipite, polpaccio, caviglia e piede. Per ciascuna di esse, è stato posizionato un *Decal* in modo tale che proiettasse perpendicolarmente l'immagine sulla superficie del corpo.

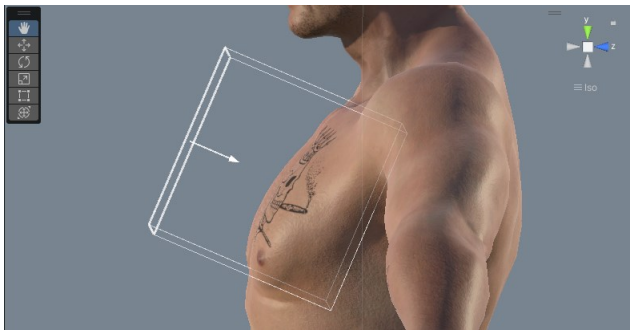

***Figura 45**. Esempio del posizionamento del Decal Projector.*

Per poter controllare i movimenti della *MainCamera.cs* sono stati utilizzati i seguenti script:

- *CameraZoom.cs*, permette di effettuare lo zoom sulla scena ruotando la rotellina del mouse;
- *CameraController.cs*, ha lo scopo di ruotare la vista attorno al soggetto, impostato come *Target* alla velocità di rotazione impostata in *Rotation Speed*;
- *CameraHeightController.cs,* consente di alzare o abbassare la quota mediante l'utilizzo delle frecce, o in maniera analoga dei tasti W e S, rispettivamente. Riceve come parametri di ingresso la velocità *Move Speed,* la quota minima (*Min Height*) e quella massima (*Max Height*);

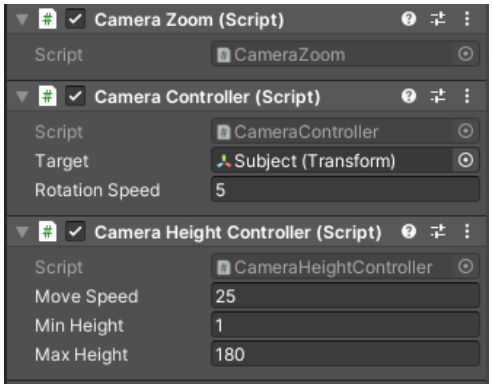

***Figura 46**. Script CameraZoom.cs, CameraController.cs, CameraHeightController.cs*

Per poter assegnare in maniera randomizzata i tatuaggi a ciascun *Decal Projector* è stato implementato lo script *RandomTextureAssigner.cs*. Esso riceve come parametri in input il *Decal Projector* a cui è stato assegnato, le *Texture* da applicare sotto forma di array e infine il tasto di cattura (*Capture Key*), settato in questo caso come "R".

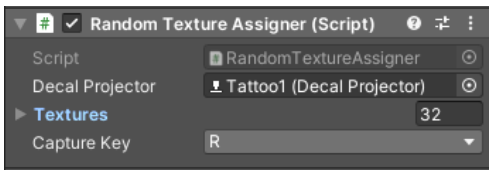

***Figura 47**. Script RandomTexureAssigner.cs*

Per poter acquisire le immagini è stato realizzato lo script *MultiCameraScreenshot.cs*. Esso ha come parametri l'array contenente le varie camere, posizionate nelle varie parti del corpo da

acquisire, le dimensioni di rendering (*Capture Width* e *Capture Height*), il tasto di cattura (*Capture Key*) ed infine la directory in cui salvare le istantanee acquisite (*Screenshot Path*).

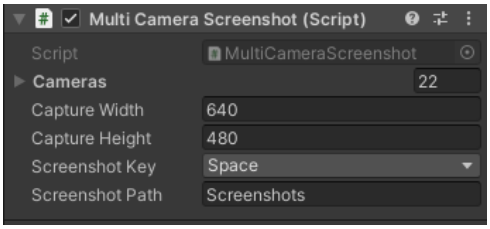

Figura 48. *Script MultiCameraScreenshot.cs*

Inoltre, per ottenere immagini delle varie parti del corpo sia contenenti i tatuaggi sia senza, è stato realizzato uno script, denominato *ToggleDecalProjectors.cs*, che disattivi e riattivi i *GameObject* corrispondenti ai *Decal Projector* (forniti come parametri nell'array *Objects*) premendo il tasto assegnato al *Key*, scelto come K.

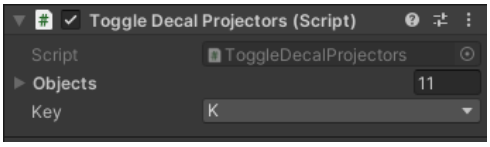

Figura 49. *Script ToggleDecalProjectors.cs*

Infine, sono stati implementati due elementi di tipo UI, per realizzare il bottone di uscita dal *tool* e per fornire indicazioni sull'acquisizione delle immagini. Con il tasto d'uscita, l'utente finale, ha la possibilità di interagire cliccando direttamente sullo schermo. Per far ciò, è stato realizzato lo script *ExitGame.cs*, di cui viene riportato il codice.

```
using UnityEngine;
using UnityEngine.UI;

public class ExitGame : MonoBehaviour
{
    private void Start()
    {
        Button button = GetComponent<Button>();
        if (button != null)
        {
            button.onClick.AddListener(Exit);
        }
    }

    private void Exit()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }
}
```

Figura 50. *Codice dello script ExitGame.cs*

Per esportare il *tool* sono stati eseguiti i seguenti passaggi chiave. Innanzitutto, è stato verificato che il progetto Unity fosse completo e pronto per l'esportazione. Quindi, è stata selezionata la piattaforma di destinazione desiderata, in questo caso il PC. Successivamente sono state impostate le impostazioni di build, come risoluzione, qualità grafica e versione di destinazione.

Una volta completate queste configurazioni, è stata selezionata l'opzione "Build" dal menu di Unity. Questo passaggio ha permesso di generare i file eseguibili, avviando il processo di compilazione dell'applicazione per la piattaforma scelta. Infine, sono stati eseguiti alcuni test per assicurarsi il corretto funzionamento.

Il *tool*, quindi, si presenta all'utente con l'interfaccia riportata nella Figura 51.

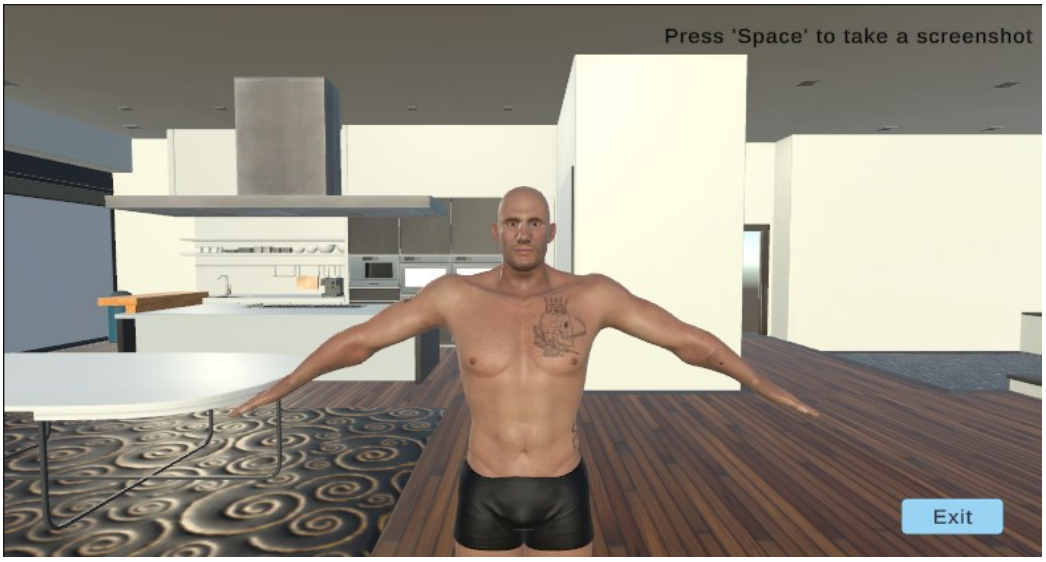

Figura 51. *Interfaccia finale del tool.*

PROCESSO DI SVILUPPO DELLA RETE NEURALE

Nella seconda parte del progetto sono stati applicati approcci di *transfer learning* al fine di sviluppare un sistema di rilevamento e di localizzazione dei tatuaggi all'interno delle immagine.

Nella realizzazione di questa sezione, è stato impiegato il software Matlab. L'utilizzo di questo software ha agevolato la prototipazione, l'ottimizzazione e la validazione dei modelli sperimentali, contribuendo in modo significativo alla precisione e all'efficacia dei risultati ottenuti.

Reti per il rilevamento delle immagini contenenti tatuaggi

Il primo *task* ad essere sviluppato è quello del rilevamento (*detection*) dei tatuaggi all'interno delle immagini. In maniera più specifica l'obbiettivo è quello di determinare se, all'interno dell'immagine fornita, sia presente o meno un tatuaggio, così come definito all'interno della *challenge* del Tatt-C.

Per farlo, è stato utilizzato l'approccio del *transfer learning* applicandolo ad alcune reti neurali convoluzionali. Le architetture implementate sono le seguenti: AlexNet, ResNet50, InceptionV3 e VGG16.

