Università degli studi di Milano-Bicocca

Advanced Machine Learning
Progetto Finale

Histopathologic cancer detection

Autori:
Confalonieri Riccardo, 830404, r.confalonieri5@campus.unimib.it
Cogo Luca, 830045, l.cogo@campus.unimib.it

A.A.: 2021/2022

Indice

1 Introduzione **1**

2 Dataset **2**

3 Approccio metodologico **2**
3.1 Definizione delle architetture 3
3.2 Configurazione del training 3
3.3 Data augmentation 3
3.4 Ottimizzazione degli iperparametri 4
3.5 Fine tuning 4
3.5.1 VGG16 5
3.5.2 DenseNet-121 6
3.5.3 MobileNet-V2 7
3.5.4 Scelta degli iperparametri 8
3.6 Rete from scratch 9

4 Risultati e valutazione **11**

5 Conclusioni **13**

Abstract

Il crescente utilizzo negli ultimi anni di immagini digitali in campo medico ha reso possibile l'utilizzo di tecniche di apprendimento automatico per la diagnosi di varie patologie. In questo report viene valutata l'efficacia delle più recenti tecniche di deep learning nel diagnosticare tumori al sistema linfatico a partire da immagini realizzate tramite appositi scanner WSI (Whole Slide Image), rese disponibili dal dataset benchmark PatchCamelyon (PCam). A tale scopo verranno descritti diversi approcci: una Convolutional Neural Network addestrata da zero e alcune architetture (VGG16, DenseNet-121, MobileNet-V2) basate su modelli pre-trainati per il riconoscimento di immagini (Imagenet). Verrà inoltre descritto l'approccio utilizzato per ottimizzare gli iperparametri dei diversi modelli e le tecniche di data augmentation sperimentate. Verranno infine confrontati i modelli considerati, i quali hanno mostrato delle performance soddisfacenti, specialmente nel caso dell'architettura realizzata from scratch, la quale ha ottenuto una accuracy del 93%.

1 Introduzione

La patologia digitale è una branca della patologia che si concentra sulla diagnosi di malattie tramite l'analisi di immagini digitali. Tale pratica trae le sue radici negli anni '60, ma è negli ultimi anni che, grazie all'introduzione del Whole Slide Imaging (WSI), sta ricevendo sempre più attenzione nel campo della medicina diagnostica, con l'obiettivo di ottenere analisi efficienti ed economiche. Nel 2017, per la prima volta, la FDA (U.S. Food and Drug Administration) ha consentito la commercializzazione di scanner WSI a scopo diagnostico [1]. Parallelamente a ciò, l'utilizzo di tecniche di apprendimento automatico in campo medico è in costante crescita, come dimostrato dal rapido aumento di strumenti di analisi basati su tecniche di machine learning [2][3]. In un contesto come quello descritto, diventa evidente la necessità di sperimentare in ambito medico le più recenti tecniche di deep learning, in modo da ottenere strumenti diagnostici sempre più affidabili.
Questo lavoro si pone l'obiettivo di realizzare un modello in grado di individuare tumori al sistema linfatico. Solitamente la diagnosi di tale patologia viene effettuata da un esperto che, tramite l'analisi al microscopio di campioni di tessuto colorato con ematossilina eosina, individua le tipiche caratteristiche delle masse tumorali. Effettuare manualmente questo controllo rappresenta un compito noioso e ridondante, pertanto è da considerarsi facilmente soggetto ad errori umani per via della ripetitività del processo. Uno sbaglio, in questo contesto, può avere ripercussioni gravi in quanto l'individuazione delle cellule cancerogene fin dalle prime fasi dello sviluppo permette cure più appropriate ed efficaci. Mettere a disposizione dei patologi strumenti automatici che li aiutino nello svolgere questo compito sarebbe estremamente utile. Sostanzialmente questo problema può essere risolto con modelli predittivi attraverso un task di classificazione binaria sulle immagini ottenute tramite scansioni WSI.
Il crescente studio degli ultimi anni sulle reti neurali ha fatto si che problemi del genere, prima considerati di difficile risoluzione, diventassero approcciabili con risultati soddisfacenti. In particolare le reti che più si addicono a questo tipo di compito, che dunque verranno utilizzate, sono le Convolutional Neural Networks (CNN), la cui architettura si ispira alla corteccia visiva animale ed è dunque pensata appositamente per task di computer vision.

