

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

Dipartimento di
Informatica - Scienza e Ingegneria
DISI

Laurea in Ingegneria Informatica

Tesi di laurea
in
Controlli Automatici

**Sviluppo di uno Stack Software basato su Reti
Neurali per la Segmentazione Automatica di
Ecografie**

Candidata:

Caterina Leonelli

Relatore:

prof. Giuseppe Notarstefano

Correlatori:

*prof. Francesco Ursini
prof. Andrea Camisa
Ing. Lorenzo Sforni*

Anno Accademico
2021–2022

Indice

Abstract	4
Introduction	5
Motivazioni e stato dell'arte	5
Contributi e organizzazione	5
1 Nozioni base di Machine Learning	6
1.1 Algoritmi di Machine Learning	6
1.2 Capacità, Overfitting e Underfitting	8
1.3 Iperparametri e set di validazione	9
1.4 Deep Learning	10
2 Segmentazione tramite Deep Learning e creazione del dataset di immagini ecografiche	16
2.1 Segmentazione semantica multiclasse per il riconoscimento di oggetti	16
2.2 Generazione del dataset	19
2.3 Rete U-Net	23
3 Implementazione	24
3.1 Interfaccia di utilizzo	24
3.2 Descrizione e caricamento del dataset	26
3.3 Creazione del modello - Reti VGG	26
3.4 Training	30
3.5 Predizioni	34
4 Risultati dei test	35
4.1 Predizioni con rete VGG16 U-Net in Transfer Learning e Fine Tuning	36
4.2 Predizioni con rete VGG19 U-Net in Transfer Learning e Fine Tuning	42
5 Conclusioni	49
6 Ringraziamenti	50

Abstract

Lo scopo di questa tesi di laurea è di implementare ed analizzare un caso di utilizzo delle tecniche di *Deep Learning* per la segmentazione semantica delle immagini ecografiche. Il set di dati utilizzato è composto da delle ecografie delle mani ricavate dal database dell'Istituto Ortopedico *Rizzoli*. I dati sono costituiti da immagini della mano, dove ciascuna immagine è stata annotata manualmente con 3 etichette diverse. Le 3 classi sono: osso, cavità sinoviale e tendine.

Per addestrare i modelli di Deep Learning, è stata utilizzata un'architettura di rete neurale convoluzionale (CNN) della tipologia U-net. L'input per la rete è un'immagine di dimensioni 224 x 224 x 3 e l'output è un vettore a 3 dimensioni corrispondente alle probabilità di classe previste per ciascuna delle 3 classi. Il modello addestrato è stato quindi valutato su un set di prova di 60 immagini. I risultati hanno mostrato che la rete è stata in grado di ottenere una accuratezza di più del 90,0% sia sul set di training sia sul set di test. Inoltre poichè il dataset risulta sbilanciato è stato necessario utilizzare altre metriche di valutazione delle predizioni diverse dalla *accuracy*, come la *F1-score*, raggiungendo un valore medio maggiore del 90,0%.

Introduzione

Motivazioni e stato dell'arte

Nell'ambito del Machine Learning e più nello specifico del Deep Learning, si possono allenare delle macchine a classificare un insieme di dati in input. In particolar modo, qualora i dati fossero delle immagini in 2D, esistono delle architetture che allenano la macchina a classificare ogni pixel di una delle immagini in input. Questo task, definito come “segmentazione”, permette dunque di allenare una macchina a riconoscere delle aree semanticamente diverse all'interno dell'immagine. Ciò può essere utilizzato per qualsiasi tipo di immagine, in particolare, in ambito medico. In questa tesi viene applicata la segmentazione semantica ad un insieme di ecografie del metacarpo per il riconoscimento della cavità sinoviale, dell'osso e del tendine. In particolare, i cambiamenti di volume della cavità sinoviale possono essere registrati ed analizzati per prevenire e/o curare l'artrite reumatoide. Infatti, quando la patologia artritica è attiva, si possono verificare delle alterazioni anatomiche a livello di osso, tendine e cavità sinoviale, che possono essere patognomoniche di artrite reumatoide. Attualmente in Italia questo tipo di diagnosi viene in generale effettuata da un medico reumatologo che con l'esperienza e occhio clinico riesce ad assegnare il grado di severità del danno. Segmentare manualmente le immagini ecografiche è dipendente dall'abilità ed dall'esperienza dell'operatore, e questo potrebbe portare a una diagnosi soggettiva; inoltre, per un operatore del settore è dispendioso in termini di tempo rivedere un grande volume di immagini cliniche. Pertanto, la segmentazione semantica potrebbe avere un ruolo importante nel facilitare la diagnosi differenziale con altre patologie articolari.

Struttura e organizzazione

Il capitolo 1 fornisce una introduzione sintetica delle nozioni base di Machine Learning, dunque si spiega che cosa sia un algoritmo di Machine Learning, per poi spiegare che cosa sia una sua sottosezione chiamata Deep Learning. Il capitolo 2 tratta in maniera più dettagliata il task di segmentazione semantica utilizzando tecniche di Deep Learning utilizzando come esempio di rete neurale la U-net. Il capitolo 3 presenta l'implementazione della segmentazione semantica multiclassificazione applicata al dataset dell'Istituto Ortopedico *Rizzoli* di ecografie al metacarpo. Il capitolo 4 mostra i test realizzati con l'implementazione descritta nel capitolo precedente.

Capitolo 1

Nozioni base di Machine Learning

1.1 Algoritmi di Machine Learning

Si definisce “algoritmo di *Machine Learning*” un algoritmo che è capace di *imparare* da dei nuovi dati che gli vengono inseriti in input [1]. Mitchell (1997) [2] propone una definizione di “learning” applicabile ad un algoritmo:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

1.1.1 Tipologie di task

I task di Machine Learning sono i diversi obiettivi che l'algoritmo può avere e vengono descritti in termini di quali passi l'algoritmo deve percorrere per processare un esempio. L'esempio è una collezione di caratteristiche (o *features*) misurate quantitativamente: viene rappresentato come un vettore $\mathbf{x} \in \mathbb{R}^n$ dove ogni sua componente x_i è un'altra feature stessa. Esempi di feature di un'immagine possono essere i valori dei pixel che la compongono. Chiameremo una collezione di esempi con il nome di dataset oppure *data points*. I più comuni task di Machine Learning sono:

- **Classificazione** (o *classification*): è un tipo di task nel quale l'algoritmo decide a quale categoria appartiene un certo dato in input. In questo tipo di task, l'algoritmo deve trovare un funzione $f : \mathbb{R}^n \mapsto \{1 \dots k\}$ tale che $y = f(x)$ dove \mathbf{x} è il vettore dei dati in input da classificare, mentre l'insieme $\{1 \dots k\}$ rappresenta l'insieme delle categorie. Perciò la macchina deve riuscire a dare in output una predizione $f(x)$ scelta tra l'insieme delle categorie disponibili il più possibile simile alla categoria reale associata y . Solitamente l'insieme delle categorie è un valore numerico oppure può essere anche rappresentato da un vettore contenenti una probabilità per ogni classe. Ad esempio, si consideri il dataset di MNIST [3] come insieme di input, ovvero un insieme di immagini di cifre da 0 a 9 disegnate a mano. L'obiettivo finale della macchina è quello di capire

qual è la cifra rappresentata da ogni immagine. Si potrebbe impostare la macchina in modo che per ogni immagine e quindi per ogni vettore in input \mathbf{x} decida di assegnare una categoria da 0 a 9; oppure si potrebbe impostare la macchina in modo che ad ogni vettore in input \mathbf{x} indichi per ogni categoria la probabilità che \mathbf{x} appartenga a quella categoria, perciò che ad ogni input \mathbf{x} trovi un vettore \mathbf{y} di dimensione 9 dove ogni componente i -esimo è la probabilità che \mathbf{x} faccia parte della categoria i -esima.

- Regressione (o *regression*): in questo task la macchina deve predire un output numerico continuo, dato un qualche tipo di input, ovvero di trovare una funzione $f : \mathbb{R}^n \mapsto \mathbb{R}$. Ad esempio, dato in input le caratteristiche di un immobile, si può chiedere alla macchina di predire quale sarà il suo prezzo di mercato. Si noti che la regressione differisce dalla classificazione, in quanto l'output non appartiene ad un insieme finito, ma può essere un valore numerico appartenente all'insieme dei numeri reali. In altre parole, la classificazione ha come obiettivo quello di predire una variabile discreta, mentre la regressione ha come obiettivo quello di predire una variabile continua. Si noti che, come discusso al punto precedente, la classificazione può usare come output una probabilità, ma anche se essa è una variabile continua, questo tipo di task non è considerato un task di regressione. Allo stesso modo, anche la regressione potrebbe produrre in output un valore discreto, ma questo si verifica quando l'output appartiene all'insieme dei numeri interi. È importante non confondere questi due task dato che le loro implementazioni presentano caratteristiche diverse.
- Trascrizione (o *transcription*): è un task dove la macchina analizza un dato in input e si ottiene in output una descrizione testuale delle informazioni processate. Un esempio di questo tipo di task è la *speech recognition*, ovvero il processo mediante il quale il linguaggio orale umano viene riconosciuto e successivamente elaborato per essere convertito in forma testuale.

1.1.2 Misurazione delle performance

Un algoritmo di Machine Learning deve poter essere valutato in maniera quantitativa. Spesso questa misura è specifica del task scelto; ad esempio, per i task di classificazione e trascrizione si sceglie di misurare la accuratezza (o *accuracy*) del modello. La accuratezza è definita come il rapporto tra il numero delle predizioni corrette e il numero totale di predizioni effettuate. Il suo corrispettivo si chiama *error rate*, ovvero il rapporto tra il numero di predizioni non corrette sul numero totale di predizioni effettuate. È buona prassi valutare il modello su dei dati non processati durante il training, dunque si deve preparare un test set, ovvero un insieme di dati diversi da quelli inseriti in input durante il training, e su questi la macchina farà delle predizioni. Scegliere le misura da utilizzare per valutare le performance di un modello è fondamentale e soprattutto dipende molto dallo scenario applicativo; si veda per questo la Sezione 1.4.3.

1.1.3 L’Esperienza - Supervised e Unsupervised Learning

Gli algoritmi di Machine Learning vengono classificati in supervisionati (o *supervised*) e non supervisionati (o *unsupervised*). La differenza tra questi due approcci sta in quella che viene definita come “esperienza” dell’algoritmo: l’algoritmo può imparare in maniera *diretta*, ovvero quando ha dei feedback diretti su ciò che deve essere processato in output, oppure in maniera *indiretta*, e quindi quando non gli viene fornito alcun feedback su ciò che deve essere ottenuto in output. Un algoritmo supervisionato processa un dataset che contiene un certo numero di features e per ognuno degli esempi che lo compongono viene associata una *label* o *target*. Considerando l’esempio di classificazione con il dataset MNIST [3], ad ogni immagine viene associato un numero indicante la cifra che rappresenta l’immagine stessa. Il termine “supervised learning” vuole richiamare il concetto di uno scenario in cui un istruttore o insegnante mostra all’algoritmo quello che deve ottenere. Anche un algoritmo non supervisionato processa un dataset contenente tante features, però solo da questi dati deve capire quali sono le proprietà utili della sua struttura stessa. Nell’analoga descritta precedentemente, un algoritmo non supervisionato non ha alcuna guida su come processare il dato, ma deve crearla da solo. Ad esempio [2], si consideri una macchina che deve imparare a giocare a scacchi: essa può essere allenata con delle informazioni dirette, ad esempio con una lista di mosse corrette e una di non corrette, oppure con informazioni indirette, ovvero con una lista di partite giocate senza definire quali mosse siano corretto o meno. In questo ultimo caso l’unico feedback che la macchina ha è il risultato della partita (vittoria o sconfitta).

1.2 Capacità, Overfitting e Underfitting

L’obiettivo di un algoritmo di Machine Learning è di imparare a compiere il task definito con un dataset specifico (chiamato dataset di training, o training set o train set), per saper realizzare lo stesso compito su dei dati nuovi e che quindi non ha ancora processato. L’abilità di performare bene su dei dati mai visti prima viene chiamata “capacità di generalizzazione” (o *generalization*). In particolare, non si vuole ridurre solo l’errore delle predizioni fatte durante il training (chiamato *training error*), come sarebbe tipico fare in un semplice problema di ottimizzazione, ma si vuole anche ridurre il più possibile l’errore commesso sui dati di test (ovvero il *test error*). La maggiore difficoltà in questo caso risiede sul fatto che il test set non è osservabile o conoscibile a priori, però il campo della teoria dell’apprendimento statistico (o *statistical learning theory*) afferma che se il training set e il test set sono stati formati con un certo criterio, allora possiamo fare delle assunzioni e a partire da queste apportare delle modifiche all’algoritmo per migliorarlo. Alcune di queste assunzioni (chiamate anche *i.i.d assumptions*) possono essere:

- Gli esempi in ogni dataset sono indipendenti l’uno dall’altro
- Il training set e il test set sono identicamente distribuiti

Con queste assunzioni, si può assegnare la stessa distribuzione probabilistica per ogni singolo esempio, ottenendo quello che si può chiamare un “framework probabilistico”. Questo framework e la i.i.d assumption ci permettono di studiare in maniera matematica e quindi a priori le relazioni tra il test error e il training error. Infatti, si può affermare che per ottenere delle buone performance da un algoritmo di Machine Learning è importante ridurre il più possibile il training error e il gap tra il training error e il test error. I valori di questi due fattori possono infatti indicare se l'algoritmo ha un comportamento di *underfitting* oppure di *overfitting*: si ottiene un comportamento di underfitting quando il modello non riesce a ridurre a sufficienza il training error; invece si ottiene un comportamento di overfitting quando il gap tra il training error e il test error è troppo grande. Questi comportamenti si manifestano in relazione alla capacità (o *capacity*) del modello. In particolare, la capacità di un modello è la sua abilità di essere adatto a una vasta gamma di funzionalità. Questo vuol dire che se un algoritmo di Machine Learning ha una bassa capacità, allora non è in grado di realizzare al meglio il task predefinito e perciò si otterrà un training error molto alto. Al contrario, se un algoritmo di Machine Learning ha una alta capacità, allora sarà in grado di performare bene sul training error, e quindi si sarà adattato all'insieme di esempi in input, tanto da poi non riuscire ad applicare ciò che ha imparato nel test set. Di conseguenza, ciò si manifesta in un maggiore gap tra training error e test error. Dunque lo scopo di un algoritmo di Machine Learning è di avere una capacità appropriata al task che deve compiere con il training set assegnato e questo viene ottenuto quando la macchina è portata a scegliere in maniera appropriata e calibrata lo spazio di soluzioni, ovvero le funzioni $f(x)$. Ci sono molti approcci per esprimere preferenza tra le varie soluzioni e questi fanno parte di quelle che vengono chiamate tecniche di regolazione (o *regularization*).