L'obbiettivo è quello di determinare quale architettura ottenga prestazioni migliori in termini di classificazione delle immagini nelle categorie 'Tattoo' e 'Non-tattoo', a seconda che contengano o meno tatuaggi.

All'interno della *Command Window* di Matlab è stato eseguito il comando deepNetworkDesigner, che permette di aprire il *tool* adibito a questo scopo. Le strutture sono state testate con l'ordine indicato in precedenza e per ognuna delle reti sono stati eseguiti i passaggi descritti.

In primo luogo, è stata caricata l'architettura della rete, così come si presenta nel modello pre-addestrato fornito da Matlab.

Sono stati rimossi gli ultimi strati della rete andando poi a personalizzarli per il problema specifico del rilevamento dei tatuaggi. Per farlo sono stati rimossi, in generale, l'ultimo strato completamente connesso (*Fully connected layer*) e lo strato di classificazione, che corrisponde all'*output* della rete.

Il problema è stato personalizzato andando a modificare l'OutputSize dell'ultimo strato completamente connesso, inserendo un valore di 2, che corrisponde alle classi definite nel

problema: *Tattoo* e *Non-Tattoo*. I pesi, associati agli strati di convoluzione, non sono stati modificati rispetto alle reti pre-addestrate.

I dati sono stati caricati mediante l'utilizzo della sezione *Data*, andando così ad importare le cartelle contenenti il dataset di immagini per il training della rete.

Come riportato in Figura 52 sono state selezionate alcune opzioni per permettere di aumentare la variabilità dei dati disponibili per il *training*. Ciascuna immagine è stata:

- Riflessa casualmente sia rispetto all'asse x e rispetto all'asse y;
- Ruotata casualmente in un *range* di gradi che varia tra -50 e 50;
- Scalata in maniera casuale nell'intervallo [1,5];
- Traslata orizzontalmente e verticalmente in modo casuale di un numero di pixel che varia tra -50 e 50.

Per la creazione del set di validazione, allo scopo di ridurre l'*overfitting* dei dati, è stato estratto il 30% delle immagini del *training set*.

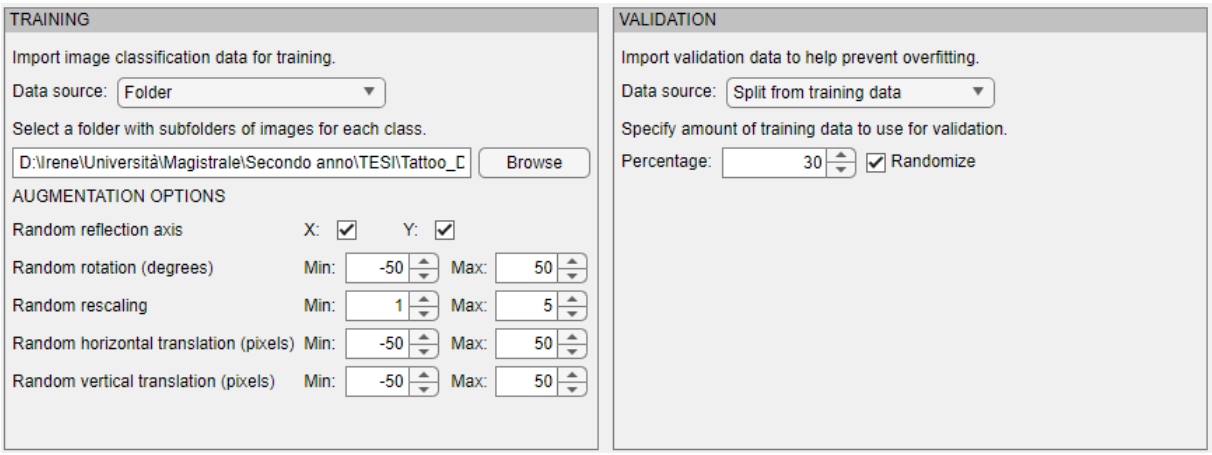

***Figura 52**. Importazione delle immagini usate per il training della rete neurale.*

Nelle Figura 53 e 54 sono riportati i dati importati, per il *training set* e per il *validation set* rispettivamente. Come si può osservare nel caso del set di addestramento sono state utilizzate 603 immagini appartenenti alla classe *Non-Tattoo* e 736 immagini appartenenti alla classe *Tattoo*. Per il set di validazione, invece sono state utilizzate 259 immagini della prima categoria e 316 della seconda.

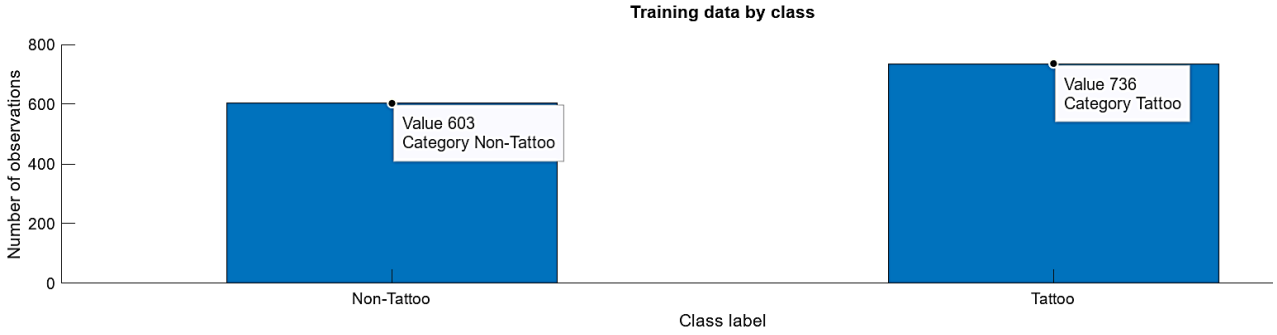

***Figura 53**. Numero di osservazioni per classe del set di addestramento*

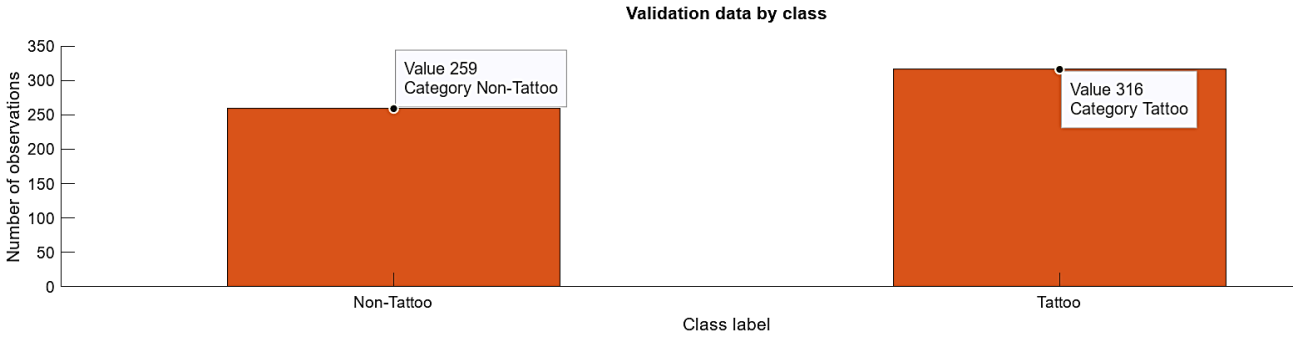

***Figura 54**. Numero di osservazioni per classe del set di validazione*

Per ogni rete neurale implementata, sono state esaminate cinque diverse configurazioni di parametri di addestramento. L'obiettivo era individuare la configurazione ottimale che garantisse il miglior livello di precisione nei risultati ottenuti durante i test.

Come si può osservare nella Tabella 2, tutte le configurazioni hanno mantenuto un numero massimo di epoche pari a 50 ed una frequenza di validazione pari a 30, mentre gli altri parametri sono stati variati come riportato.

	CONFIGURAZIONE				
	1	**2**	**3**	**4**	**5**
Solver	sgdm	sgdm	sgdm	adam	sgdm
InitialLearnRate	0.01	0.0001	0.0001	0.0001	0.00001
MiniBatchSize	32	32	32	32	16
MaxEpochs	50	50	50	50	50
ValidationFrequency	30	30	30	30	30
LearnRateSchedule	none	none	piecewise	none	none

***Tabella 2**. Tabella riassuntiva dei parametri di training.*

Le reti sono state addestrate mediante l'utilizzo di un singolo GPU ed hanno impiegato tempi variabili a seconda dell'architettura e della configurazione dei parametri di addestramento scelta.

Realizzazione del *test set* e valutazione delle reti

Per poter eseguire poi una validazione delle CNN implementate, è stato realizzato un dataset ristretto di immagini ricavate da internet mediante l'utilizzo dei motori di ricerca Google Immagini e Bing Immagini. Sono state ricavate 342 immagini contenenti soggetti tatuati e 196 immagini in cui non sono presenti tatuaggi.

Le immagini sono state memorizzate in ordine casuale e rinominate mediante uno script e salvate in formato PNG. Infine, è stato realizzato un file .MAT contenente un'etichetta manuale della categoria di appartenenza.

In Matlab è stato implementato lo script *tattooDetection.mlx* il quale ha lo scopo di leggere le immagini contenute nella cartella di riferimento, analizzare ciascuna di esse andando a classificarla con la rete neurale scelta, memorizzando l'accuratezza con cui essa viene assegnata alla classe. Dopo aver eseguito la classificazione, lo script consente di verificare, mediante il confronto con il file *ground truth*, quante di queste immagini siano state assegnate correttamente alla classe di appartenenza reale. I risultati ottenuti sono riportati nel Capitolo 4.