2 Dataset

Il dataset utilizzato per lo sviluppo del progetto è stato pubblicato nel 2018 e reso disponibile come progetto su GitHub [4]. Nella sua versione originale il si compone di 327.680 immagini a colori di dimensione 96x96px estratte da scansioni istopatologiche di sezioni linfonodali. Il dataset risulta essere già suddiviso in un training set di 262.144 immagini e un validation e test set di 32.768 immagini ciascuno. Inoltre non vi è alcuna sovrapposizione nei WSI tra i diversi set e tutti i set mantengono preventivamente un bilanciamento 50/50 tra gli esempi positivi e negativi. Ogni immagine è annotata con un'etichetta binaria che indica la presenza di tessuto metastatico: un'etichetta positiva indica che nella regione centrale di 32x32px è contenuto almeno un pixel di tessuto tumorale mentre, allo stesso modo, un'etichetta negativa indica l'assenza di cellule tumorali nella regione in analisi. Il tessuto tumorale eventualmente contenuto nella regione esterna non influenza in alcun modo l'etichetta associata all'immagine. Tale regione, come riportato nei riferimenti del dataset, è fornita solamente per consentire la progettazione di modelli che non utilizzino il padding nei diversi layer convoluzionali, questo garantisce un comportamento coerente quando il modello è applicato a un'immagine dell'intera diapositiva. In figura 1 sono riportati alcuni esempi in cui si evidenzia la regione d'interesse.
Si è però notato che il dataset così fornito contiene diverse immagini duplicate in quanto realizzato attraverso un campionamento probabilistico. Per questa ragione si è scelto di utilizzare come dataset di riferimento quello reso disponibile tramite la piattaforma Kaggle [5], il quale non contiene duplicati. La dimensione totale del dataset quindi disponibile risulta pari a 220.025 immagini, senza alcuna divisione prioritaria tra training, validation e test. Il dataset Kaggle, avendo eliminato i duplicati, risulta essere sbilanciato a favore della classe negativa, ovvero quella senza cellule tumorali. Questo problema verrà risolto tramite un ribilanciamento delle classi, come verrà precisato nella sezione 3.2. Inoltre una prima analisi ha evidenziato la presenza di immagini mal digitalizzate che non riportano alcuna informazione e sono completamente bianche o nere. Tuttavia non è stata condotto alcun preprocessing mirato a rimuove queste immagini per via della complessità del task.
L'unico preprocessing, apportato a tutte le immagini è la normalizzazione del valore dei pixel nel range $[0, 1]$.

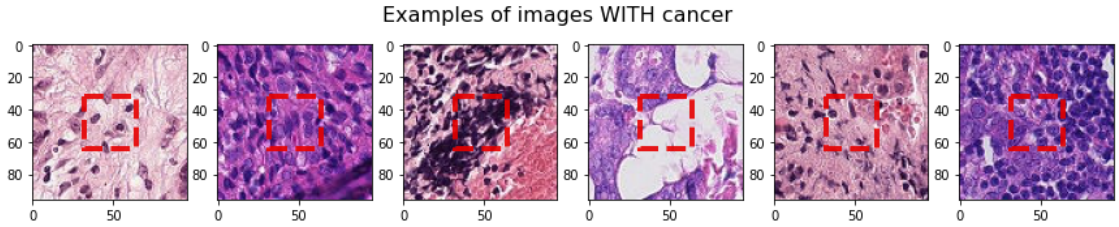

Figura 1: Esempio di immagini con cellule tumorali nella regione d'interesse

3 Approccio metodologico

Questa sezione è dedicata a descrivere l'approccio metodologico seguito per lo sviluppo del progetto. Verranno anzitutto indicate le scelte comuni a tutti i modelli sperimentati, passando poi alla descrizione delle tecniche di data augmentation provate. Tutti gli esperimenti sono stati condotti utilizzando il servizio cloud Google Colab, che mette a disposizione un ambiente virtuale su cui eseguire codice Python [6] (in versione 3.7). Per la definizione e l'addestramento dei modelli si sono utilizzate la libreria TensorFlow [7] e in, particolar modo, l'interfaccia ad alto livello Keras [8].Per gli approcci di ottimizzazione degli iperparametri è stata invece utilizzata la libreria Keras-Tuner [9].

3.1 Definizione delle architetture

In fase di sviluppo del progetto si è deciso di sperimentare due distinte tipologie di approcci, così da poter poi valutare e comparare i risultati ottenuti e selezionare il migliore. Il primo approccio prende il nome di fine tuning e prevede l'utilizzo di reti precedentemente addestrate per un altro task di riconoscimento di immagini (*i.e.* Imagenet), opportunamente modificate per adattarsi al nuovo task di classificazione binaria. Questa tecnica è stata applicata su tre diverse architetture note in letteratura: *VGG16*, *DenseNet-121* e *MobileNet-V2*. Il secondo approccio consiste invece nella realizzazione di una Convolutional Neural Network progettata ad hoc per la risoluzione del task affrontato.