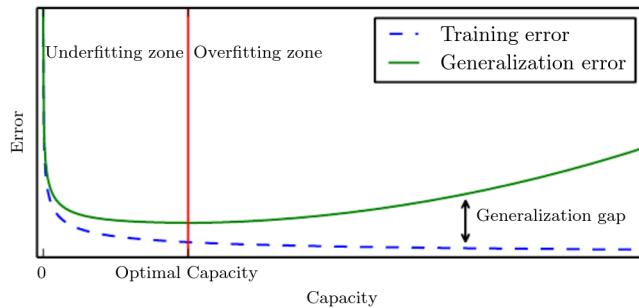


Figura 1.1: Grafico della relazione tra training error, test error e capacità di un algoritmo [2]

1.3 Iperparametri e set di validazione

Per controllare i comportamenti di un algoritmo di Machine Learning spesso si utilizzano dei parametri specifici chiamati *iperparametri*. Sono dei parametri regolati tipicamente dall'esterno e non internamente dall'algoritmo stesso durante il training: grazie agli iperparametri si può infatti controllare la capacità del modello e se questi fossero calcolati dall'algoritmo

come per il calcolo dei parametri del training allora si incorrerebbe in un problema di overfitting degli iperparametri stessi. Perciò è buona norma utilizzare un altro insieme di dati in input, chiamato *validation set* che l'algoritmo si serve per trovare il valore migliore di questi iperparametri. Solitamente il validation set si crea a partire dal training set: un 20% del training set iniziale viene usato per formare il validation set, mentre il restante 80% viene usato per l'effettivo training set. Per non rendere ambigua la notazione di questi due tipi di dataset, d'ora in avanti verrà chiamato “training set” l'insieme di dati utilizzato per allenare i parametri del modello durante il training, invece il “validation set” è l'insieme di dati usato per stimare il generalization error o test error dopo o durante la fase di training con lo scopo di aggiornare gli iperparametri.

1.4 Deep Learning

Il Deep learning viene definito come una sottoclasse di tecniche di Machine Learning che sfruttano molti layers (o layers) di elaborazione non lineare delle informazioni. Viene sfruttato principalmente per l'apprendimento supervisionato o non supervisionato di *features*, con i task di trasformazione o per l'analisi e la classificazione di pattern all'interno dei dati [4]. Il Deep Learning è dunque un modo per definire un approccio dell'*Artificial Intelligence* che adopera una *Artificial Neural Network*, detta anche *Neural Network*. Il concetto di rete neurale è stato proposto la prima volta nel 1944 da Warren McCullough e Walter Pitts, due ricercatori dell'università di Chicago [5]. Una rete neurale consiste in una vasta quantità di nodi attivi densamente interconnessi tra loro, a somiglianza con quello che vuole essere una descrizione semplificata del cervello umano costituito da neuroni. La maggior parte dei nodi sono organizzati in l layers dove un singolo nodo può essere connesso ad una molteplicità di nodi dei layers successivi ad esso. Dunque ogni nodo riceve dei dati processati dai nodi precedenti, applica il suo processamento e invia i nuovi dati ai nodi dello layer successivo. Per ogni connessione entrante, un nodo assegna un numero noto di “pesi”: quando la rete è attiva il nodo riceve dei dati differenti ad applica ad essi il peso assegnato. Successivamente vengono sommati tutti questi prodotti e se il risultato supera una certa soglia (chiamata *threshold*) allora esso viene passato allo layer successivo, altrimenti viene scartato. Durante il training i pesi e le soglie vengono continuamente aggiustati con lo scopo di diminuire l'errore nelle predizioni fatte dall'algoritmo [6].

Considerando l'esempio del dataset di MNIST [3], una possibile rete neurale che utilizza quei dati in input potrebbe essere realizzata con la seguente forma:

1.4.1 Deep Feedforward Networks

Le *Deep Feedforward Networks*, chiamate anche *Feedforward Neural Networks* o *Multilayer Perceptrons* (MLPs) sono delle reti neurali che hanno lo scopo di approssimare il più possibile la funzione $f(x)$ alla predizione reale y . Ad esempio, in un task di classificazione la rete neurale ha l'obiettivo di calcolare $f(x)$ tale che, dato \mathbf{x} come vettore di input e y come valore della categoria corretta, risulti valere $y = f(x)$. Solitamente è molto complicato

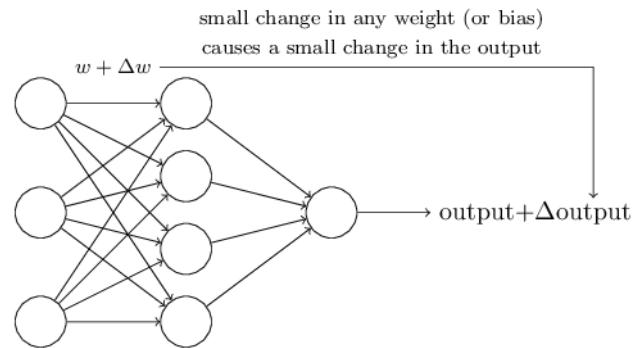


Figura 1.2: Disegno che mostra la relazione tra il cambiamento dei pesi da un layer della rete neurale all'altro e il cambiamento dell'output [7]

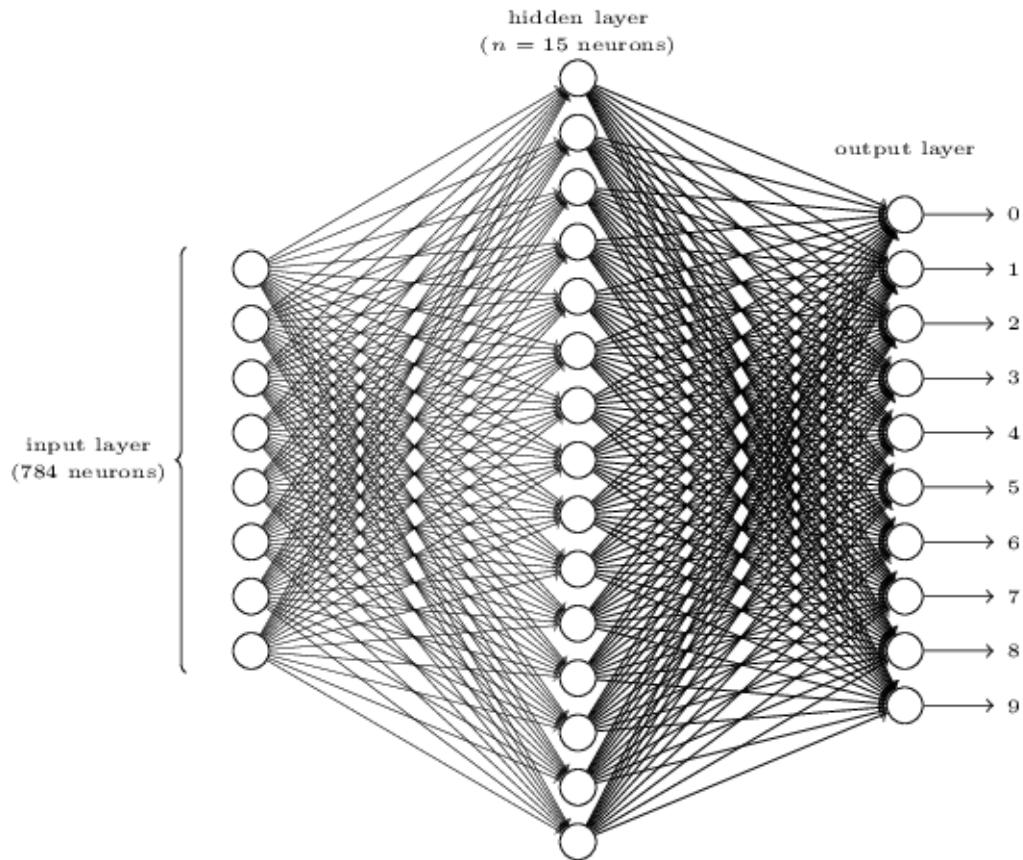


Figura 1.3: Esempio di una rete neurale in feedforwarding che processa i dati del dataset MNIST [7]

ottenere un'uguaglianza di questo tipo, quindi lo scopo finale di questo tipo di rete neurale è quello di trovare una funzione $f(x, \theta)$ che, dati certi parametri aggiuntivi di ingresso θ , approssimi il più possibile la y . Il modello si dice essere in “feedforward” perché le informazioni fluiscono dal vettore in input \mathbf{x} , attraverso i layers intermedi usati per definire $f(x, \theta)$, fino a all’output y . Si definisce così *input layer* il layer dove i dati vengono inseriti in ingresso, *output layer* il layer finale, mentre i layer intermedi vengono chiamati *hidden layer*. Si consideri un vettore di input \mathbf{x} e il vettore delle predizioni corrette associate y

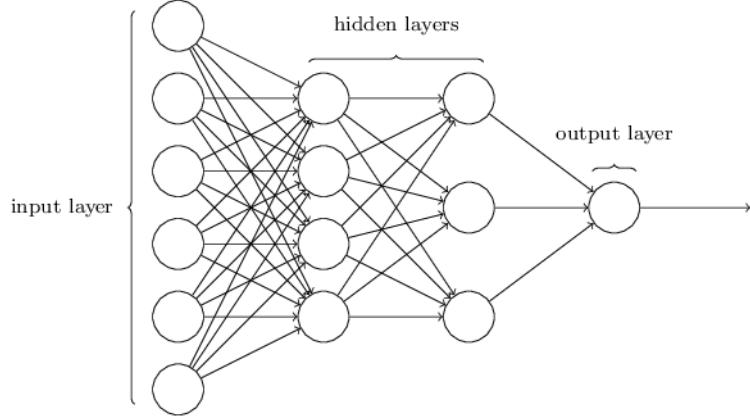


Figura 1.4: Disegno che mostra una rete neurale suddivisa in input layer, hidden layers e output layer [7]

e si consideri che la macchina produca in output un vettore di predizioni y^{pred} . Allora si definisce *loss function* $l(y_i, y_i^{pred})$ la funzione che misura la qualità della predizione, dove y_i è la predizione corretta, mentre y_i^{pred} è la predizione prodotta dall’algoritmo.

Se la loss function risulta valere 0 allora $y = y^{pred}$, altrimenti la loss function risulterà essere maggiore di zero. Per allenare l’algoritmo si definisce una funzione di costo (o *cost function*):

$$F(\theta) = \frac{1}{N} \sum_{n=1}^N l(y_i, f(\theta, x_i)) \quad (1.1)$$

con N il numero di tuple (x_i, y_i) . Lo scopo della rete neurale è quello di minimizzare la cost function per ottenere così performance migliori. Per fare ciò si ottiene un problema di ottimizzazione, ovvero si cerca di trovare i parametri θ della rete neurale che minimizzano la cost function. Quando si utilizza una rete neurale, la funzione $f(\theta, x_i)$ è una composizione di semplici unità chiamate neuroni e ogni neurone non è altro che una funzione di attivazione (*activation function*):

$$\sigma\left(\sum_{k=1}^n x_{ik} w_k + b\right) \quad (1.2)$$

la quale effettua la somma di tutti gli input x_{ik} moltiplicati per un peso w_k assegnato da un vettore di coefficienti \mathbf{w} sommando poi una costante b detta *bias*. Si possono usare

molteplici tipi di funzione di attivazione e le più famose sono la sigmoide (o *logistic function*) e la funzione ReLU (*Rectified Linear Unit*). Un esempio molto noto di Deep Feedforward Network sono le reti convoluzionali. Le reti convoluzionali o convolutional neural networks (CNNs) sono reti neurali specifiche che processano dei dati caratterizzati da una “topologia a griglia”, come ad esempio delle immagini che sono rappresentate da una griglia di pixel in due dimensioni. Il nome deriva dal fatto che la rete applica una operazione di convoluzione: nel contesto delle reti neurali, la convoluzione è un’operazione lineare che moltiplica l’input con un set di pesi. Dato che l’input è solitamente un’immagine in 2D, la convoluzione consiste nel prodotto tra un array di dati in input, e un una matrice di pesi di due dimensioni, chiamata filtro (o *filter*) oppure più comunemente kernel. Dato che il kernel è più piccolo dell’array in input, si considera una porzione o *patch* dell’input con le stesse dimensioni del kernel per poter applicare un prodotto scalare tra la patch e il kernel. Per ottenere le varie patch si fa spostare il kernel attraverso tutto l’input (*sliding kernel*) da sinistra verso destra e dall’alto verso il basso. Dato che il filtro o kernel è applicato sistematicamente a tutti i dati in input, il risultato in output viene definito *feature map*.

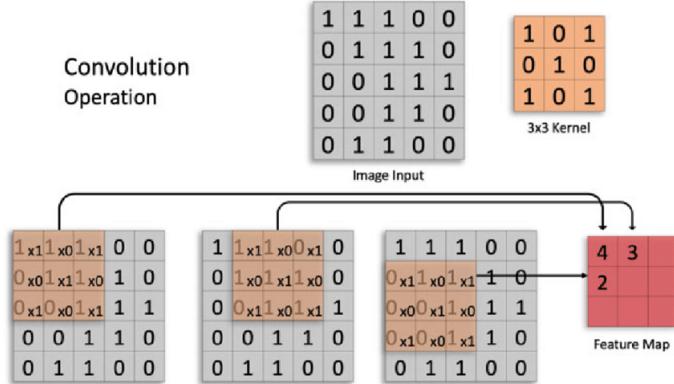


Figura 1.5: Un prodotto di convoluzione con un kernel di dimensione 3×3 . Il kernel si sposta con passo 1, cioè viene spostato un pixel alla volta per effettuare il prodotto di convoluzione. Si noti che la regione di sovrapposizione del kernel sulla matrice in input [8]

1.4.2 Layers più comuni delle Deep Feedforward Networks

- **Layer di input** Il layer di input è il primo layer della rete neurale e contiene i dati in input. Il numero di nodi in questo layer è uguale al numero di features del dataset. Questo layer è sempre presente e non viene mai modificato durante il training.
- **Layer Denso o Fully Connected (FC):** Ogni neurone di un layer FC è connesso a tutti i nodi dello layer precedente. Il numero di nodi in questo layer è un parametro che viene scelto dall’utente e può essere modificato durante il training.