Localizzazione dei tatuaggi

Dopo aver analizzato e selezionato l'architettura e la configurazione di CNN che mostra risultati migliori in termini di riconoscimento di immagini contenenti soggetti tatuati, la rete è stata utilizzata per addestrare una Fast R-CNN, la quale permette di identificare anche la posizione in cui si trova il tatuaggio all'interno dell'immagine, andando poi a realizzare delle *bounding boxes* attorno al tatuaggio stesso.

Il primo passaggio ad essere realizzato è quello della preparazione del dataset per il *training* della rete.

In questo caso sono state considerate solamente le 1052 contenenti tatuaggi, in quanto la funzione Matlab utilizzata, richiede in input solamente immagini contenenti *bounding boxes*. Le immagini sono state caricate all'interno del *tool* Image Labeler di Matlab. Dopo aver creato, quindi, un nuovo progetto, è stata definita una nuova etichetta di forma rettangolare, così come mostrato nella Figura 55. L'etichetta è stata denominata "Tattoo".

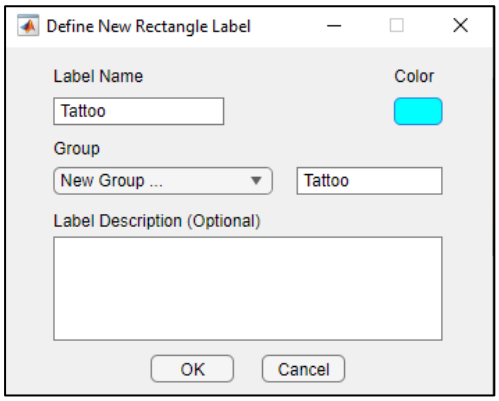

***Figura 55**. Definizione dell'etichetta rettangolare 'Tattoo'.*

Per ognuna delle immagini contenute nel dataset, sono state create manualmente delle *bounding boxes* attorno ai tatuaggi presenti. Questo è stato fatto sia nel caso in cui fosse presente un solo tatuaggio, sia nel caso in cui ce ne fossero molteplici. La Figura 56 mostra un esempio di etichettatura realizzato.

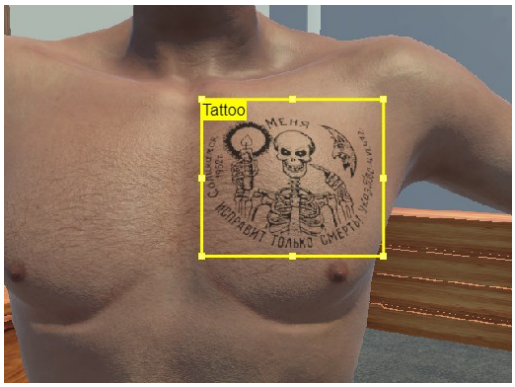

***Figura 56.** Esempio di immagine etichettata.*

Dopo aver ripetuto il processo per ognuna delle immagini, il file è stato esportato nel formato *table* ed importato nel *workspace* di Matlab.

Il training del *Fast R-CNN* è stato eseguito mediante la *function* già implementata in Matlab *trainFastRCNNObjectDetector.m*.

Le opzioni di *training*, così come riportato anche nella Figura 57 sono le seguenti:

- *Mini Batch Size*: 1
- *Initial Learn Rate*: 0.001
- *Max Epochs*: 8
- *Validation Frequency*: 50

Inoltre, sono stati specificati anche i valori '*NegativeOverlapRange'* e *'PositiveOverlapRange'* i quali si riferiscono rispettivamente all'intervallo di sovrapposizione tra le *bounding box*

generate dal modello e gli oggetti reali nell'immagine che vengono considerati campioni negativi e positivi durante l'addestramento.

Le *bounding box* con una sovrapposizione inferiore [0 0.3] vengono considerate campioni negativi e vengono utilizzate per l'addestramento del modello per identificare gli sfondi o le regioni non interessanti.

Impostando, invece, *PositiveOverlapRange* tra 0.6 e 1, significa che le *bounding box* devono sovrapporsi con almeno il 60% e al massimo il 100% dell'area degli oggetti reali per essere considerate campioni positivi.

```
options = trainingOptions('sgdm', ...
      'MiniBatchSize', 1, ...
      'InitialLearnRate', 1e-3, ...
      'MaxEpochs', 8, ...
      'VerboseFrequency', 200, ...
      'ValidationData', ds_validation, ...
      'ValidationFrequency', 50, ...
      'CheckpointPath', tempdir, ...
       Plots="training-progress");


detector = trainFastRCNNObjectDetector(ds_training, 'inceptionv3_5', options, ...
        'NegativeOverlapRange',[0 0.3], ...
        'PositiveOverlapRange',[0.6 1]);
```

Figura 57. *Training options per la rete Fast R-CNN.*

Realizzazione del *test set* e valutazione della rete

Per verificare la rete così realizzata, è stato successivamente implementato un *test set*, contenente 192 immagini reali, appartenenti alla classe 'Tattoo' e corrispondenti a circa il 20% delle immagini fornite alla rete in fase di allenamento. Queste immagini sono state selezionate tra quelle del *test set* della sezione precedente.

Ciascuna di queste immagini è stata etichettata manualmente mediante il *tool* Image Labeler, andando a definire l'etichetta "Tattoo".

All'interno di Matlab è stato definito uno script, denominato *tattooLocalizer.mlx,* che ha lo scopo di caricare le immagini ed esaminarle, attraverso la Fast R-CNN, mediante la funzione *detect.m*. Essa restituisce i seguenti argomenti:

- *bboxes*: coordinate delle *bounding boxes* espresse come [x y larghezza altezza];
- *scores*: punteggio che determina l'accuratezza della previsione;
- *labels*: etichetta mediante la quale viene identificato il tatuaggio.

Dopo aver ricavato la classificazione delle immagini della rete, i dati sono stati filtrati in modo tale da verificare la corrispondenza tra le righe del file *groundTruth* e quello ottenuto dalla Fast

R-CNN. Questa operazione è stata eseguita al fine di assicurarsi una corrispondenza tra le righe per essere sicuri di confrontare le stesse immagini tra loro e per verificare, nel caso di tatuaggi multipli, che il *target* fosse lo stesso.

Le classificazioni così ottenute sono state poi valutate per ricavare i risultati riportati nel Capitolo 4.

Capitolo 4

Risultati

In questo capitolo vengono esposti i risultati ottenuti dall'implementazione del *tool* realizzato mediante Unity e successivamente delle reti neurali.

RISULTATI OTTENUTI DALLA CREAZIONE DEL DATASET

Dopo aver esportato il *tool*, è stata creata una *directory* denominata "Tattoo_Dataset" nella quale sono state inserite due sottocartelle, ciascuna per ognuna delle seguenti categorie: *Tattoo* e *Non-Tattoo*.

Mediante l'utilizzo del *tool* sono state acquisite diverse immagini, spostando il soggetto principale in varie posizioni allo scopo di diversificare lo sfondo dell'immagine.

In totale sono state realizzate 1052 immagini contenenti tatuaggi e 862 immagini in cui sulla superficie del corpo non erano presenti, per un totale di 1914 immagini.

In Figura 58 e Figura 59 vengono riportate alcune immagini appartenenti rispettivamente alla categorie *Tattoo* e *Non-Tattoo*.

Figura 58. *Alcune delle immagini appartenenti alla categoria 'Tattoo'.*

Figura 59. *Alcune delle immagini appartenenti alla categoria 'Non-tattoo'.*

RISULTATI OTTENUTI MEDIANTE LE RETI NEURALI

Nei paragrafi successivi, verranno esposti i risultati derivanti dall'implementazione delle reti neurali. Inizialmente, sono riportati i risultati ottenuti dalla fase di *detection*, che è il processo di identificazione degli oggetti d'interesse all'interno di un'immagine.

Successivamente, è presentata la sezione dedicata alla localizzazione. La localizzazione è la fase successiva, in cui la rete neurale non solo identifica gli oggetti, ma li localizza anche all'interno dell'immagine, fornendo coordinate precise sulla loro posizione.

Valutazione delle reti per la *detection*

Al fine di eseguire una valutazione delle reti utilizzate per il task iniziale della *detection*, è stata calcolata l'accuratezza di validazione (*validation accuracy*) per ogni configurazione testata. In questa analisi preliminare è stato utilizzato il set di validazione, utilizzato in fase di *training*, composto da immagini sintetiche.

Considerando TP il numero di veri positivi, TN il numero di veri negativi e assumendo FP e FN rispettivamente come il numero di falsi positivi e falsi negativi, l'accuratezza può essere calcolata mediante la seguente formula:

$$accuratezza = \frac{TP + TN}{N_{tattoo} + N_{non-tattoo}}$$

Si configura come falso positivo l'attribuzione della classe *'Tattoo'* a un'immagine che in realtà non contiene alcun tatuaggio. Al contrario, si considera falso negativo il caso in cui un'immagine contenente effettivamente un tatuaggio viene erroneamente classificata nella categoria *'Non-tattoo'*.