- **Layer Convoluzionale:** Il layer convoluzionale è spesso utilizzato per processare delle immagini in 2D. Esso viene utilizzato per effettuare delle operazioni di convoluzione.
- **Layer di Pooling** Il layer di pooling è spesso utilizzato dopo un layer convolutional. Il suo scopo è quello di ridurre la dimensione dell'input, riducendo la complessità del modello. Il layer di pooling è composto da un insieme di nodi che prendono in input un insieme di nodi di un layer precedente e restituiscono un singolo nodo in output. Il pooling è un'operazione non lineare che prende in input un insieme di valori e restituisce un singolo valore. Il pooling più comune è il max pooling che prende in input un insieme di valori e restituisce il valore massimo.
- **Layer di Dropout** Il layer di dropout è spesso utilizzato dopo un layer fully connected. Il suo scopo è quello di ridurre la complessità del modello e questo è utile dato che come spiegato in precedenza, se l'algoritmo ha una capacità troppo elevata, si ottiene un comportamento di overfitting [9]. Il layer di dropout è composto da un insieme di nodi che prendono in input un insieme di nodi di un layer precedente e restituiscono un singolo nodo in output. Il dropout è un'operazione non lineare che prende in input un insieme di valori e restituisce un singolo valore. Il dropout più comune è il max dropout che prende in input un insieme di valori e restituisce il valore massimo. Esso si attiva solo durante la fase di training. Al layer di dropout si assegna un parametro chiamato *rate* che indica la probabilità che un neurone possa essere deattivato.
- **Layer di output** Il layer di output è l'ultimo layer della rete neurale e contiene i dati in output. Il numero di nodi in questo layer è uguale al numero di classi del dataset. Questo layer è sempre presente e non viene mai modificato durante il training.

1.4.3 Imbalanced Classification

In condizioni ottimali, un dataset presenta una classificazione omogenea, ovvero per ogni classe ci sono un numero circa uguale di elementi specifici di quella classe. Normalmente però il dataset originario contiene uno sbilanciamento tra le varie classi. Nel corso degli anni sono state proposte molte tecniche per risolvere questo problema, e una buona trattazione di queste tecniche è stata fatta da He, Haibo and Garcia et al in [10]. In questa tesi per risolvere il problema del dataset sbilanciato durante il training sono stati inseriti dei pesi calcolati con la funzione

1.4.4 Transfer Learning e Fine Tuning

Il Transfer Learning è un metodo di Machine Learning dove un modello di apprendimento che è stato sviluppato per un primo task specifico viene poi riutilizzato per un secondo task. In pratica, l'intento è quello di salvare varie caratteristiche apprese durante il primo apprendimento, come ad esempio i vari pesi della funzione di costo oppure della funzione di accuratezza, per poi eseguire il secondo training mantenendo le informazioni apprese;

in questo modo, il secondo training risulterà più veloce e quindi più efficace. La maggior parte delle soluzioni che usano il Transfer Learning consistono nell'allineare lo spazio di input del dataset della sorgente con quello del dataset dei target. Qualora il dataset del modello sorgente (ovvero del modello dove è stato realizzato il primo task di apprendimento) differisca di molto dal dataset del secondo task di apprendimento (dataset target), si può procedere con la tecnica del Fine Tuning; in questo modo, l'apprendimento diventa più efficiente grazie al Transfer Learning e si specializza sul dataset target grazie al Fine Tuning. Il Fine Tuning solitamente si compone dei seguenti quattro passaggi:

1. Si pre-addestra un modello di rete neurale, ovvero il modello di origine, su un dataset di origine (ad esempio il dataset ImageNet).
2. Si crea un nuovo modello di rete neurale, ovvero il modello di destinazione. Questo copia tutti parametri sul modello sorgente eccetto il livello di output. Si assume che questi parametri del modello contengano le conoscenze apprese dal dataset di origine e che queste conoscenze siano applicabile anche al dataset di destinazione. Si assume inoltre che il layer di output del modello di origine è strettamente correlato alle *label* del dataset di origine; quindi non viene utilizzato nel modello di destinazione.
3. Si aggiunge un layer di output al modello di destinazione, il cui numero di output è il numero di categorie nel dataset di destinazione. Quindi si inizializza in modo casuale i parametri del modello di questo layer.
4. Si addestra il modello di destinazione sul dataset di destinazione. Il livello di output sarà addestrato da zero, mentre i parametri di tutti gli altri livelli sono ottimizzati in base ai parametri del modello sorgente.

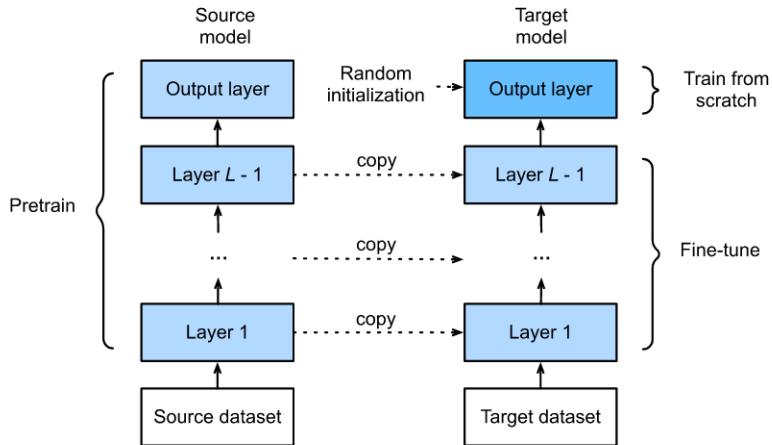


Figura 1.6: Schema dei passaggi per applicare la tecnica del Fine Tuning

Capitolo 2

Segmentazione tramite Deep Learning e creazione del dataset di immagini ecografiche

2.1 Segmentazione semantica multiclasse per il riconoscimento di oggetti

Il campo della Computer Vision ha, dal punto di vista delle scienze biologiche, l’obiettivo di elaborare modelli computazionali del sistema visivo umano, mentre, dal punto di vista ingegneristico, ha l’obiettivo di costruire sistemi autonomi che riproducano alcuni dei compiti che il sistema visivo umano riesce a performare. I due obiettivi sono strettamente legati dato che studiare le proprietà del sistema visivo umano dà spesso ispirazione agli ingegneri che progettano i sistemi autonomi di Computer Vision e, viceversa, gli algoritmi di visione artificiale possono offrire spunti su come funziona il sistema visivo umano [11]. Questi scopi sono alquanto complessi in quanto, a differenza di ciò che il sistema visivo umano compie, un sistema di Computer Vision deve affrontare un “problema inverso”: ovvero interpretare il mondo che gli umani vedono a partire da una o più immagini, valutandone le proprietà come, ad esempio, la forma, il grado di illuminazione e la distribuzione dei colori. È per questo motivo che questi sistemi devono basarsi su modelli della teoria della fisica o della probabilità oppure devono fare affidamento su tecniche di Machine Learning con grandi dataset [12].

Uno dei tipici task della Computer Vision è quello di determinare se una immagine in input contiene uno specifico oggetto, oppure una specifica caratteristica o attività. Le sottocategorie più comuni della recognition sono:

- **Image Classification:** il sistema cerca di comprendere un’intera immagine nel suo insieme, con l’obiettivo di classificarla assegnandola a un’etichetta specifica.

- **Object Detection:** consiste nell'individuare un oggetto all'interno dell'immagine, coinvolge sia la classificazione che la localizzazione e viene utilizzato per analizzare casi più realistici in cui possono esistere più oggetti in un'immagine [13].
- **Semantic Segmentation:** nella segmentazione semantica si vuole dividere un'immagine in regioni che appartengono a classi semantiche diverse. Diversamente dall'object detection, la segmentazione semantica riconosce e comprende cosa ci sono nelle immagini a livello di pixel, dunque si dice che la sua etichettatura e le predizioni sono a livello di pixel [14]. In base al numero di categorie, la segmentazione semantica può essere:
 - Binaria: si utilizzano due categorie dove in generale una rappresenta la categoria dell'oggetto che si vuole identificare, mentre l'altra rappresenta tutto il resto (il background).
 - Multi classe: si utilizzano più di due categorie per identificare più oggetti specifici all'interno dell'immagine.
- **Instance Segmentation:** è una estensione della segmentazione semantica nella quale l'obiettivo è di ottenere una visualizzazione degli oggetti della stessa classe suddivisi in istanze diverse. Quindi, la instance segmentation può essere definita come la tecnica per risolvere simultaneamente il problema della object detection così come quello della segmentazione semantica. Automatizzare questo processo non è facile, perché il numero di istanze non è noto in anticipo e la valutazione delle istanze ottenute non è basata sui pixel come avveniva con la segmentazione semantica.

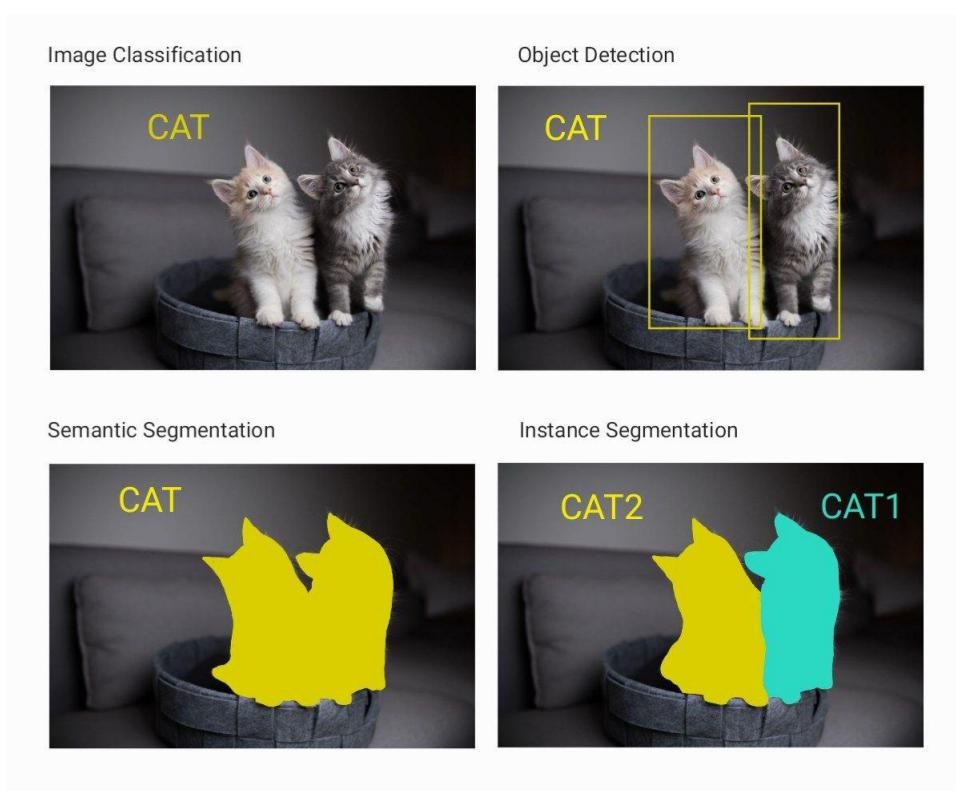


Figura 2.1: Task più comuni di recognition

2.2 Generazione del dataset

L'articolazione della mano è composta da 27 componenti ossee che suddividono mano e polso in tre compartimenti articolari (carpo, metacarpo e falangi) e cinque tratti articolari:

- Articolazioni del polso
- Articolazioni del carpo
- Articolazioni del metacarpo
- Articolazioni delle dita

In una normale articolazione i capi articolari sono contenuti in una cavità delimitata da una capsula articolare rivestita da una membrana, denominata membrana sinoviale. I capi articolari sono rivestiti da una superficie perfettamente liscia, la cartilagine articolare, che consente un libero scivolamento tra di loro. La cavità articolare viene “lubrificata” da un liquido, denominato liquido sinoviale, che fa sì che lo scivolamento avvenga in modo ottimale. Quando la cartilagine si consuma o si altera per fisiologica usura o abnorme utilizzo dell'articolazione (gesti ripetitivi in attività lavorative), o per traumi o fratture che interessano la superficie dei capi articolari, o per altre cause infiammatorie che agiscono sul liquido sinoviale come l'artrosi primaria o l'artrite reumatoide, l'articolazione colpita diventa spesso rigida e dolente modificando a volte anche l'asse del dito, che si deforma [15]. Dunque, qualora un paziente riferisca dolore alla mano con sede e caratteristiche non ben definite, viene consigliato l'esame ecografico, dato che esso permette di visualizzare tessuti che con esami differenti, quali la radiografia oppure la risonanza magnetica, non si potrebbero analizzare. L'ecografia è un sistema di indagine diagnostica medica che utilizza ultrasuoni e si basa sul principio dell'emissione di eco e della trasmissione delle onde ultrasonore. In particolare, ogni ecografia viene realizzata da una sonda mantenuta a contatto con la pelle del paziente grazie a del gel. La sonda emette un ultrasuono con una frequenza decisa dall'operatore, tenendo conto che una frequenza maggiore permette di ottenere una maggiore risoluzione dell'immagine, ma una visione dei layers più superficiali dell'articolazione. L'onda emessa dall'ecografo, quando incontra un tessuto osseo non riesce a penetrare più in profondità, bensì rimbalza. La stessa sonda è quindi in grado di catturare il segnale di ritorno che viene opportunamente elaborato da un computer e presentato su un monitor. È per questo che l'ecografia è, in ogni caso, una procedura operatore-dipendente, poiché vengono richieste particolari doti di manualità e di osservazione, oltre a cultura dell'immagine ed esperienza clinica.

Prima di iniziare a definire il modello di Machine Learning che si vuole utilizzare è importante analizzare i dati che si hanno a disposizione. Nel caso di un task di segmentazione semantica, è utile valutare il livello di illuminazione e di contrasto delle immagini, la loro risoluzione e la presenza di artefatti. Ad esempio, è utile decidere se ritagliare le immagini per far risaltare meglio gli oggetti da classificare: infatti ritagliando l'immagine si diminuisce la quantità di pixel di una o più classi spesso meno importanti, come ad esempio

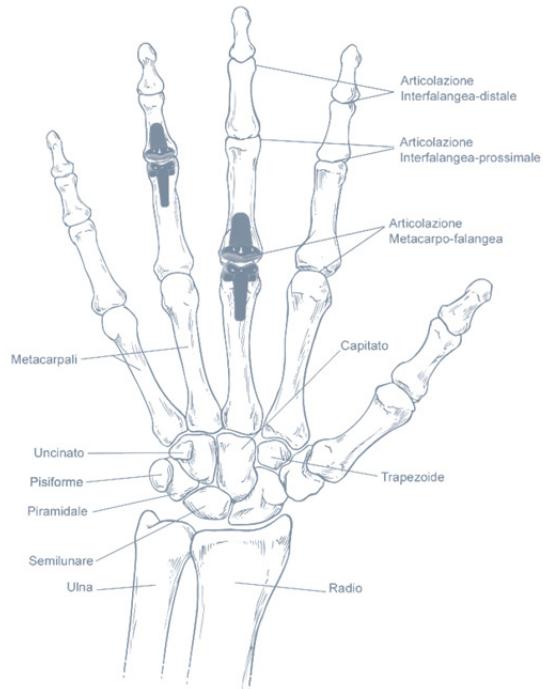


Figura 2.2: Anatomia della mano umana [15]

il background. Un'altra importante caratteristica da considerare è il numero di immagini: se esso è molto piccolo, è spesso opportuno applicare delle tecniche di data augmentation per aumentare il numero di immagini. Il dataset disponibile è composto da 60 immagini figuranti sezioni sagittali del metacarpo, realizzate dalla dottoressa Luana Mancarella presso l'Istituto Ortopedico Rizzoli di Bologna. Come mostrato in figura, all'interno delle immagini si possono distinguere la presenza dell'osso, del tendine, della cavità sinoviale e della cartilagine articolare.

Per annotare le immagini è stato utilizzato il software Apeer¹, che permette di disegnare le maschere di una immagine caricata e esportare l'annotazione in formato tiff. Apeer si presenta come una web application supportata dai browser più comuni ed è facilmente utilizzabile grazie a una dashboard intuitiva e a una documentazione completa. Per semplicità di implementazione le immagini sono state convertite in formato png e sono stati realizzati tre dataset diversi:

- dataset osso-tendine
- dataset sinovia-tendine
- dataset sinovia-osso

¹link to Apeer web application: <https://www.Apeer.com/app>

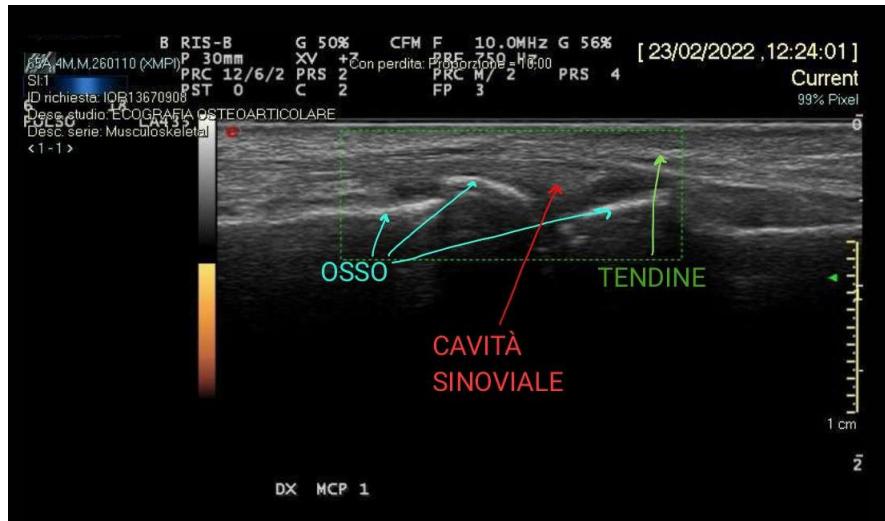


Figura 2.3: Elementi all'interno di una generica ecografia del dataset

Ad ogni ecografia viene associata una sola immagine, chiamata maschera, che presenta il risultato che si vuole far ottenere alla macchina. Di seguito si riportano alcuni esempi di associazioni tra un'ecografia e la sua maschera.

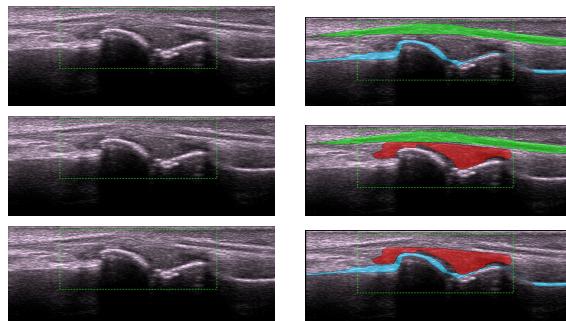
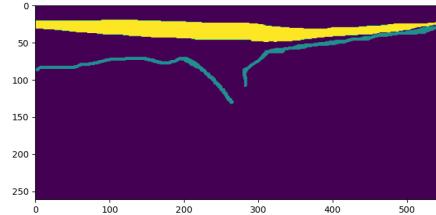


Figura 2.4: A sinistra le immagini ecografiche del metacarpo raccolte e ritagliate. A destra i disegni delle maschere associate ad ognuna.

Nella tabella 2.1 vengono riportati i dettagli che descrivono il processo di raccolta e annotazione delle immagini.

Il dataset risulta essere sbilanciato, ovvero ogni immagine risulta avere una o più classi il quale numero di pixel è molto inferiore rispetto alle altre: si veda per questo la Sezione 1.4.3. Considerando il dataset delle ecografie del Rizzoli, nella maschera seguente: il numero di pixel classificati come background è notevolmente superiore rispetto al numero di pixel delle altre due classi. In particolare, il background occupa circa il 90% del totale dei pixel,

Descrizione	Valore
Numero di immagini raccolte	2148
Numero di pazienti analizzati	80
Numero di immagini del metacarpo	620
Numero di immagini annotate	60

Tabella 2.1: Tabella dei dettagli che descrivono la formazione del dataset**Figura 2.5:** Esempio di maschera delle ecografie dell'Istituto Ortopedico Rizzoli che mostra un dataset sbilanciato.

l'osso il 5% e il tendine il 5%. L'Imbalanced Classification può creare problemi durante il training: nell'esempio precedente, se si allena la macchina senza adoperare tecniche di risoluzione della Imbalanced Classification, si otterranno delle predizioni tutte background e con il valore della funzione di accuratezza molto alto; infatti, se la macchina predice che tutti i pixel sono “background” allora otterrà un valore complessivo dell'accuratezza del 90%. Di seguito vengono riportate delle informazioni tecniche più dettagliate sulle immagini di ogni dataset:

Dataset osso-tendine	Background	Osso	Tendine	Totale
N. classe	0	1	2	
N. immagini	48	48	48	48
Tot immagini per il train set	12	12	12	12
Tot immagini per il validation test	12	12	12	12
Tot immagini per il test set	3	3	3	3
Tot pixel di una immagine esempio	46710	1248	2218	50176
Numero di pixel medio nel dataset	2770454	50380	189726	3010560
% media di pixel per immagine	92.03%	1.67%	6.30%	100.00%
Pesi inseriti per ogni classe	0.36	19.92	5.29	

Tabella 2.2: Dettagli della composizione del dataset osso-tendine

Dataset sinovia-osso	Background	Sinovia	Osso	Totale
N. classe	0	1	2	
N. immagini	48	48	48	48
Tot immagini per il train set	12	12	12	12
Tot immagini per il validation test	12	12	12	12
Tot immagini per il test set	3	3	3	3
Tot pixel di una immagine esempio	45342	3854	980	50176
Numero di pixel medio nel dataset	2646475	116783	46598	3010560
% media di pixel per immagine	87.9%	3.88%	1.55%	100.00%
Pesi inseriti per ogni classe	0.35	8.02	20.1	

Tabella 2.3: Dettagli della composizione del dataset sinovia-osso

Dataset sinovia-tendine	Background	Sinovia	Tendine	Totale
N. classe	0	1	2	
N. immagini	48	48	48	48
Tot immagini per il train set	12	12	12	12
Tot immagini per il validation test	12	12	12	12
Tot immagini per il test set	3	3	3	3
Tot pixel di una immagine esempio	41045	3854	5277	50176
Numero di pixel medio nel dataset	2468047	115217	176416	3010560
% media di pixel per immagine	82.00%	3.83%	5.86%	100.00%
Pesi inseriti per ogni classe	0.37	7.98	5.21	

Tabella 2.4: Dettagli della composizione del dataset sinovia-tendine

2.3 Rete U-Net

Quando si effettua della segmentazione semantica bisogna scegliere con cura la tipologie di rete sulla base del dataset considerato. Ad esempio, la U-net è una rete completamente convolutiva per la segmentazione delle immagini. Fu sviluppata in origine per la segmentazione delle immagini biomeditica, ma può essere utilizzata per qualsiasi tipo di attività di segmentazione dell'immagine. La rete si basa sull'architettura encoder-decoder con connessioni tra i livelli encoder e decoder corrispondenti. La rete è addestrata end-to-end e può essere utilizzata per attività di segmentazione delle immagini con pochissima pre e post-elaborazione. Si veda la Sezione 3.3 per maggiori dettagli su questa tipologia di rete.

Capitolo 3

Implementazione

3.1 Interfaccia di utilizzo

Il software può essere utilizzato tramite un’interfaccia a linea di comando. L’interfaccia è stata realizzata con l’ausilio di *argparse*, una libreria in Python con la quale lo sviluppatore definisce un programma e gli argomenti richiesti e sarà poi argparse ad analizzarli da sys.argv, ovvero l’elenco degli argomenti della riga di comando passati a uno script Python. Il modulo argparse genera inoltre automaticamente messaggi di aiuto e di utilizzo ed emette errori quando gli utenti forniscono argomenti non validi al programma. Per lanciare l’applicazione è necessario eseguire il comando:

```
python3 main.py
```

L’interfaccia di utilizzo è composta da due programmi:

- **Train and predict** che permette di addestrare una rete neurale e di eseguire un test di predizioni
- **Only predict** che permette di eseguire un test di predizioni su una rete neurale già addestrata

Per ognuno dei programmi possono essere specificati vari parametri o opzioni come mostrato in figura.

```
> python3 main.py -TP --help
usage: TP [options]

optional arguments:
-h, --help            show this help message and exit
-TP, --train-and-predict
                      choose train and predict program
-P, --only-predict  choose only predict program
-v, --version         show program's version number and exit
-e EPOCHS, --epochs EPOCHS
                      number of epochs to train a model
-pat PATIENCE, --patience PATIENCE
                      number of patience for earlystopping callback during fit
-mon MONITOR, --monitor MONITOR
                      Metric to monitor for earlystopping and modelcheckpoint callbacks during fit
-c [COMMENT], --comment [COMMENT]
                      general comment printed to log file
-d DIR_DATASET, --dir-dataset DIR_DATASET
                      absolute directory path where dataset is stored
-m {UNET,TRANSFER_LEARNING_VGG16,TRANSFER_LEARNING_VGG19}, --model-name {UNET,TRANSFER_LEARNING_VGG16,TRANSFER_LEARNING_VGG19}
                      name of the model to train
```

Figura 3.1: Opzioni e parametri del programma “Train and predict”

```
> python3 main.py -P --help
usage: P [options]

optional arguments:
-h, --help            show this help message and exit
-TP, --train-and-predict
                      choose train and predict program
-P, --only-predict  choose only predict program
-v, --version         show program's version number and exit
-mw MODEL_WEIGHTS, --model-weights MODEL_WEIGHTS
                      absolute file path where weights of a model are saved
-c [COMMENT], --comment [COMMENT]
                      general comment printed to log file
-m {UNET,TRANSFER_LEARNING_VGG16,TRANSFER_LEARNING_VGG19}, --model-name {UNET,TRANSFER_LEARNING_VGG16,TRANSFER_LEARNING_VGG19}
                      name of the model to load
-d DIR_DATASET, --dir-dataset DIR_DATASET
                      absolute directory path where dataset is stored
```

Figura 3.2: Opzioni e parametri del programma “Only Predict”

3.2 Descrizione e caricamento del dataset

Le ecografie usate in questa tesi di laurea sono presenti nel database dell'Istituto Ortopedico Rizzoli in formato DICOM. Il formato DICOM è un formato standard per lo scambio di immagini mediche ed è spesso scelto per la sua diffusione e per la sua capacità di contenere informazioni aggiuntive, come ad esempio la data di acquisizione dell'immagine, il nome del paziente, il nome del medico che ha effettuato l'ecografia, il nome del medico che ha effettuato la diagnosi, ecc. Anche se esiste il modulo *pydicom* che permette di leggere file DICOM e di accedere alle informazioni contenute all'interno, per semplicità di implementazione in questa tesi è stato scelto di esportare le immagini in formato tiff dal database per poi convertirle in png. Una volta raccolte le immagini e definito il formato, è stato necessario effettuare un'analisi del dataset per capire come procedere con il *pre-processing*. Dato che le classi di segmentazione contengono elementi in maniera sbilanciata, è stato deciso di ritagliare le immagini di modo da diminuire il numero di pixel della classe più presente (il background), effettuando così del “downsampling”. Le immagini sono state salvate già in RGB ed è stato mantenuto questo formato dei colori dato che le reti neurali utilizzate in questa tesi di laurea sono state implementate con la libreria “Keras” [16] basata su Tensorflow [17] che supporta l'input di immagini con tre canali, come ad esempio l'RGB. Successivamente, sono state create le maschere associate, tramite l'applicazione web Apeer. Le maschere esportate da Apeer sono già normalizzate e in particolare ogni pixel è rappresentato da un valore intero compreso tra 0 e N-1, dove N è il numero di classi. Siccome i dataset creati hanno 3 classi, ogni pixel può assumere uno dei seguenti valori: 0, 1, 2. Ad esempio, nel dataset osso-tendine il valore 0 rappresenta il background, il valore 1 rappresenta l'osso e il valore 2 rappresenta il tendine. Si ottiene dunque un dataset formato da 60 immagini di dimensione 547x261 pixel e per ogni immagine è associata una sola maschera della stessa dimensione. Per caricare il dataset è stata utilizzata la libreria OpenCV [18] e il suo modulo *cv2* in Python, che permette di caricare le immagini e le maschere associate in formato png e di convertirle in array *numpy*. I numpy sono array multidimensionali che permettono di rappresentare matrici e vettori. Al momento del caricamento delle immagini, viene effettuato un ridimensionamento delle immagini e delle maschere in modo da avere tutte le immagini e le maschere con la stessa dimensione di 224x224, necessaria per l'input layer della rete neurale VGG.

3.3 Creazione del modello - Reti VGG

Nell'ambito del Machine Learning bisogna adoperare una o più reti neurali. Come definito in precedenza, una rete neurale è un modello matematico composto da “neuroni” artificiali che mira a somigliare ad una rete neurale biologica e per quegli obiettivi che lavorano su immagini si utilizzano maggiormente le reti neurali convoluzionali. La rete U-Net[19] è stata sviluppata da Olaf Ronneberger et al. con l'obiettivo di rendere più efficace la segmentazione di immagini biomediche. In particolare, la sua architettura è composta da due parti: la prima è chiamata encoder o contraction path e serve a localizzare caratteristiche significative

di contesto dell'immagine via via che se ne diminuisce la risoluzione; la seconda parte è chiamata decoder o expanding path e serve a localizzare in maniera più precisa i dettagli dell'immagine man mano che se ne aumenta la risoluzione. Dato che il decoder è applicato successivamente alla parte dell'encoder e dato che solitamente i layers dell'encoder sono simmetrici a quelli del decoder, l'architettura assume una forma ad "U" da cui deriva la sua denominazione.