Per ogni rete sono state calcolate allora 5 *validation accuracy*, dopo di che è stata calcolata la media tra i valori ottenuti. Nella Figura 60 sono indicati i valori di media ± deviazione standard di ogni rete presa in esame.

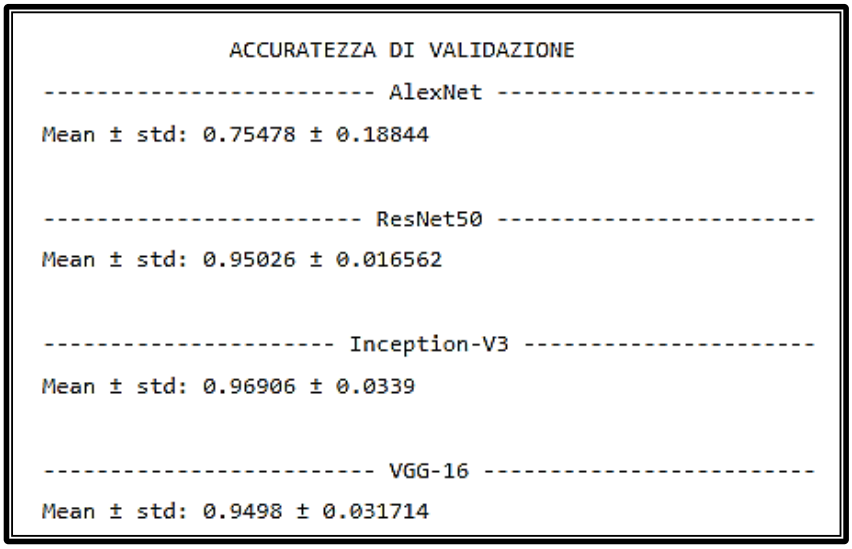

***Figura 60**. Media ± deviazione standard dell'accuratezza ottenuta con il set di validazione.*

Come si può osservare nella Figura, AlexNet è la rete che ha ottenuto prestazioni peggiori in termini di *validation accuracy*. Le altre reti invece mostrano accuratezze più vicine tra di loro. L'architettura che presenta valori di accuratezza più alti è la rete InceptionV3.

Al fine di stabilire se le variazioni osservate tra le reti sono dovute al caso o rappresentano differenze reali e significative, è stato eseguito un t-test di Student. Esso è un test statistico utilizzato per confrontare le medie di due gruppi e determinare se le differenze tra di essi sono statisticamente significative.

Prendendo come riferimento prima la rete AlexNet, e poi le altre reti, sono stati eseguiti dei t-test mediante la funzione ttest2, implementata all'interno di Matlab.

```
[h,p,ci,stats] = ttest2(x,y)
```

Il risultato principale da considerare è il valore p restituito dalla funzione. Un valore *p* inferiore alla soglia 0.05 indica che ci sono differenze statisticamente significative tra i gruppi.

I risultati di questa prima analisi sono riportati in un grafico a barre che riassume le grandezze incontrate.

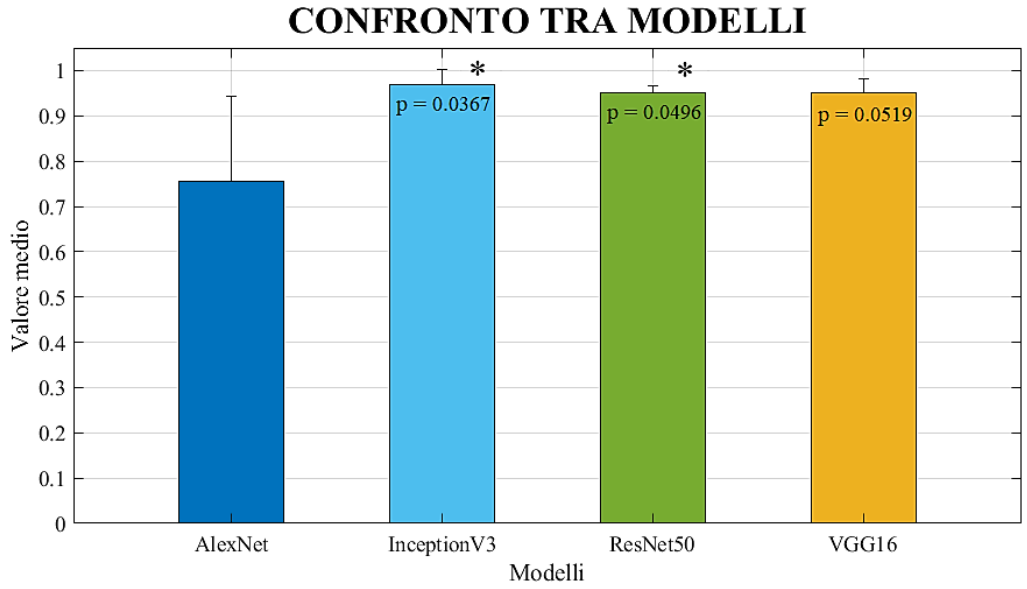

Figura 61. *Valori di media ± deviazione standard calcolati per ogni modello di rete considerato. Il simbolo * indica un valore p< 0.05 considerando come campione di riferimento la rete AlexNet.*

Sono stati osservati valori inferiori alla soglia 0.05 solamente nei confronti AlexNet-ResNet50 ed AlexNet-InceptionV3. In tutti gli altri casi i valori di *p* erano superiori alla soglia indicata; pertanto, non erano presenti differenze statisticamente significative tra i gruppi.

È possibile valutare poi le architetture mediante il *test set* realizzato in precedenza mediante l'utilizzo di immagini reali, le quali non sono mai state presentate alla rete, nemmeno in fase di validazione.

Anche in questo caso è stata calcolata l'accuratezza e i risultati sono stati riportati in Figura 62, indicando media ± deviazione standard, come nel caso precedente.

```
                ACCURATEZZA CON TEST-SET
------------------------- AlexNet ------------------------
Mean ± std: 0.67508 ± 0.036433

------------------------ ResNet50 ------------------------
Mean ± std: 0.67546 ± 0.048209

---------------------- Inception-V3 ----------------------
Mean ± std: 0.75314 ± 0.069136

------------------------- VGG-16 -------------------------
Mean ± std: 0.71053 ± 0.03565
```

Figura 62. *Media ± deviazione standard dell'accuratezza ottenuta con il set di testing.*

Come è possibile osservare, utilizzando il *test set* le prestazioni delle reti diminuiscono andando a raggiungere valori di accuratezza molto più bassi.

Al fine di effettuare un confronto tra i risultati ottenuti in questo caso e quelli ottenuti con il *validation set* è stato realizzato un grafico a barre che riporta i valori media ± deviazione standard nei due casi.

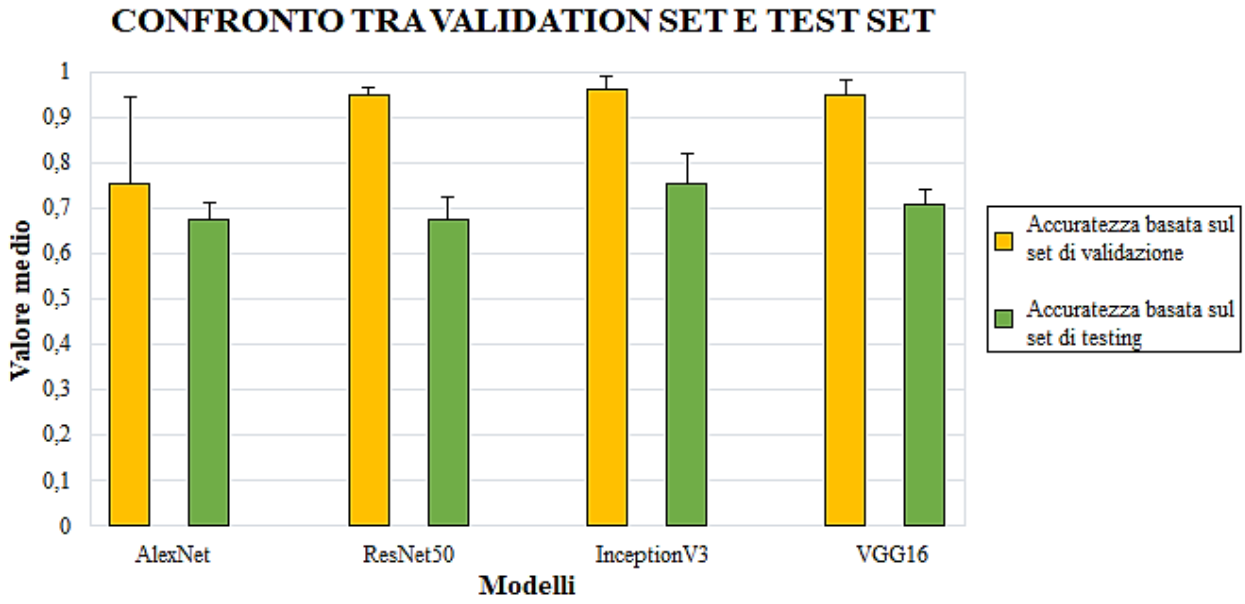

Figura 63. *Confronto tra accuratezza media ottenuta mediante il set di validazione e mediante il set di testing.*

In generale, è possibile osservare come le prestazioni delle reti neurali diminuiscano, al variare del dataset di immagini utilizzato. Anche con l'utilizzo di immagini reali InceptionV3 è la rete che mostra valori superiori alle altre, in termini di *accuracy*. In particolare, attraverso un'analisi approfondita delle diverse configurazioni, si è individuata la quinta configurazione come la più promettente, poiché presenta risultati superiori e più consoni alle aspettative, con un tasso di accuratezza pari all'87,17%.

Per questa rete, nella Figura 64, è riportata anche la matrice di confusione (*confusion matrix*) ottenuta mediante la funzione:

```
confusionchart(trueLabels,predictedLabels)
```

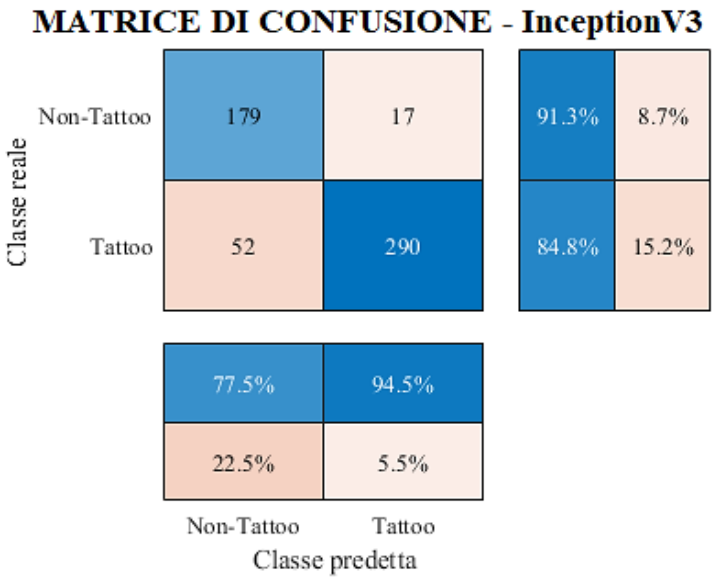

***Figura 64**. Matrice di confusione ricavata, mediante l'architettura InceptionV3, dalla configurazione che mostra i risultati migliori in merito al riconoscimento corretto delle immagini, appartenenti al set di testing.*

La matrice di confusione evidenzia in modo chiaro la performance del nostro modello di classificazione. I valori sulla diagonale principale rappresentano le predizioni corrette, indicando che il modello ha classificato correttamente la maggior parte delle istanze. Tuttavia, possiamo notare alcune discrepanze nei valori al di fuori della diagonale, che rappresentano errori di classificazione. Osservando questi valori, notiamo come 17 immagini rappresentino il numero di falsi positivi, suggerendo che il modello tende a classificare erroneamente alcune istanze come positive quando in realtà sono negative. D'altra parte, i falsi negativi, rappresentano un numero maggiore, mostrando che il modello tende a classificare immagini che contengono tatuaggi al loro interno, come se non ne avessero.

Sono state calcolate due ulteriori grandezze. A partire dalle informazioni contenute nella *confusion matrix,* il tasso di falsi positivi (*false positive rate*) è stato ottenuto come:

$$FPR = \frac{FP}{FP + TN}$$

Il tasso di falsi negativi (*false negative rate*) è stato, invece, ricavato mediante l'utilizzo della formula:

$$FNR = \frac{FN}{FN + TP}$$

Per la rete Inception-V3, considerando la configurazione 5 dei parametri di training, sono stati ottenuti allora i valori:

```
accuracy = 0.8717
FPR = 0.0867
FNR = 0.1520
```

Dai valori ottenuti, si possono trarre importanti conclusioni sulle prestazioni del modello di classificazione. Un tasso di falsi positivi (FPR) del 0.0867 indica che il modello commette un errore nell'identificare istanze negative come positive nell'8.67% dei casi. D'altra parte, un tasso di falsi negativi (FNR) del 0.1520 suggerisce che il modello perde il 15.20% delle istanze positive, classificandole erroneamente come negative. Questi dati sono stati confermati anche dall'output della *confusion matrix*.

Infine, è stata ottenuta la curva ROC (*Receiver Operating Characteristic*), la quale rappresenta uno strumento utilizzato per valutare le prestazioni di un modello di classificazione binaria. Rappresenta il trade-off tra il tasso di veri positivi (sensibilità) e il tasso di falsi positivi al variare della soglia di decisione del modello.

L'AUC (*Area Under the Curve*) misura l'area sotto la curva ROC ed è una metrica che fornisce una valutazione complessiva delle prestazioni del modello. Un'AUC più vicina a 1 indica un modello con ottime prestazioni nel discriminare tra le classi.

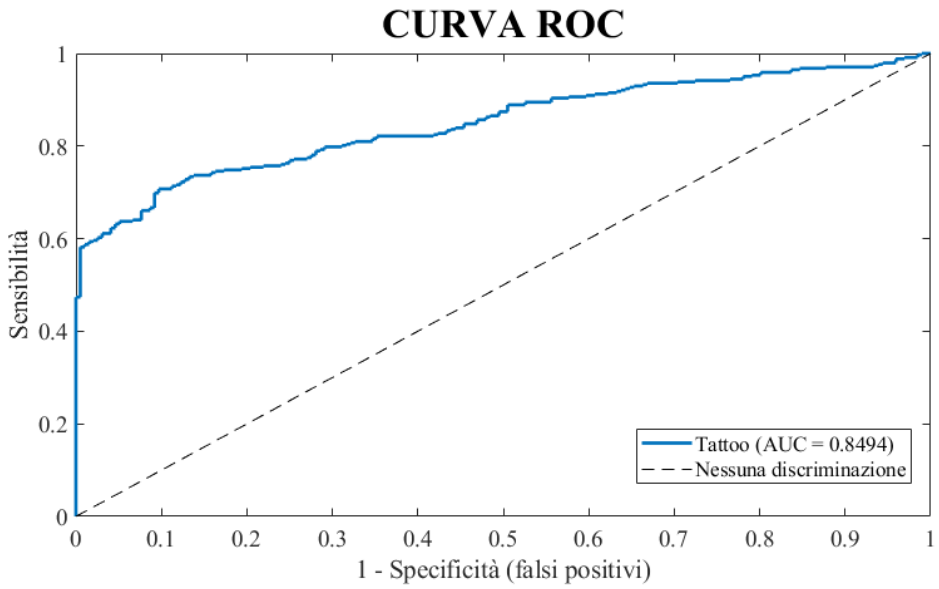

***Figura 65**. Curva ROC ottenuta per la rete InceptionV3 nella sua configurazione più performante.*

Dall'analisi del grafico emerge inizialmente che l'area sottesa alla curva raggiunge un valore di 0.8494, superiore rispetto all'area corrispondente alla linea tratteggiata, la quale si attesta a 0.5, indicando una classificazione casuale delle immagini. Il valore ottenuto suggerisce una buona capacità discriminante del modello, indicando un'elevata precisione nella classificazione

L'obiettivo preposto consiste nel massimizzare il parametro AUC, determinando curve che si avvicinino al massimo possibile all'angolo superiore sinistro del grafico. Questa posizione ideale corrisponde a una situazione ottimale con il 100% di sensibilità e il 100% di specificità. Pertanto, la ricerca di un AUC elevato riflette l'aspirazione a un modello di classificazione che dimostri un'elevata capacità di discriminazione e diagnostica.

Valutazione della FAST R-CNN

Per valutare i risultati ottenuti dall'applicazione del *test set* a questa architettura, è stata utilizzata la funzione Matlab

```
metrics = evaluateObjectDetection(detectionResults, groundTruthData, threshold)
```

Ad essa devono essere forniti:

- i risultati dell'operazione di *localization* contenenti: *bounding boxes, labels, scores;*
- il file *groundTruth* con: *bounding boxes, labels;*
- una o più soglie di sovrapposizione.

Al fine dell'analisi sono state considerate tre diverse soglie di sovrapposizione:

- 0.3 → per essere considerata una sovrapposizione valida, almeno il 30% dell'area di una regione deve sovrapporsi all'altra. Questa soglia è relativamente bassa e consente di rilevare regioni con una sovrapposizione parziale;
- 0.5 → questa soglia richiede che almeno il 50% dell'area di una regione si sovrapponga con l'altra regione per essere considerata valida. Questa è una soglia comune utilizzata in molte applicazioni, ad esempio nell'*object detection*, in cui si vuole essere abbastanza sicuri che le regioni si sovrappongano significativamente;
- 0.7 → con questo valore di soglia, almeno il 70% delle aree delle due regioni devono sovrapporsi tra di loro. Questo valore è più elevato e viene usato qualora si desideri ottenere una sovrapposizione molto significativa tra le regioni.

La funzione restituisce un oggetto di tipo objectDetectionMetric, il quale permette di analizzare diverse metriche associate al processo di riconoscimento e localizzazione.

Precision, *Recall* e la curva PR (*Precision-Recall*) sono metriche fondamentali nella valutazione dei modelli di rilevamento oggetti e di classificazione in vari campi, compresi l'elaborazione delle immagini e il *machine learning*. Queste metriche consentono di misurare l'efficacia e le prestazioni di un modello in diverse situazioni.