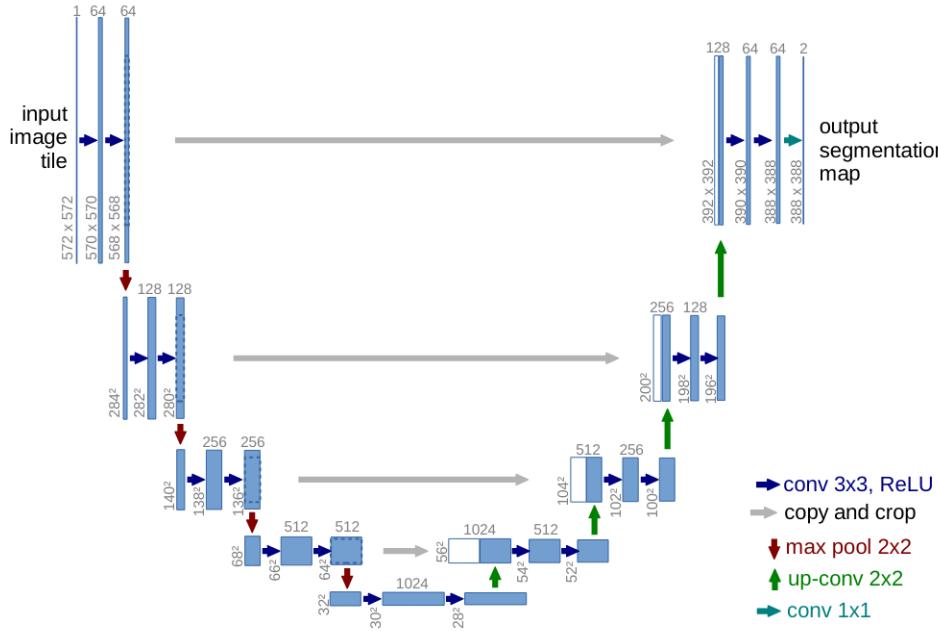


Figura 3.3: Architettura U-net (esempio per 32x32 pixel con la risoluzione più bassa). Ogni rettangolo blu corrisponde a una mappa delle caratteristiche multicanale. Viene indicato il numero di canali sopra il rettangolo. La dimensione x-y è fornita sul bordo inferiore sinistro del rettangolo. I rettangoli bianchi rappresentano le feature maps. Le frecce indicano le diverse operazioni come specificato in legenda. [19]

3.3.1 Rete VGG

Un blocco VGG è costituito da una sequenza di convoluzioni con un kernel di dimensione 3×3 e padding di 1 (mantenendo altezza e larghezza) seguita da un layer di *max-pooling* di dimensione 2×2 con *stride* di 2 (dimezzando altezza e larghezza dopo ogni blocco) [14]. La rete VGG può essere suddivisa in due parti: la prima costituita principalmente da layers convoluzionali e di pooling e la seconda costituita da layers fully connected. I layers convoluzionali sono raggruppati in trasformazioni non lineari che lasciano inalterata la dimensionalità, seguite da una riduzione della risoluzione, come illustrato in figura. La rete

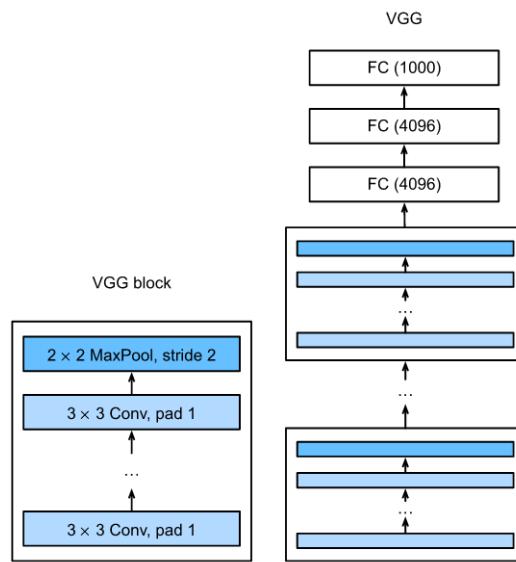


Figura 3.4: Schema a blocchi di una rete VGG [14]

VGG originale aveva 5 blocchi convoluzionali, tra i quali i primi due hanno uno layer convoluzionale ciascuno e gli ultimi tre contengono due layers convoluzionali ciascuno. Il primo blocco ha 64 canali di uscita e ogni blocco successivo raddoppia il numero di canali di uscita, finché quel numero non raggiunge 512. Poiché questa rete utilizza 8 layers convoluzionali e 3 layers fully connected, è spesso chiamato VGG-11 [14]. La parte di convoluzione della rete collega diversi blocchi VGG in successione e questo raggruppamento di convoluzioni è uno schema che è rimasta pressoché invariata nell'ultimo decennio, nonostante la specifica scelta di operazioni ha subito notevoli modifiche. In quanto tale, con il termine "VGG" si definisce una famiglia di reti piuttosto che solo una manifestazione specifica; a tal proposito nel paper di Simonyan e Zisserman (2014) [20] vengono descritte diverse varianti di VGG. I modelli utilizzati in questa tesi di laurea utilizzano l'architettura di rete VGG [20]. Creata da Karen Simonyan e Andrew Zisserman nel 2014, la rete VGG è una rete neurale convoluzionale che ha ottenuto buoni risultati in diversi campi, come ad esempio la classificazione di immagini. In particolare, la rete VGG rispecchia il concetto di profondità: i primi layer sono di dimensioni più grandi e presentano un numero piccolo di kernel, man mano che la

rete procede il numero di kernel aumenta, mentre le dimensioni diminuiscono. In questa tesi sono state utilizzate la rete VGG-16 e la VGG-19 che differiscono per il numero di layer utilizzati: 16 e 19 rispettivamente. Per creare ogni modello è stata utilizzata la libreria *Keras* che permette di creare reti neurali in modo semplice e veloce. Dato che in questa tesi è stata utilizzata per la segmentazione su delle immagini mediche, è stato necessario modificare l'architettura della rete neurale VGG16, in particolare è stato necessario rimuovere i *top layer*, ovvero i layer fully connected, e aggiungere un blocco decoder simmetrico rispetto al blocco encoder. Dunque in questo modo l'architettura presenta la forma a “U” della U-net. Al momento del caricamento del modello, sono stati caricati i pesi pre-addestrati su ImageNet [21], che è un dataset di 14 milioni di immagini suddivise in 1000 classi.

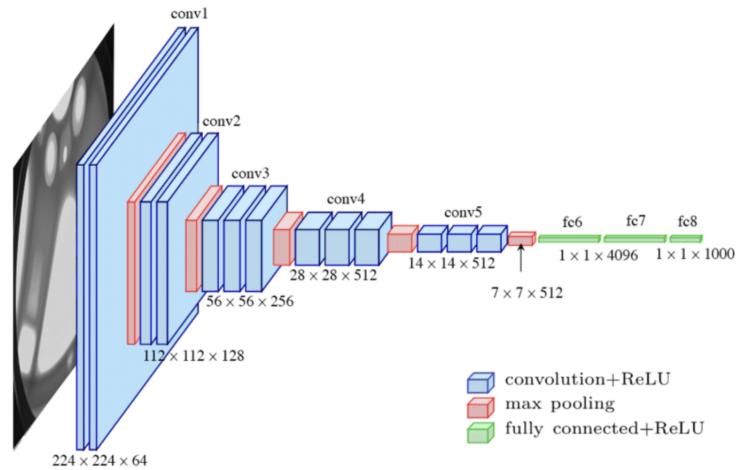


Figura 3.5: Architettura della rete neurale VGG-16 utilizzata per la classificazione di immagini ¹

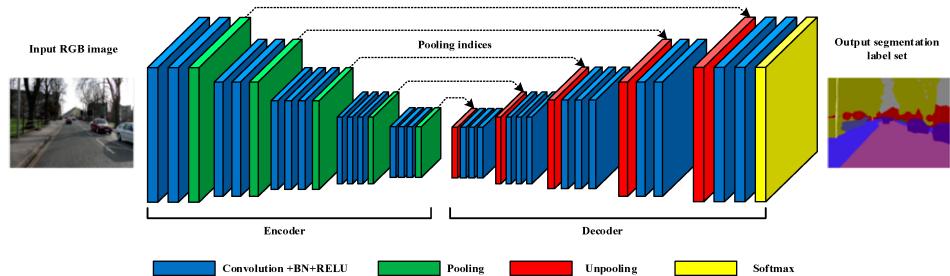


Figura 3.6: Architettura della rete neurale VGG16 utilizzata in questa tesi per la segmentazione semantica ²

¹Image link: <https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models-96e4bf398484>

²Image link: <https://wikidocs.net/164366>

3.4 Training

Per la compilazione del modello, è stato deciso di utilizzare l'optimizer Adam [22] che è un ottimizzatore che utilizza la tecnica di “momentum” per accelerare il processo di addestramento. Inoltre come loss function è stato deciso di utilizzare la Categorical Crossentropy, che è una funzione di loss che viene utilizzata per la classificazione multiclasse. Per la fase di training è stato utilizzato il dataset creato precedentemente, che è stato suddiviso in due parti: 80% di immagini per il training e 20% immagini per il validation. Sempre durante la compilazione del modello, devono essere specificate anche le metriche da utilizzare per valutare l'andamento del training, in particolare:

- **Accuracy:** metrica che viene utilizzata in generale per la classificazione e viene definita come:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Essa rappresenta la somma dei valori predetti correttamente diviso la somma di tutti i valori. È una metrica che dà un risultato intuitivo, ma non è molto utile quando si ha un dataset sbilanciato, come nel caso preso in esame.

- **IoU:** l'Intersection Over Union è l'indice di intersezione sovrapposizione, che è una metrica che viene utilizzata per la segmentazione semantica binaria. Viene calcolata come segue:

$$IoU = \frac{TP}{TP + FP + FN} \quad (3.2)$$

dove TP è il numero di pixel correttamente classificati, FP è il numero di pixel falsamente classificati come positivi e FN è il numero di pixel falsamente classificati come negativi. Dato che in questo caso si utilizza la segmentazione semantica multiclasse, l'indice di intersezione sovrapposizione viene calcolato per ogni classe.

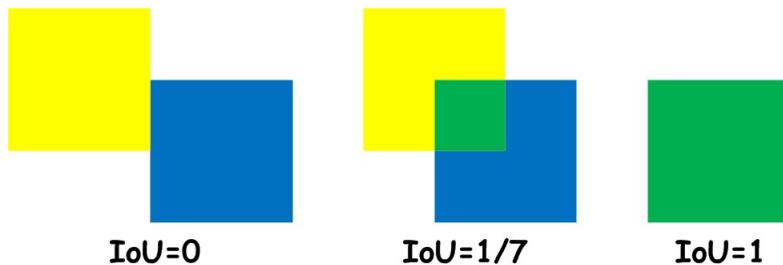


Figura 3.7: Rappresentazione grafica della Intersection Over Union ³

³Image link: <https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation-model-8b22e2e84686>

- **MeanIoU:** metrica che viene utilizzata per la segmentazione semantica e viene calcolata come:

$$MeanIoU = \frac{1}{N} \sum_{i=1}^N IoU_i \quad (3.3)$$

dove N è il numero di classi.

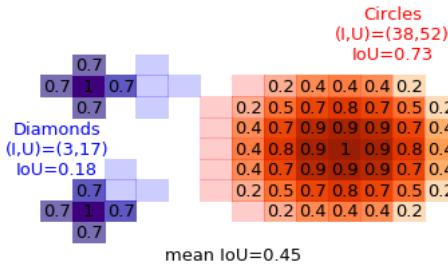


Figura 3.8: Rappresentazione grafica della MeanIoU ⁴

- **Precision:** è una metrica che viene utilizzata per la classificazione multiclass e è definita come:

$$Precision = \frac{TP}{TP + FP} \quad (3.4)$$

Essa rappresenta la somma dei valori predetti correttamente diviso la somma di tutti i valori predetti come positivi. Qualora la questa metrica debba essere utilizzata per un task multiclass, si può calcolare una precision media e in base alla tipologia di media eseguita sui dati si distingue tra:

- **Macro Average Recall:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media non ponderata. Ciò non tiene conto dello squilibrio dell'etichetta, dunque questa metrica non è così significativa qualora si utilizzi un dataset sbilanciato [23].
- **Weighted Average Recall:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media ponderata in base al supporto (il numero di istanze vere per ciascuna etichetta). Ciò è diverso dalla “macro” perch è tiene conto dello squilibrio dell'etichetta [23].
- **Recall o Sensitivity:** è una metrica che viene utilizzata per la classificazione multiclass ed è definita come:

$$Recall = \frac{TP}{TP + FN} \quad (3.5)$$

⁴Image link: <https://il monteux.github.io/2019/05/10/segmentation-metrics.html>

Essa rappresenta la somma dei valori predetti correttamente diviso la somma di tutti i valori che sono veramente positivi. Qualora la questa metrica debba essere utilizzata per un task multiclasse, si può calcolare una precision media e in base alla tipologia di media eseguita sui dati si distingue tra:

- **Macro Average Precision:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media non ponderata. Ciò non tiene conto dello squilibrio dell’etichetta, dunque questa metrica non è così significativa qualora si utilizzi un dataset sbilanciato [23].
- **Weighted Average Precision:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media ponderata in base al supporto (il numero di istanze vere per ciascuna etichetta). Ciò è diverso dalla “macro” perch è tiene conto dello squilibrio dell’etichetta [23].

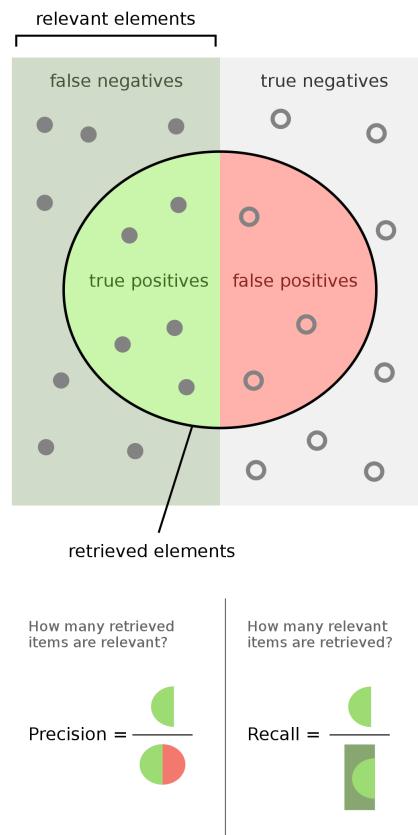


Figura 3.9: Rappresentazione grafica della Precision e della Recall ⁵

⁵Image link: https://en.wikipedia.org/wiki/Precision_and_recall

- **AUC on PR curve:** Precision e recall sono utili misure del successo della previsione quando le classi sono molto sbilanciate. La precisione è una misura della pertinenza dei risultati, mentre la recall è una misura di quanti risultati che vengono restituiti sono veramente rilevanti. La PR curve mostra il compromesso tra precisione e recall per diverse soglie. Viene soprattutto considerata l'area sotto la curva (AUC, ovvero Area Under Curve), che è una misura di quanto il modello è in grado di distinguere tra le classi. Un'area grande sotto la curva rappresenta sia una recall elevata che un'alta precisione, dove un'alta precisione si riferisce a un basso tasso di falsi positivi e una recall alta si riferisce a un basso tasso di falsi negativi. Punteggi elevati per entrambi mostrano che il classificatore restituisce risultati accurati (alta precisione), oltre a restituire la maggior parte di tutti i risultati positivi (recall elevata). Un sistema con una recall elevata ma una precisione bassa restituisce molti risultati, ma la maggior parte delle etichette predette non sono corrette rispetto alle etichette vere. Un sistema con alta precisione ma bassa recall è esattamente l'opposto, poiché restituisce pochissimi risultati, ma la maggior parte delle etichette predette sono corrette rispetto alle etichette vere. Un sistema ideale con alta precisione e alto richiamo restituirà molti risultati, con tutti i risultati etichettati correttamente [23].

- **F1 score:** è una metrica che viene utilizzata per la classificazione multiclasse ed è definita come:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3.6)$$

Essa rappresenta la media armonica tra la precisione e la recall. Dato che precisione e recall sono calcolate in maniera binaria, allora si ottiene una F1 score per ogni classe. Qualora la questa metrica debba essere utilizzata per un task multiclass, si può calcolare una F1-score media e in base alla tipologia di media eseguita sui dati si distingue tra:

- **Macro Average F1-score:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media non ponderata. Ciò non tiene conto dello squilibrio dell'etichetta, dunque questa metrica non è così significativa qualora si utilizzi un dataset sbilanciato [23].
- **Weighted Average F1-score:** viene calcolata la F1-score per ciascuna etichetta e si trova la loro media ponderata in base al supporto (il numero di istanze vere per ciascuna etichetta). Ciò è diverso dalla “macro” perch è tiene conto dello squilibrio dell'etichetta; può risultare in una F1-score che non è compresa tra la precisione e la recall associate [23].

Dato che viene utilizzata la tecnica del Fine Tuning, il training è suddiviso in due fasi: la prima fase è la fase di training del modello pre-addestrato, mentre la seconda fase è la fase di training del modello fine-tuned. Si veda la sezione dedicata al Fine Tuning per maggiori dettagli. Durante il training, grazie all'API di Tensorflow *ModelCheckpoint*, vengono salvati i pesi del modello che raggiungono il miglior valore di una variabile chiamata “monitor”

in modo da poterli utilizzare per il programma di solo test (only predict). La variabile viene assegnata come parametro in ingresso dall'utente, in base alla metrica che egli vuole minimizzare o massimizzare. La variabile monitor viene usata anche per l'*EarlyStopping*, che permette di interrompere il training se la metrica monitor non migliora per un numero di epocha specificato dall'utente. Tutti i parametri e i risultati del training vengono poi salvati in un file .json, che viene utilizzato per il test del modello.

3.5 Predizioni

Vengono realizzate le predizioni sul dataset di test, utilizzando il modello fine-tuned. Successivamente, per valutare la qualità delle predizioni, vengono calcolate delle metriche definite nella sezione precedente comprendendo anche:

- **Multiclass Confusion Matrix:** è una matrice che mostra la frequenza con cui i dati sono stati classificati in una classe diversa da quella corretta. In altre parole, è una buona rappresentazione grafica delle predizioni TP, TN, e FN. La matrice presenta sull'asse orizzontale le classi predette e sull'asse verticale le classi reali.
- **Classification Report:** è una tabella che mostra la precisione, la recall, il f1-score e il support per ogni classe, fornita dalla libreria di Sklearn [23].

Grazie all'utilizzo del modulo *pyplot* di *matplotlib* [24] vengono salvate le immagini predette, così come i grafici delle metriche di training, di validation e di test.

Capitolo 4

Risultati dei test

Sono stati effettuati dei test utilizzando delle immagini diverse da quelle del train-set di modo da valutare in maniera effettiva l'algoritmo. Infatti, se le predizioni fossero fatte con le stesse immagini date in input al training, allora si otterrebbe un comportamento di overfitting o sovradattamento. Ciò si verifica quando il modello è troppo legato alle caratteristiche di dati di input nella fase di training e quindi performa relativamente molto bene con quei dati, ma non con dei nuovi dati durante la fase di test.

I test sono stati realizzati tutti con la stessa macchina fisica presente nel laboratorio del CASY, ovvero una workstation con un processore Intel i9 octa-core, 64 GB RAM e 2 NVIDIA QUADRO P4000 GPU. In ogni training, a tempo di esecuzione, tramite la libreria di Tensorflow, è stato diviso il dataset in 3 parti. Per tutti e tre i dataset:

- train-set: 48 immagini + 48 maschere
- validation-set: 12 immagini + 12 maschere
- test-set: 3 immagini + 3 maschere

I parametri principali di configurazione di ogni training sono:

- N. epoche: 160 epoche per il training e 40 per il fine tuning
- Dimensione dell'immagine (224,224)
- Numero di classi = 3
- Dimensione dei batch = 8
- Algoritmo di ottimizzazione = “Adam”
- Funzione di costo (o loss function) = “Categorical Crossentropy”

Per la valutazione del training riporto i grafici della funzione di accuratezza e della funzione di costo dove i valori delle due funzioni sono riferiti ad ogni epoca del training. Invece

per la valutazione del comportamento della rete neurale durante il test riporto i grafici della matrice di confusione e del report di classificazione con le metriche descritte al Capitolo 3.

4.1 Predizioni con rete VGG16 U-Net in Transfer Learning e Fine Tuning

Predizioni del dataset osso-tendine

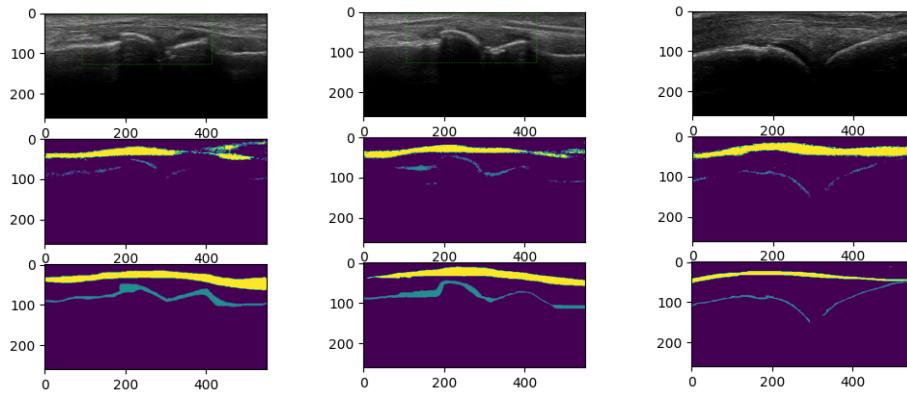


Figura 4.1: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset osso-tendine

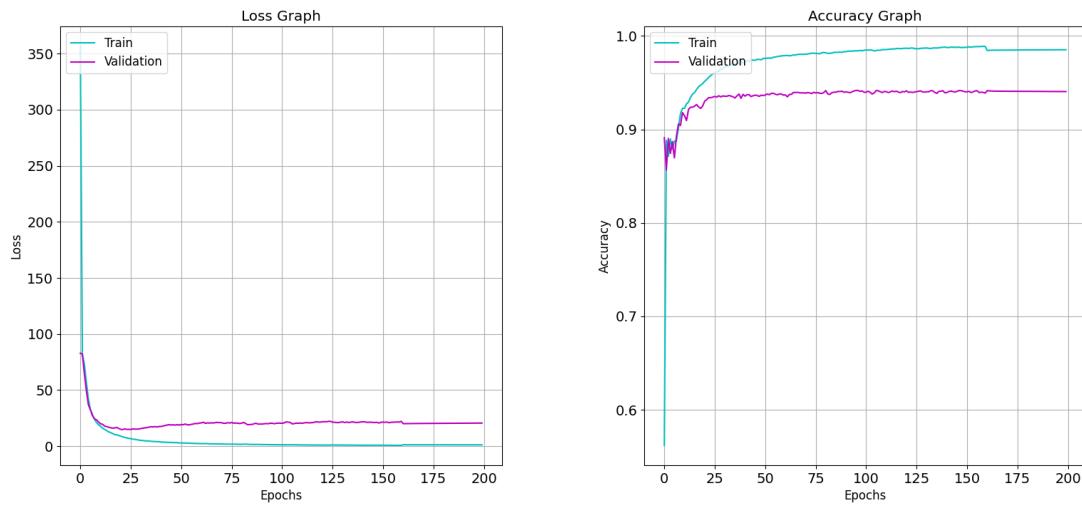


Figura 4.2: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset osso-tendine

	precision	recall	f1-score
Background	95%	97%	96%
Osso	68%	19%	30%
Tendine	56%	59%	58%
accuracy	93%	93%	93%
macro avg	73%	58%	61%
weighted avg	93%	93%	92%

Tabella 4.1: Tabella del report di classificazione per le predizioni con la rete VGG16 sul dataset osso-tendine

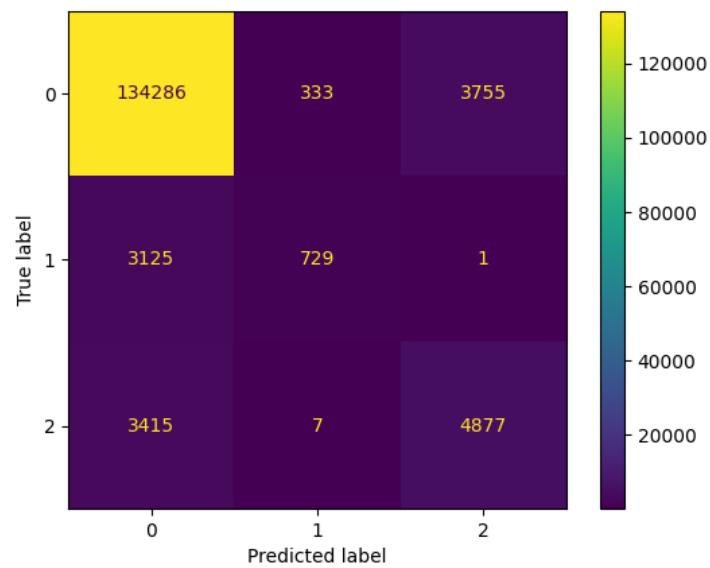


Figura 4.3: Grafico della matrice di confusione multiclass per le predizioni con la rete VGG16 sul dataset osso-tendine

Predizioni del dataset sinovia-osso

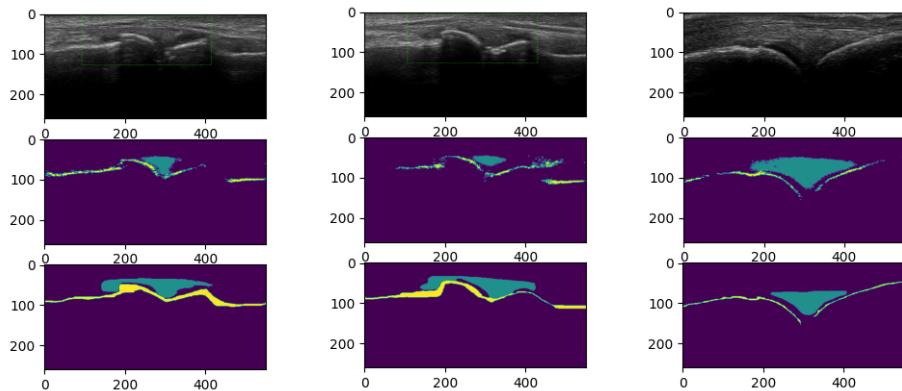


Figura 4.4: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset sinovia-osso

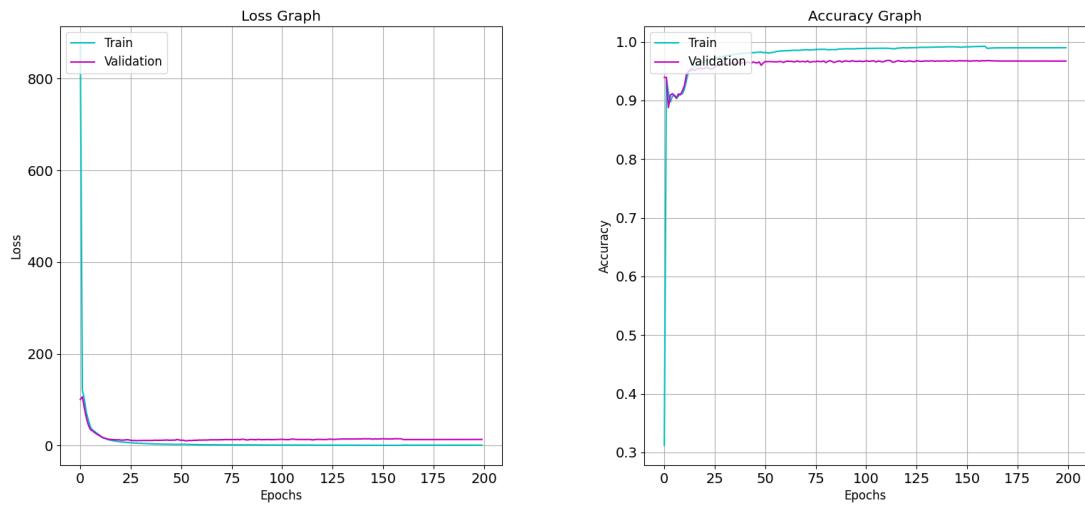


Figura 4.5: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset sinovia-osso

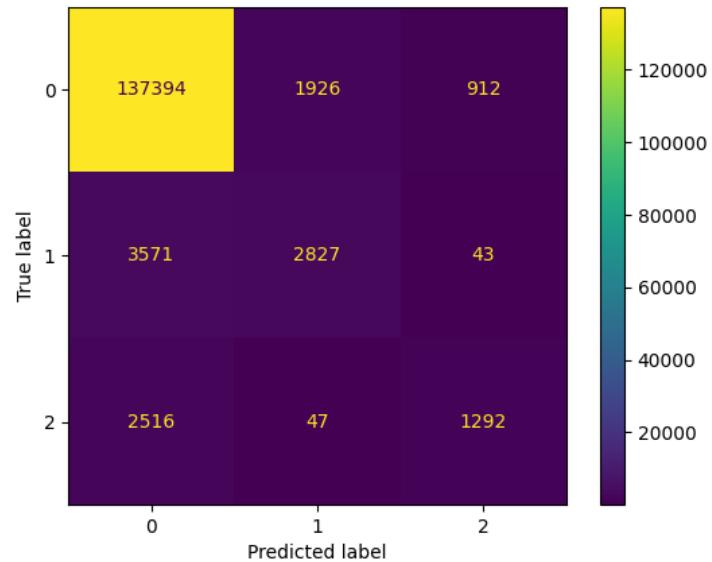


Figura 4.6: Grafico della matrice di confusione multiclasse per le predizioni con la rete VGG16 sul dataset sinovia-osso

	precision	recall	f1-score
Background	96%	98%	97%
Sinovia	59%	44%	50%
Osso	57%	34%	42%
accuracy	94%	94%	94%
macro avg	71%	58%	63%
weighted avg	93%	94%	93%