La *Precision* rappresenta la frazione di oggetti rilevati correttamente rispetto a tutti gli oggetti rilevati dal modello. In altre parole, misura quanto le previsioni positive del modello siano effettivamente corrette. Una *Precision* alta indica che il modello produce pochi falsi positivi, cioè poche previsioni erronee. Essa può essere calcolata mediante la formula:

$$precision = \frac{TP}{TP + FP}$$

Il *recall*, invece, misura la frazione degli oggetti rilevati correttamente rispetto a tutti gli oggetti veri presenti nel dataset. Rappresenta la capacità del modello di identificare correttamente tutti gli oggetti rilevanti. Un alto *recall* indica che il modello riesce a catturare la maggior parte degli oggetti veri. Si può ricavare come:

$$recall = \frac{TP}{TP + FN}$$

La curva *Precision-Recall* (PR) è uno strumento che permette di visualizzare il trade-off tra *precision* e *recall* in un modello di classificazione. Essa permette di valutare le performance del modello su diverse configurazioni. In generale, il modello viene considerato un buon modello predittivo se la precisione rimane elevata all'aumentare del *recall*. La forma della curva PR riflette le prestazioni del modello in modo più completo rispetto all'utilizzo di una singola metrica.

L'*Average Precision* (AP) rappresenta la media delle precisioni ottenute a diversi livelli di *Recall* e indica l'area sotto la curva PR. È una misura della qualità complessiva del modello di rilevamento, dove valori più alti indicano una maggiore precisione.

A partire da questi valori è possibile ricavare il valore di mAP (*mean Average Precision*), calcolata attraverso la media delle AP corrispondenti nella stessa tabella attraverso tutte le soglie di sovrapposizione.

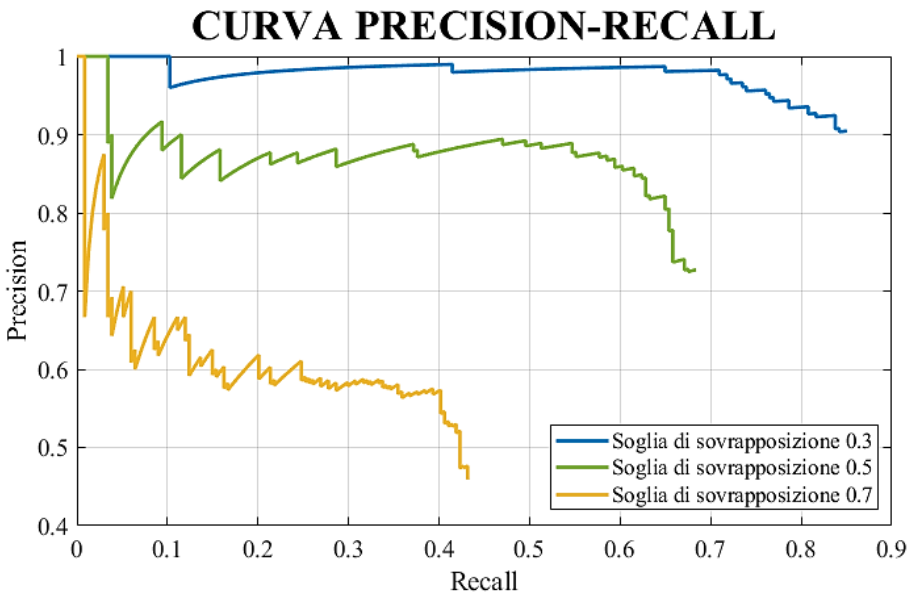

***Figura 66**. Grafico Precision-Recall per la rete Fast R-CNN nelle tre soglie di sovrapposizione oggetto dell'analisi.*

Nel grafico riportato in Figura 66, si nota che tutte e tre le curve sono inclinate verso il basso. Questo avviene perché, a mano a mano che la fiducia diminuisce, vengono fatte più previsioni (ciò che favorisce il *recall*), ma allo stesso tempo le previsioni diventano meno precise (danneggiando la precisione).

Guardando l'area sotto le curve, notiamo che, quando la soglia di sovrapposizione è pari a 0.3, otteniamo l'area più ampia. Ciò suggerisce che la precisione media è più elevata in questo caso. Infatti, con una soglia di sovrapposizione di 0.3, consideriamo valida la sovrapposizione del 30%. Ragionamenti analoghi possono essere applicati alle altre curve, suggerendo che al diminuire della superficie sotto la curva, le precisioni medie diminuiscono. Ciò suggerisce che potrebbero essere scartate più immagini in quanto non raggiungono la soglia specificata.

Nella Tabella 3 sono riportati i valori di AP ottenuti per ciascuna soglia di sovrapposizione.

	SOGLIA DI SOVRAPPOSIZIONE		
	0.3	0.5	0.7
AP	0.8317	0.5978	0.2677

***Tabella 3**. Valori di Average Precision (AP) ottenuti per ciascuna delle soglie di sovrapposizione.*

A partire da questi risultati è possibile ottenere il parametro *mAP*, il quale si attesta con un valore di:

$$mAP = 0.5657$$

Per ogni tatuaggio rilevato, sono stati poi calcolati il valore dell'indice IoU (*Intersection over Union*) e l'indice DICE (*Dice coefficient*), che sono due metriche comunemente utilizzate per valutare la sovrapposizione o la similitudine tra due insiemi o regioni, spesso nell'ambito della segmentazione delle immagini o dell'analisi degli oggetti. Per farlo, sono state implementate le funzioni *calculateIoU* e *calculateDICE*, le quali calcolano per ogni coppia di *bounding boxes* il valore corrispondente.

L'indice IoU, noto anche come coefficiente di Jaccard, può essere calcolato mediante l'utilizzo della seguente formula:

$$IoU = \frac{Area\ di\ sovrapposizione}{Area\ dell'unione}$$

Dove:

- "Area di sovrapposizione" indica l'area in cui la segmentazione prevista e la segmentazione di riferimento si sovrappongono.
- "Area dell'unione" rappresenta l'area totale coperta dalla combinazione di entrambe le regioni, ovvero è l'area della regione prevista più l'area della regione di riferimento meno l'area di sovrapposizione.

I valori che questo indice può assumere variano tra 0 e 1. In particolare:

- IoU = 0: significa che non c'è sovrapposizione tra le due regioni. Le regioni sono completamente separate o non si toccano affatto.
- IoU vicino a 0: indica una sovrapposizione molto limitata o parziale tra le regioni. Maggiore è il valore di IoU rispetto a 0, maggiore è la sovrapposizione tra le regioni.
- IoU = 0.5: un IoU di 0.5 è spesso utilizzato come soglia comune in applicazioni di rilevamento di oggetti. Significa che almeno il 50% dell'area delle due regioni si sovrappone.
- IoU = 1: un IoU di 1 indica una sovrapposizione perfetta tra le due regioni. Le regioni sono identiche o completamente sovrapposte.

Il valore medio di IoU, ricavato facendo la media tra gli IoU di ogni coppia di immagini, è:

$$mIoU = 0.5805$$

Questo valore indica una discreta sovrapposizione tra le predizioni della rete e i dati di riferimento forniti nel file *groundTruth*.

Inoltre, nella Figura 67, è riportata la distribuzione degli IoU per ogni coppia di immagini analizzate.

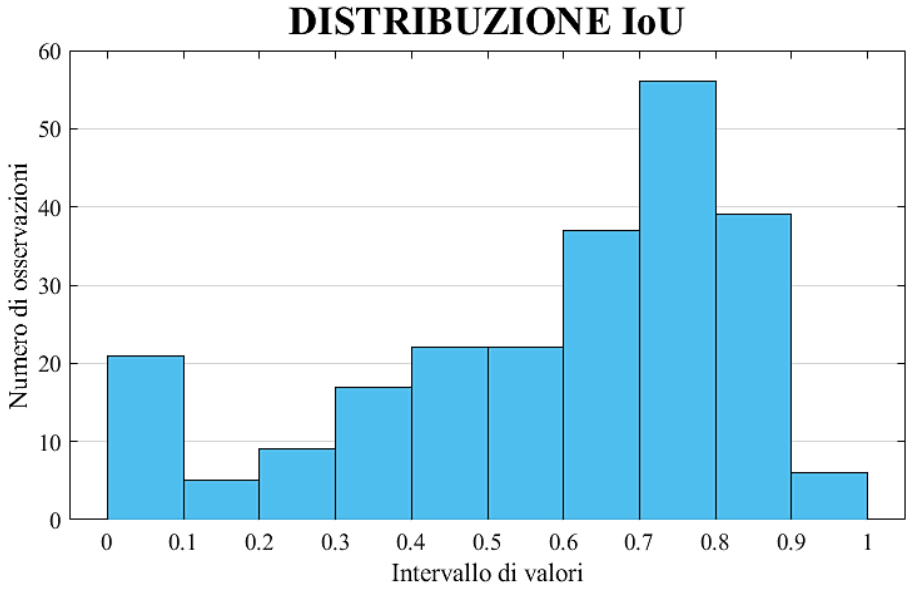

***Figura 67**. Distribuzione dei valori di IoU per ciascuna delle immagini contenuta nel set di testing.*

Anche l'indice DICE, noto anche come coefficiente di Sørensen-Dice, è una metrica di similarità utilizzata comunemente per misurare la sovrapposizione tra due insiemi o regioni. La formula utilizzata in questo caso è la seguente:

$$DICE = \frac{2 \times Area\ di\ intersezione}{(Area\ di\ A) + (Area\ di\ B)}$$

Dove:

- "Area di sovrapposizione" rappresenta l'area in comune tra due regioni o insiemi, come nella formula dell'Indice IoU.
- "Area di A" rappresenta l'area totale dell'insieme o regione *A*.
- "Area di B" rappresenta l'area totale dell'insieme o regione *B*.