Tabella 4.2: Tabella del report di classificazione per le predizioni con la rete VGG16 sul dataset sinovia-osso

Predizioni del dataset sinovia-tendine

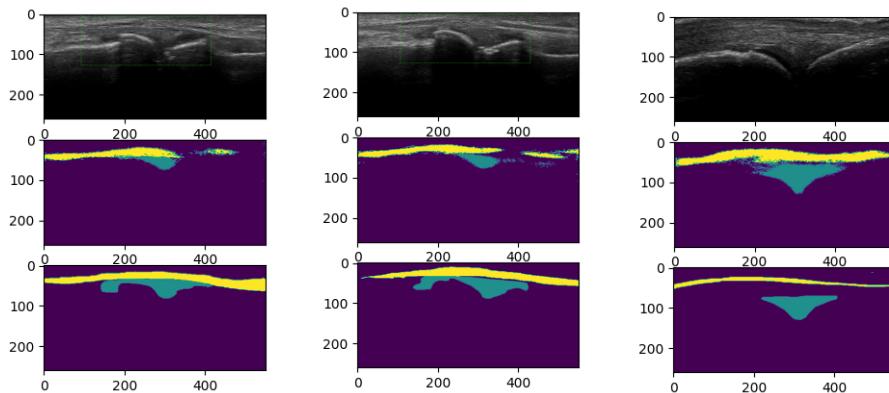


Figura 4.7: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset sinovia-tendine

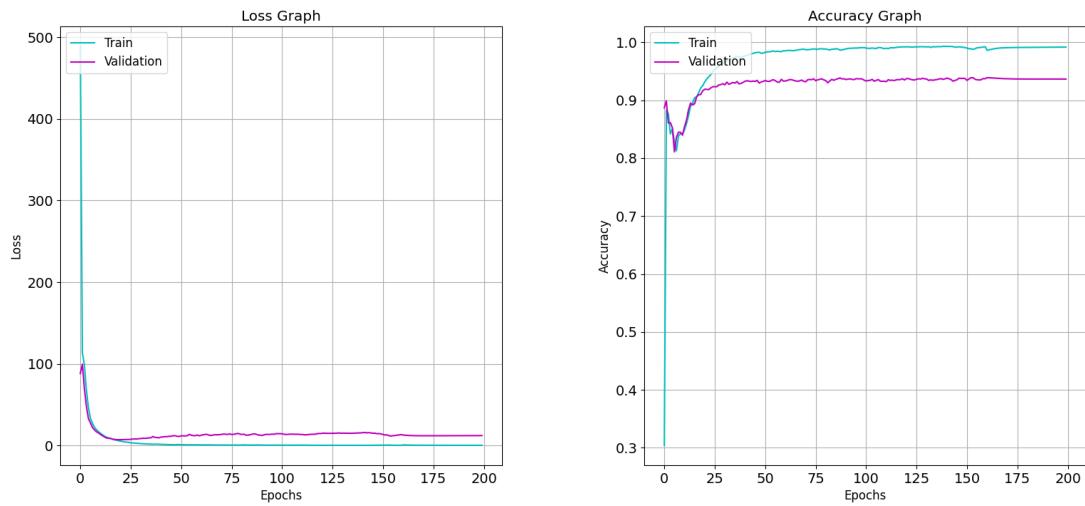


Figura 4.8: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset sinovia-tendine

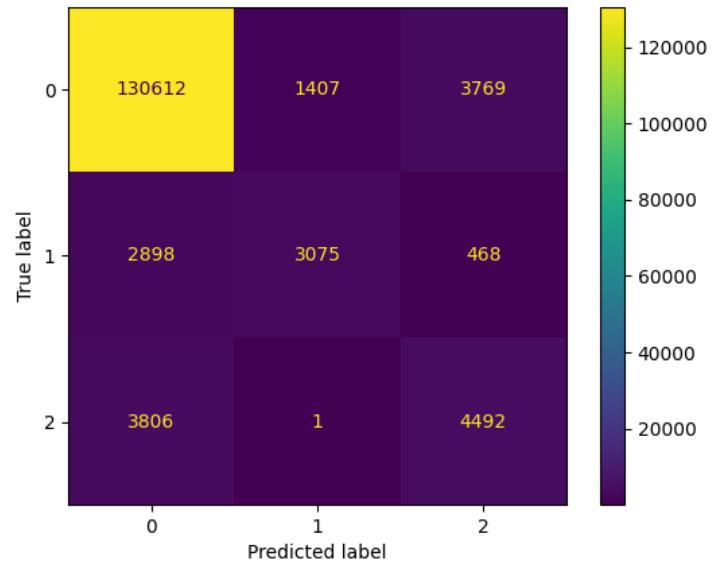


Figura 4.9: Grafico della matrice di confusione multiclasse per le predizioni con la rete VGG16 sul dataset sinovia-tendine

	precision	recall	f1-score
Background	95%	96%	96%
Sinovia	69%	48%	56%
Tendine	51%	54%	53%
accuracy	92%	92%	92%
macro avg	72%	66%	68%
weighted avg	92%	92%	92%

Tabella 4.3: Tabella del report di classificazione per le predizioni con la rete VGG16 sul dataset sinovia-tendine

4.2 Predizioni con rete VGG19 U-Net in Transfer Learning e Fine Tuning

Predizioni del dataset osso-tendine

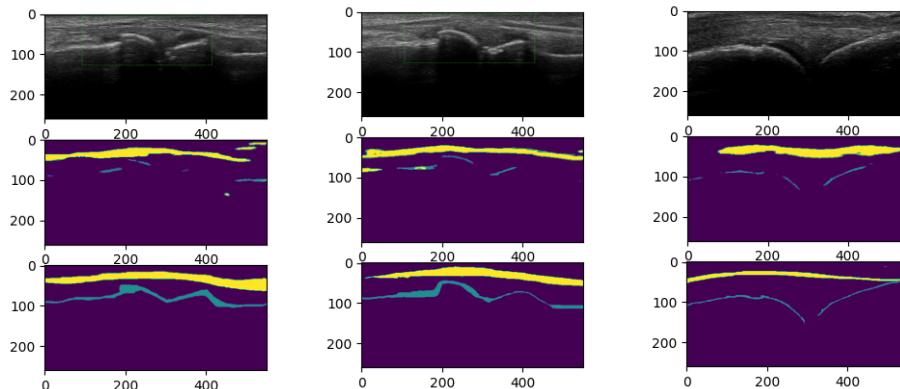


Figura 4.10: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset osso-tendine

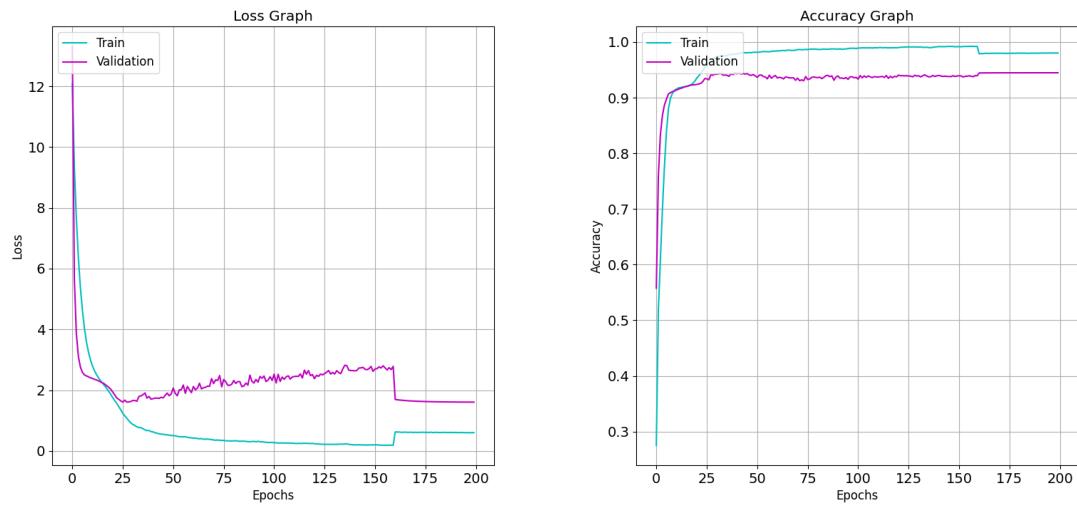


Figura 4.11: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset osso-tendine

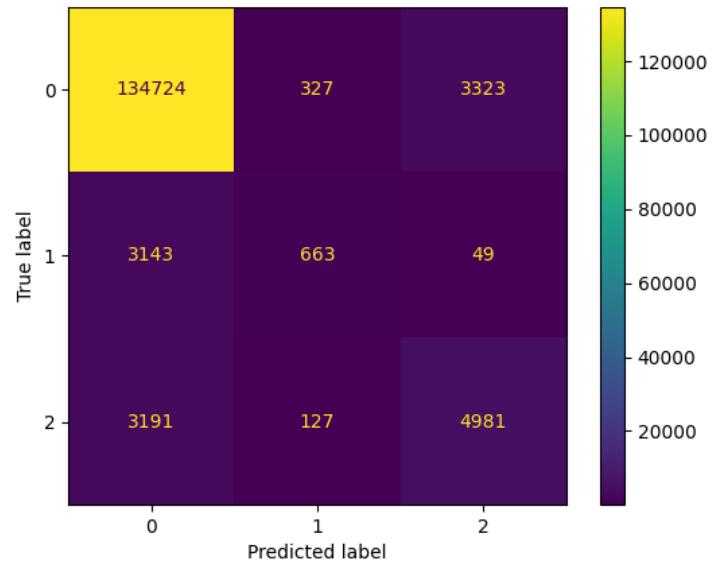


Figura 4.12: Grafico della matrice di confusione multiclasse per le predizioni con la rete VGG19 sul dataset osso-tendine

	precision	recall	f1-score
Background	96%	97%	96%
Osso	59%	17%	27%
Tendine	60%	60%	60%
accuracy	93%	93%	93%
macro avg	71%	58%	61%
weighted avg	93%	93%	93%

Tabella 4.4: Tabella del report di classificazione per le predizioni con la rete VGG19 sul dataset osso-tendine

Predizioni del dataset sinovia-osso

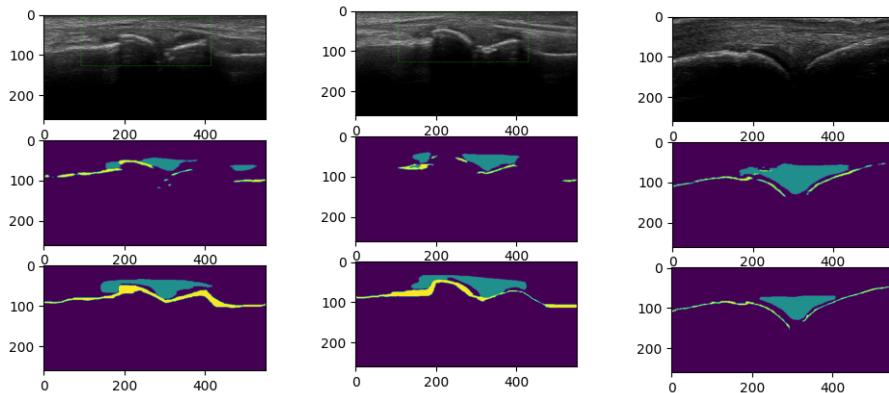


Figura 4.13: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset sinovia-osso

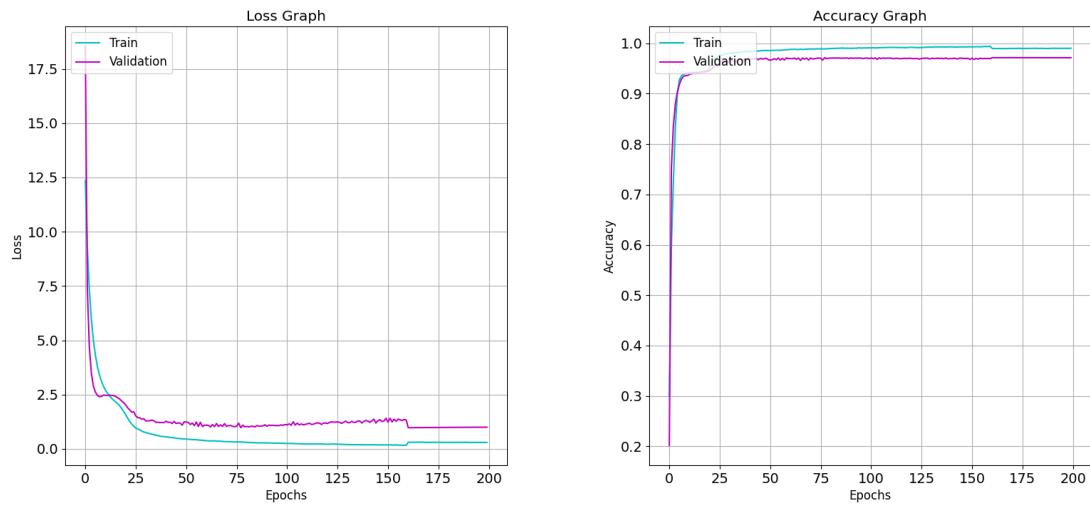


Figura 4.14: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset sinovia-osso

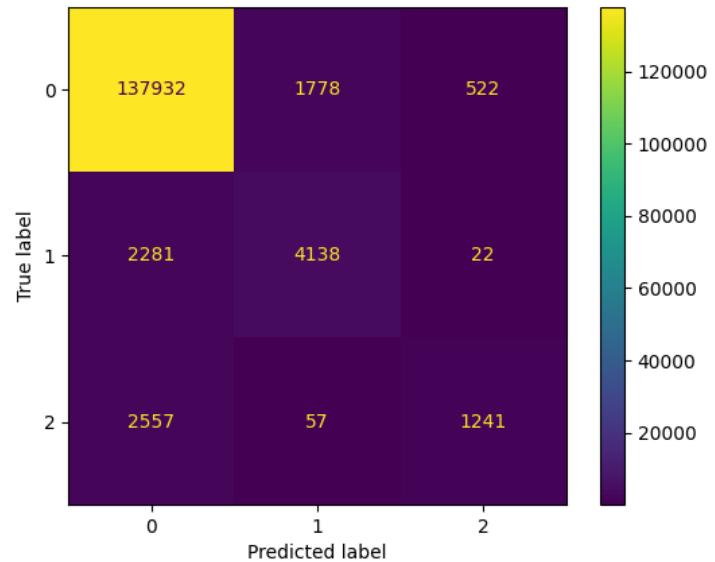


Figura 4.15: Grafico della matrice di confusione multiclasse per le predizioni con la rete VGG19 sul dataset sinovia-osso

	precision	recall	f1-score
Background	97%	98%	97%
Sinovia	69%	64%	67%
Osso	70%	32%	44%
accuracy	95%	95%	95%
macro avg	78%	65%	69%
weighted avg	95%	95%	95%