Come per l'IoU, anche l'indice DICE restituisce un valore appartenente all'intervallo [0,1]. In modo particolare:

- DICE = 0: indica che non c'è sovrapposizione tra le due regioni. Questa metrica fornisce risultati simili a IoU.
- DICE vicino a 0: indica una sovrapposizione molto limitata o parziale tra le regioni.
- DICE ≈ 0.5: un DICE di circa 0.5 suggerisce una sovrapposizione moderata tra le regioni. Questo può essere interpretato come una buona corrispondenza parziale tra le regioni.
- DICE = 1: un DICE di 1 indica una sovrapposizione perfetta tra le due regioni.

Il valore medio di DICE, ricavato facendo la media tra gli DICE di ogni coppia di immagini, è:

$$mDICE = 0.6916$$

Questo risultato suggerisce una buona sovrapposizione parziale tra le regioni predette dalla rete neurale e i dati di riferimento forniti nel file *groundTruth*.

Nella Figura 68, è possibile osservare la distribuzione degli indici DICE per ogni coppia di immagini analizzate.

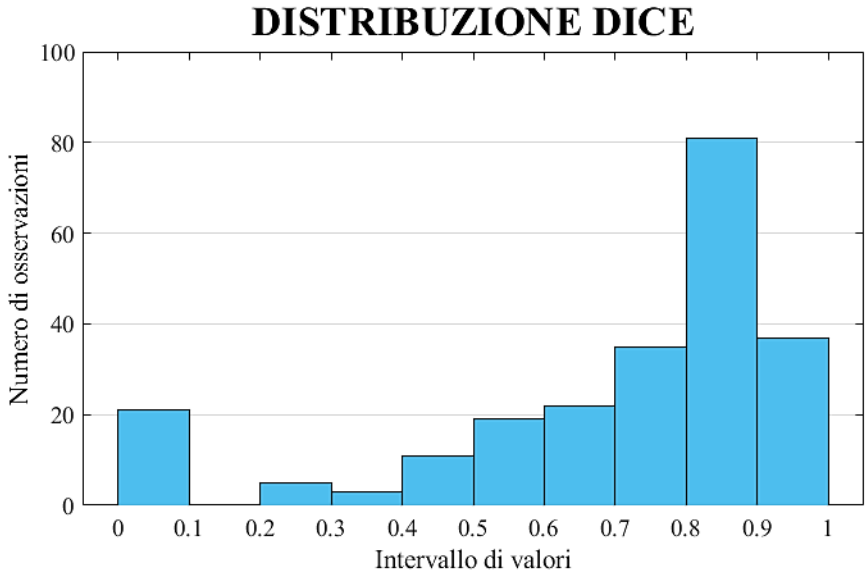

***Figura 68**. Distribuzione dei valori di DICE per ciascuna delle immagini contenuta nel set di testing.*

Conclusioni

I tatuaggi si ergono sempre più come tratti biometrici rilevanti poiché offrono un profilo unico e distintivo. Questa unicità è basata sulla permanenza fisica e sulla varietà di design possibili, creando un quadro personalizzato di identificazione. La stabilità temporale dei tatuaggi li rende inoltre pertinenti per applicazioni biometriche, in cui la precisione e la persistenza delle caratteristiche giocano un ruolo cruciale per garantire un'identificazione personale affidabile.

L'evoluzione nei metodi di individuazione di soggetti tatuati presenta un notevole passaggio dal tradizionale approccio manuale, basato sulla revisione visiva e l'analisi umana, all'impiego di tecnologie avanzate nell'ambito dell'intelligenza artificiale. Questo cambiamento si traduce in un significativo miglioramento dell'automazione e dell'efficienza del processo di riconoscimento, grazie all'utilizzo di algoritmi avanzati in grado di analizzare, confrontare e identificare tatuaggi con una precisione e velocità superiori rispetto alle capacità umane.

L’obiettivo di questo elaborato è stato quello di sviluppare un dataset di immagini sintetiche contenenti soggetti tatuati, al fine di fornire una risorsa essenziale per l'addestramento di reti neurali dedicate al riconoscimento di tatuaggi. L'utilizzo di simulazioni generate nell’ambiente di realtà virtuale Unity ha rappresentato quindi una soluzione innovativa per la creazione di un dataset che rispetti rigorosamente le leggi sulla privacy e le considerazioni etiche, senza compromettere l'efficacia delle applicazioni che ne derivano. È stato acquisito un set di 1052 immagini raffiguranti soggetti tatuati e 862 immagini prive di tatuaggi. Le differenze tra di esse comprendono variazioni nelle posizioni del corpo, soggetti ritratti, sfondi e condizioni di illuminazione.

In seguito, le immagini sintetiche sono state integrate in un sistema di intelligenza artificiale mediante l'utilizzo di Matlab al fine di addestrare reti neurali convoluzionali. Questo processo è stato eseguito per validare il dataset precedentemente creato. Nello specifico, le immagini sono state testate su reti neurali quali AlexNet, ResNet50, InceptionV3 e VGG16. Queste reti sono state valutate sia con immagini sintetiche, che con immagini reali, allo scopo di valutare le prestazioni in uno scenario reale di applicazione. La rete che ha ottenuto i risultati migliori, in termini di accuratezza nella classificazione, è stata InceptionV3, andando ad ottenere un valore di 0.96906 ± 0.0339 nel caso delle immagini sintetiche e 0.75314 ± 0.0691 utilizzando immagini reali. Questi valori sono stati ottenuti mediando le accuratezze delle reti di ognuna delle configurazioni testate. La configurazione che ha ottenuto valori più alti è stata la configurazione 5, ottenendo in questo caso un’accuratezza dell’87.17% con immagini reali. I

tassi di falsi positivi e negativi, assumono valori rispettivamente dell'8.7% e 15.2%. Inoltre, valutando la curva ROC è possibile osservare un'AUC di 0.8494.

Questa rete è stata scelta come modello principale per la fase successiva di training di un Fast R-CNN. Questo approccio garantisce che il modello di base sia performante nella rilevazione dei tatuaggi, fornendo una base solida per il successivo raffinamento e addestramento della rete Fast R-CNN per compiti di rilevamento specifici.

Attraverso l'impiego di Fast R-CNN, è stato possibile determinare la presenza di un tatuaggio nell'immagine ed anche localizzarne con precisione la posizione all'interno della stessa. Esaminando tre diverse soglie di sovrapposizione (0.3, 0.5, 0.7), si sono registrati i seguenti valori di precisione media (AP): 0.8317, 0.5978 e 0.2677. Aumentando la soglia di sovrapposizione, la precisione media diminuisce poiché meno immagini sono considerate come riscontri positivi. La media dei valori ottenuti restituisce un valore di mAP=0.5657. Gli indici IoU e DICE hanno registrato valori di 0.5805 e 0.6916, rispettivamente.

I risultati ottenuti hanno evidenziato che, nonostante un calo di prestazioni, le immagini sintetiche hanno consentito di discriminare anche le immagini reali. Tale capacità di generalizzazione testimonia l'efficacia del dataset sintetico nell'addestramento delle reti neurali, sottolineando la sua rilevanza nella creazione di modelli robusti capaci di affrontare sfide reali e garantire una maggiore versatilità nell'applicazione pratica.

Per affinare ulteriormente l'applicazione del *tool* sviluppato, sviluppi futuri potrebbero concentrarsi sull'arricchimento del dataset, ampliando la diversità di soggetti, tipologie di tatuaggi e sfondi utilizzati. Un'espansione qualitativa e quantitativa garantirebbe una maggiore robustezza nell'addestramento delle reti neurali. Inoltre, un'evoluzione della parte relativa alle reti neurali potrebbe coinvolgere l'implementazione di funzionalità avanzate, come la segmentazione del tatuaggio ed una successiva classificazione del soggetto raffigurato, al fine di poter valutare la similarità tra due o più immagini. Un ulteriore sviluppo strategico, pensato per un ipotetico scenario forense, potrebbe riguardare la creazione di un modello per attribuire un punteggio di pericolosità al soggetto rappresentato nel tatuaggio, collegandolo eventualmente alla criminalità organizzata.

Questi approfondimenti promuoverebbero la versatilità e l'applicabilità pratica del sistema, contribuendo ad una più ampia comprensione e utilità nel settore della biometria.

Bibliografia e sitografia

[1] Lee, Jung-Eun & Jin, Rong & Jain, Anil & Tong, Wei. (2012). **Image Retrieval in Forensics: Tattoo Image Database Application**. IEEE MultiMedia. 19. 40-49. 10.1109/MMUL.2011.59.

[2] Jain Anil K. and Ross Arun, 2015, **Bridging the gap: from biometrics to forensics**, *Phil. Trans. R. Soc. B***370**20140254025420140254 http://doi.org/10.1098/rstb.2014.0254

[3] Saini, Monika, and A. Kumar Kapoor. **Biometrics in forensic identification: applications and challenges**. *J Forensic Med* 1.108 (2016) DOI: 10.4172/2472-1026.1000108

[4] Meuwly, D., & Veldhuis, R. (2012, September). **Forensic biometrics: From two communities to one discipline**. In *2012 BIOSIG-Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG)* (pp. 1-12). IEEE.