Tabella 4.5: Tabella del report di classificazione per le predizioni con la rete VGG19 sul dataset sinovia-osso

Predizioni del dataset sinovia-tendine

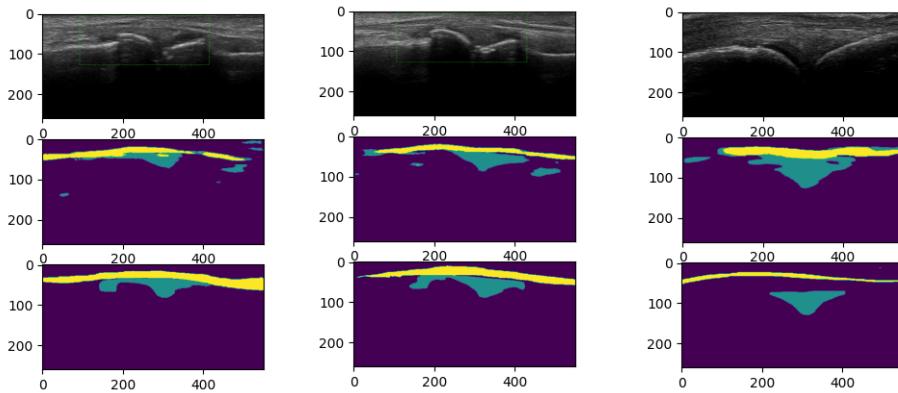


Figura 4.16: Dall'alto, l'immagine di una ecografia in input, la predizione che ha fornito la macchina, la maschera associata all'immagine in input per il dataset sinovia-tendine

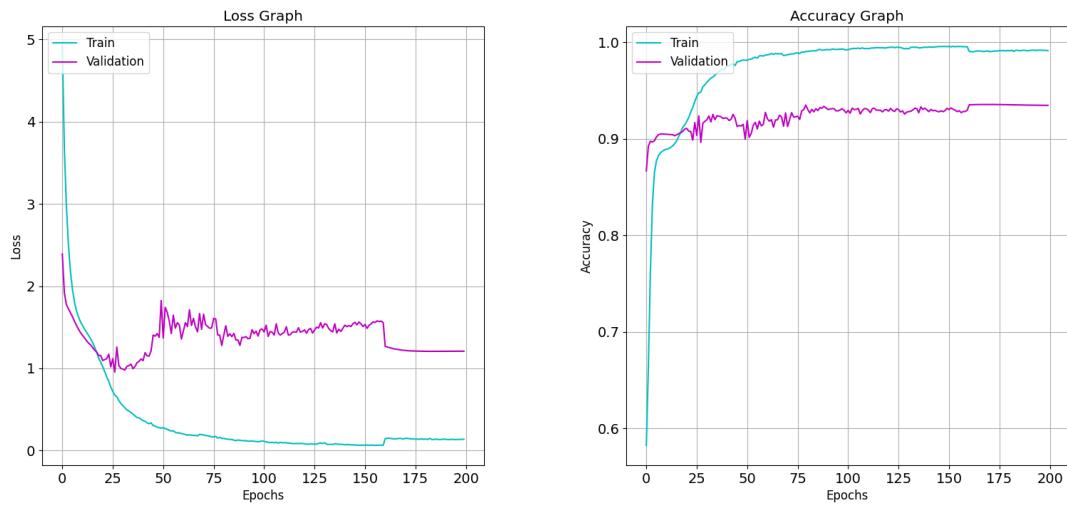


Figura 4.17: Da sinistra, il grafico della funzione di costo e il grafico della funzione di accuratezza per le predizioni con la rete U-Net Base sul dataset sinovia-tendine

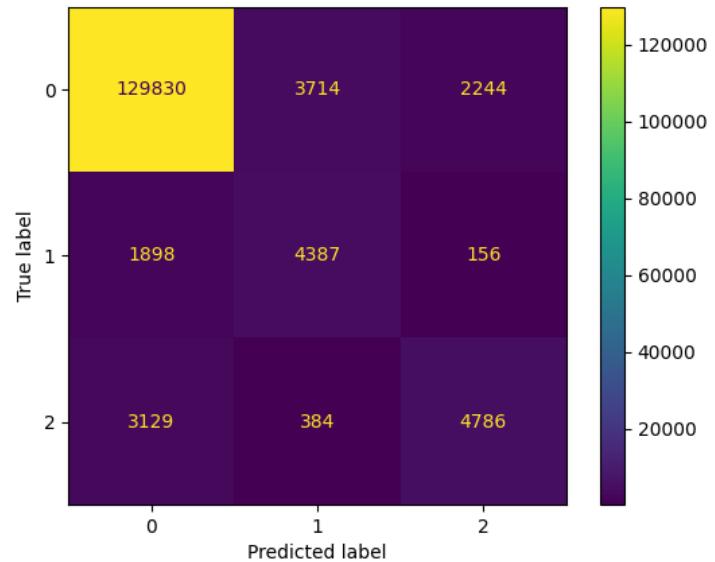


Figura 4.18: Grafico della matrice di confusione multiclasse per le predizioni con la rete VGG19 sul dataset sinovia-tendine

	precision	recall	f1-score
Background	96%	96%	96%
Sinovia	52%	68%	59%
Tendine	67%	58%	62%
accuracy	92%	92%	92%
macro avg	72%	74%	72%
weighted avg	93%	92%	92%

Tabella 4.6: Tabella del report di classificazione per le predizioni con la rete VGG19 sul dataset sinovia-tendine

4.2.1 Commento finale sui test effettuati

Analizzando i test effettuati, si può notare come le reti scelte sono particolarmente adatte a processare immagini mediche. Infatti, nonostante il dataset abbia un numero esiguo di immagini, le metriche prodotte sono sufficienti: in generale, la accuracy risulta essere maggiore del 90% e la F1-score raggiunge livelli maggiori del 70%. Aumentare il dataset con tecniche di *data augmentation* potrebbe migliorare notevolmente questo ultimo dato. Questo permetterebbe anche di effettuare dei training più consistenti e con un numero molto maggiore di epoche.

Nel dettaglio, guardando i grafici della funzione di validazione della accuracy e della loss function si può notare che durante la fase del Fine Tuning queste migliorano, soprattutto si verifica un miglioramento repentino con la rete che usa la VGG19. Riguardo le immagini predette, si può notare che la VGG19 produce in output delle figure con bordi più definiti rispetto a quelle prodotte dalla VGG16. Nonostante ciò, la VGG19 confonde maggiormente i pixel delle classi 1 e 2 rispetto alla VGG16, come si può vedere nella matrice di confusione.

Capitolo 5

Conclusioni

In questa tesi sono state esplorate tecniche di Deep Learning per la segmentazione semantica delle immagini ecografiche. In particolare, è stato analizzato il caso di una segmentazione semantica multiclass, dato che lo scopo assegnato alla macchina è stato quello di individuare l'area rappresentante l'osso, il tendine e la cavità sinoviale delle ecografie in input. I risultati hanno mostrato e confermato che è possibile utilizzare le tecniche di Deep Learning per questo compito, ottenendo delle prestazioni sufficienti anche considerando le dimensioni esigue del dataset in esame. Riguardo questo ultimo aspetto, tecniche varie di espansione del dataset e quindi di data augmentation possono essere sfruttate come sviluppi futuri per rendere più consistenti i training delle reti considerate.

Capitolo 6

Ringraziamenti

Vorrei dedicare questo piccolo spazio alle persone che, con il loro supporto, mi hanno dato da sempre un grande aiuto in questo spettacolare percorso universitario.

Ringrazio il mio relatore e professore Giuseppe Notarstefano che mi ha fornito l'opportunità di effettuare un tirocinio in cui ho imparato argomenti nuovi e che mi ha permesso di stendere questa tesi di laurea lasciandomi approfondire ulteriormente le conoscenze acquisite. Grazie anche al mio correlatore Francesco Ursini per i suoi preziosi consigli e per avermi suggerito puntualmente le giuste modifiche da apportare alla tesi.

Ringrazio i miei correlatori Andrea Camisa e Lorenzo Sforni che sono stati per me due mentori e due guide pronte a fornirmi strumenti, migliorie e suggerimenti utili ai fini della stesura dell'elaborato.

Ringrazio i miei innumerevoli colleghi e prima di tutto amici che hanno condiviso con me gioie e fatiche di questi anni trascorsi insieme. In maniera significativa, ringrazio Andrea, Mirko e Vittorio con i quali ho condiviso grandi momenti di ilarità tra il bar e la gelateria di Borgo Panigale.

Ringrazio Patrick e Lorenzo che con il loro fare scherzoso e paziente sono stati validi insegnanti.

Ringrazio Leonardo che mi ha aiutato ad avere un tetto sopra la testa per il primo periodo a Bologna e che con il suo entusiasmo siamo riusciti ad ampliare questa rete di amicizie.

Un enorme grazie va a Zak e Steve che oltre a sopportarmi hanno anche avuto cura di me e accettato per quella che sono, insegnandomi quanto è importante la gentilezza e l'amore incondizionato verso il prossimo.

Grazie a Emanuele che con il suo spirito di bontà mi ricorda quanto sia bello aiutare le altre persone.

Un ringraziamento speciale e pieno di fierezza va al mio trio di ragazz* e ad oggi anche Ingegner*: a Serena che con il suo esempio di gentilezza e di apertura mentale mi spinge a cercare di fare altrettanto; ad Alice, persona intraprendente e affascinante che si è subito trovata in sintonia con me anche quando abbiamo condiviso i ragionamenti più oscuri di fisica e di analisi 1; infine ad Anna, con la quale dal primo giorno abbiamo coltivato assieme

un comune spirito avventuriero che ci accompagna tutt'ora nelle esperienze più vivaci e piene di emozioni.

Ringrazio soprattutto la mia amica del cuore, che da perfetta compagna di abbuffate di muffin al cioccolato si è rivelata una persona forte d'animo e roccia su cui appoggiarmi nei miei momenti più bui.

Senza di voi oggi avrei meno storie belle da raccontare e da ricordare.

Ringrazio tutta la mia famiglia, che continua ad essere molto unita e che per questo rimarrà un forte punto di riferimento.

Ringrazio Paolo e Lisa come esempi di professionisti Ingegneri, i quali mi hanno ispirato a intraprendere questa carriera. In particolar modo, ringrazio Lisa per l'ispirazione di Donna e Ingegnera.

Ringrazio le mie nonne come esempi di figure femminili forti ognuna a proprio modo, che mi hanno insegnato la costanza, la perseveranza e a non mollare di fronte alle difficoltà.

Ringrazio mia cugina Virginia per aver colorato i mesi durante la pandemia passata e che mi ha dato ogni giorno un motivo per sorridere e tenermi attiva.

Grazie ai miei genitori che mi hanno permesso di studiare credendo in me e che mi continueranno a supportare nelle mie future esperienze.

Vorrei ringrazie di nuovo mamma e papà per il loro impegno nell'aiutare me e mia sorella a creare il rapporto unico che ci lega. Quindi ringrazio anche Ada, che rimane un punto fondamentale della mia vita e che senza la quale non sarei riuscita a definire la mia persona.

Concludo dicendo che sono sinceramente orgogliosa di avervi conosciuto e di potervi ringraziare tutti quanti. Grazie.

Bibliografia

- [1] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [2] T. M. Mitchell e T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997, vol. 1.
- [3] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, n. 6, pp. 141–142, 2012.
- [4] L. Deng e D. Yu, “Deep Learning: Methods and Applications.,” *Found. Trends Signal Process.*, vol. 7, n. 3-4, pp. 197–387, 2014. indirizzo: <http://dblp.uni-trier.de/db/journals/ftsig/ftsig7.html#DengY14>.
- [5] W. McCulloch e W. Pitts, “A Logical Calculus of Ideas Immanent in Nervous Activity,” *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943.
- [6] L. Hardesty, “Explained: Neural networks - Ballyhooed artificial-intelligence technique known as âdeep learningâ revives 70-year-old idea.,” *MIT News*, 2017.
- [7] M. A. Nielsen, *Neural Networks and Deep Learning*, misc, 2018. indirizzo: <http://neuralnetworksanddeeplearning.com/>.
- [8] D. Ng e M. Feng, “Medical Image Recognition: An Explanation and Hands-On Example of Convolutional Networks,” in ago. 2020, pp. 263–284, ISBN: 978-3-030-47993-0. DOI: [10.1007/978-3-030-47994-7_16](https://doi.org/10.1007/978-3-030-47994-7_16).
- [9] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever e R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.,” *Journal of Machine Learning Research*, vol. 15, n. 1, pp. 1929–1958, 2014. indirizzo: <http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>.
- [10] H. He e E. Garcia, “Learning from Imbalanced Data,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, n. 9, pp. 1263–1284, 2009, ISSN: 1041-4347. DOI: [10.1109/TKDE.2008.239](https://doi.org/10.1109/TKDE.2008.239). indirizzo: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5128907&tag=1.
- [11] T. S. Huang, “Computer Vision: Evolution And Promise,” 1996.
- [12] R. Szeliski. “Computer vision algorithms and applications.” (2011), indirizzo: <http://dx.doi.org/10.1007/978-1-84882-935-0>.

- [13] S. Wang e Z. Su, *Metamorphic Testing for Object Detection Systems*, 2019. DOI: 10.48550/ARXIV.1912.12162. indirizzo: <https://arxiv.org/abs/1912.12162>.
- [14] A. Zhang, Z. C. Lipton, M. Li e A. J. Smola, “Dive into Deep Learning,” *arXiv preprint arXiv:2106.11342*, 2021.
- [15] D. P. P. Borelli, “L’artrosi delle dita,” indirizzo: chirurgiadellamanobrescia.it/patologie-mano/curare-artrosi-delle-dita/.
- [16] F. Chollet et al. “Keras.” (2015), indirizzo: <https://github.com/fchollet/keras>.
- [17] M. Abadi, A. Agarwal, P. Barham et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015. indirizzo: <https://www.tensorflow.org/>.
- [18] Itseez, *Open Source Computer Vision Library*, <https://github.com/itseez/opencv>, 2015.
- [19] O. Ronneberger, P. Fischer e T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. DOI: 10.48550/ARXIV.1505.04597. indirizzo: <https://arxiv.org/abs/1505.04597>.
- [20] K. Simonyan e A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014. DOI: 10.48550/ARXIV.1409.1556. indirizzo: <https://arxiv.org/abs/1409.1556>.
- [21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li e L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [22] D. P. Kingma e J. Ba, *Adam: A Method for Stochastic Optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. indirizzo: <https://arxiv.org/abs/1412.6980>.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [24] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, n. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.