[5] Unar, J. A., Seng, W. C., & Abbasi, A. (2014). **A review of biometric technology along with trends and prospects**. *Pattern recognition*, *47*(8), 2673-2688.

[6] Mendoza, a. I. P., 2017, **L'interpretazione delle emozioni per il criminologo sulla scena del crimine**, Sicurezza e Giustizia.

[7] A. K. Jain, A. Ross, S. Prabhakar, **An introduction to biometric recognition**, IEEE Trans. CircuitsSyst. VideoTechnol. 14 (2004) 4–20.

[8] https://www.treccani.it/vocabolario/tatuaggio/

[9] Buncke, Harry J. Jr. M.D; Conway, Herbert M.D.. **Surgery of decorative and traumatic tattoos**. *Plastic and reconstructive surgery* 20(1):p 67-77, july 1957.

[10] Sperry, Kris M.D.. **Tattoos and Tattooing Part I: History and Methodology**. *The American Journal of Forensic Medicine and Pathology* 12(4):p 313-319, December 1991.

[11] Birngruber, C.G., Martinez Peña, E.G., Corrales Blanco, L. *et al.* **The use of tattoos to identify unknown bodies**. *Rechtsmedizin* 30, 219–224 (2020). DOI: 10.1007/s00194-020-00396-y

[12] https://iltatuatore.it/dove-fare-il-tatuaggio/

[13] https://melmagazine.com/en-us/story/where-people-get-tattoos-is-highly-divided-by-gender

[14] Lee, J. E., Jain, A. K., & Jin, R. (2008, September). **Scars, marks and tattoos (SMT): Soft biometric for suspect and victim identification**. In *2008 Biometrics symposium* (pp. 1-8). IEEE.

[15] Rambhatla R, Jamgochian M, Ricco C, Shah R, Ghani H, Silence C, Rao B, Kourosh AS. **Identification of skin signs in human-trafficking survivors**. Int J Womens Dermatol. 2021 Oct 2;7(5Part B):677-682. doi: 10.1016/j.ijwd.2021.09.011.

[16] Tchalski, Rodrigo & Lopes, Heitor. (2021). **A Transfer Learning Approach for the Tattoo Detection Problem**. 1-8. 10.21528/CBIC2021-34.

[17] M. Ngan and P. Grother, **Tattoo recognition technology - challenge (Tatt-C): an open tattoo database for developing tattoo recognition research**, *IEEE International Conference on Identity, Security and Behavior Analysis (ISBA 2015)*, Hong Kong, China, 2015, pp. 1-6, doi: 10.1109/ISBA.2015.7126369.

[18] Jain, Anil K., Yi Chen and Unsang Park. **Scars , marks & tattoos (SMT) : physical attributes for person identification**. (2007).

[19] Jain, Anil & Lee, Jung-Eun & Jin, Rong. (2007). **Tattoo-ID: Automatic Tattoo Image Retrieval for Suspect and Victim Identification**. 256-265. 10.1007/978-3-540-77255-2_28.

[20] Heflin, Brian & Scheirer, Walter & Boult, Terrance. (2012). **Detecting and classifying scars, marks, and tattoos found in the wild**. 2012 IEEE 5th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2012. 31-38. 10.1109/BTAS.2012.6374555.

[21] Manger, Daniel. (2012). **Large-Scale Tattoo Image Retrieval**. 10.1109/CRV.2012.67.

[23] Han H, Li J, Jain AK, Shan S, Chen X. **Tattoo Image Search at Scale: Joint Detection and Compact Representation Learning**. IEEE Trans Pattern Anal Mach Intell. 2019 Oct;41(10):2333-2348. doi: 10.1109/TPAMI.2019.2891584. Epub 2019 Jan 9. PMID: 30629491.

[24] Hrkać, Tomislav & Brkić, Karla & Ribaric, S. & Marcetic, Darijan. (2016). **Deep learning architectures for tattoo detection and de-identification**. 1-5. 10.1109/SPLIM.2016.7528402.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. **ImageNet classification with deep convolutional neural networks**. Commun. ACM 60, 6 (June 2017), 84–90. https://doi.org/10.1145/3065386

[26] https://www.garanteprivacy.it/

[27] S. T. Acton and A. Rossi, **Matching and Retrieval of Tattoo Images: Active Contour CBIR and Glocal Image Features**, *2008 IEEE Southwest Symposium on Image Analysis and Interpretation*, Santa Fe, NM, USA, 2008, pp. 21-24, doi: 10.1109/SSIAI.2008.4512275.

[28] D. G. Lowe, **Distinctive image features from scale-invariant keypoints**, Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, Nov. 2004.

[29] Q. Xu, S. Ghosh, X. Xu, Y. Huang, and A. W. K. Kong, **Tattoo detection based on CNN and remarks on the NIST database**, in Proc. Int. Conf. Biometrics, Jun. 2016, pp. 1–7.

[30] X. Di and V. M. Patel, **Deep tattoo recognition**, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop, Jun. 2016, pp. 119–126.

[31] X. Di and V. M. Patel, **Deep Learning for Tattoo Recognition**, B. Bhanu and A. Kumar, Eds, Berlin, Germany: Springer, 2017.

[32] K. Simonyan and A. Zisserman. **Very deep convolutional networks for large-scale image recognition**. CoRR, abs/1409.1556, 2014.

[33] Z. Sun, J. Baumes, P. Tunison, M. Turek, and A. Hoogs, **Tattoo detection and localization using region-based deep learning**, in Proc. 23rd Int. Conf. Pattern Recognit., Dec. 2016, pp. 3055–3060.

[34] Chander, B., 2020. **Deep Learning Network**. , pp. 1-30. doi: 10.4018/978-1-7998-1159-6.ch001.

[35]https://it.mathworks.com/discovery/deeplearning.html?s_tid=srchtitle_support_results_1_tecnica%20del%20transfer%20learning

[36] Wei J. T., Zhang Z., Barnhill S. B., Madyastha K. R., Zhang H., Oesterling J. E., **Understanding artificial neural networks and exploring their potential applications for the practicing urologist**, Urology, Volume 52, Issue 2, 1998, Pages 161-172, ISSN 0090-4295, https://doi.org/10.1016/S0090-4295(98)00181-2.

[37] Rosenblatt F: **The perceptron: a probabilistic model for information storage and organization in the brain**. Psychol Rev **65:** 386–408, 1958.

[38] LeCun, Y., Bengio, Y. & Hinton, G. **Deep learning**. *Nature* **521**, 436–444 (2015). https://doi.org/10.1038/nature14539

[39] O'Shea, Keiron & Nash, Ryan. (2015). **An Introduction to Convolutional Neural Networks**. ArXiv e-prints

[40] S. Dara and P. Tumma, **Feature Extraction By Using Deep Learning: A Survey**, *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2018, pp. 1795-1801, doi: 10.1109/ICECA.2018.8474912.

[41] F. Zhuang *et al*., **A Comprehensive Survey on Transfer Learning**, in *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43-76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[42] S. Jialin Pan and Q. Yang, **A survey on transfer learning**, *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[43] Y. LeCun*, et al.*, **Gradient-based learning applied to document recognition**, *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[44] Saad Albawi, Tareq Abed Mohammed, **Understanding of a Convolutional Neural Network** *(2017)*

[45] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, **A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects**, in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827

[46] Sakib, S., Ahmed, N., Kabir, A.J., & Ahmed, H. (2019). **An Overview of Convolutional Neural Network: Its Architecture and Applications**. Preprints. DOI: 10.20944

[47] https://it.wikipedia.org/wiki/AlexNet

[48] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). **ImageNet Classification with Deep Convolutional Neural Networks**. Neural Information Processing Systems. 25. 10.1145/3065386

[49] S. Mascarenhas and M. Agarwal, **A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification**, *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, Bengaluru, India, 2021, doi: 10.1109/CENTCON52345.2021.9687944

[50] P. Yang, C. Dong, X. Zhao and X. Chen, **The Surface Damage Identifications of Wind Turbine Blades Based on ResNet50 Algorithm**, *2020 39th Chinese Control Conference (CCC)*, Shenyang, China, 2020, pp. 6340-6344, doi: 10.23919/CCC50068.2020.9189408

[51] Simonyan, Karen & Zisserman, Andrew. (2014). **Very Deep Convolutional Networks for Large-Scale Image Recognition**. arXiv 1409.1556.

[52] Pravitasari, Anindya & Iriawan, Nur & Almuhayar, Mawanda & Azmi, Taufik & Irhamah, Irhamah & Fithriasari, Kartika & Purnami, Santi & Ferriastuti, Widiana. (2020). **UNet-VGG16 with transfer learning for MRI-based brain tumor segmentation**. TELKOMNIKA (Telecommunication Computing Electronics and Control). 18. 1310. 10.12928/telkomnika.v18i3.14753.

[53] https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/

[54] R. Girshick, **Fast R-CNN**, *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[55] https://it.wikipedia.org/wiki/Unity_(motore_grafico)

[56] https://docs.unity3d.com/Manual/UsingTheEditor.html

[57] https://it.wikipedia.org/wiki/MATLAB

[58] https://it.mathworks.com/products/matlab.html

[59]https://it.mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html?searchHighlight=deepNetworkdesigner&s_tid=srchtitle_support_results_1_deepNetworkdesigner

[60] https://it.mathworks.com/help/deeplearning/deep-network-designer-app.html]

[61] https://it.mathworks.com/help/vision/ref/imagelabeler-app.html