3.2 Configurazione del training

Per poter utilizzare correttamente il dataset è stato necessario applicare alcuni step di preprocessing che hanno permesso una gestione più efficace del caricamento e dei training dei modelli. Si è deciso di utilizzare, per una prima fase, soltanto una piccola porzione dell'intero dataset. Questo ha permesso di effettuare un primo training dei modelli valutando diverse architetture e anche di analizzare le tematiche relative agli approcci di ottimizzazione degli iperparametri. Nello specifico il primo subset creato è caratterizzato da 21.326 di immagini di training, 3762 di validation e 5000 immagini di test. Ciascuno di questi set è stato opportunamente ri-bilanciato mantenendo lo stesso numero di esempi positivi e negativi. Si è scelto di mantenere il bilanciamento anche per il test set in quanto non si è a conoscenza della distribuzione reale delle casistiche, sebbene si può ipotizzare che i casi positivi siano meno rispetto ai casi negativi. Questo garantisce risultati il più possibile neutri e comparabili anche se non riporterà una casistica reale al 100%. Inoltre il bilanciamento è lo stesso proposto dai realizzatori del dataset e quindi permette di confrontare i risultati ottenuti con altre architetture proposte negli anni. Infine, durante questa fase di creazione del subset, le immagini di training e validation sono state opportunamente divise in due sottocartelle così da dividere le immagini con cellule tumorali (positive) dalle immagini sane (negative). Questo, attraverso l'utilizzo di metodi ad hoc forniti dalla libreria Keras [8] per il caricamento delle immagini, ha permesso di elaborare le immagini in piccoli batch evitando così di caricare tutto il set in memoria garantendo quindi una facile scalabilità a dataset più grandi. Seguendo lo stesso ragionamento si è preparato un secondo dataset di dimensione maggiore da utilizzarsi in una seconda fase del progetto per addestrare nuovamente i modelli migliori. Lo scopo di questo secondo dataset è quello di migliorare le performance sfruttando maggiormente la grande quantità di immagini a disposizione. Il secondo subset è caratterizzato da 100.352 immagini tra train e validation, e 18.000 di test. Il campionamento delle immagini, per entrambi i subset, è definito da un random_state garantendo così la replicabilità.
Per quanto concerne l'addestramento dei modelli si è deciso di fissare un massimo di 50 epoche, utilizzando l'approccio dell'EarlyStopping per interrompere il processo dopo tre epoche senza alcun miglioramento della validation loss. Tale approccio, oltre a fermare l'apprendimento, ripristina anche i pesi corrispondenti all'epoca migliore individuata, di fatto eliminando il training delle ultime tre epoche. Tutte le reti inoltre condividono la stessa loss function, ovvero *binary crossentropy*, che rappresenta la scelta più comune quando si affrontano problemi di classificazione binaria. Prima di effettuare tutti gli esperimenti si è provveduto a fissare un seed per la generazione randomica dei pesi, in modo da garantire la ripetibilità di ogni operazione.

3.3 Data augmentation

Le tecniche di data augmentation permettono di ampliare il dataset a disposizione senza effettivamente raccogliere nuovi elementi. Queste tecniche applicano ai dati già esistenti

dei cambiamenti casuali controllati, introducendo dunque rumore nelle immagini e creando così delle copie modificate. Questo permette di ottenere dei modelli più robusti e flessibili, permettendo una migliore generalizzazione su nuove immagini. Si è scelto di provare ad introdurre queste tecniche per gestire eventuali rumori presenti nelle immagini, come emerso nell'esplorazione iniziale del dataset sono infatti presenti alcune immagini completamente bianche o nere. Inoltre, pur non avendo una conoscenza approfondita dello specifico dominio, si è immaginato che macchinari differenti potessero restituire immagini lievemente diverse. Si è dunque provato ad utilizzare dei layer di data augmentation per introdurre dei cambiamenti specifici e safe nelle immagini. Nello specifico si è scelto di introdurre le seguenti distorsioni alle immagini:

- *Random flip*: che capovolge casualmente l'immagine in orizzontale e/o verticale.
- *Random rotation*: che ruota l'immagine in un range di gradi predefinito.
- *Random contrast*: regola in modo il contrasto di un'immagine in base a un fattore casuale. Il contrasto viene regolato indipendentemente per ciascun canale di ciascuna immagine durante il training.

Queste tecniche sono state applicate alle sole immagini di training, e soltanto in una seconda fase. Inizialmente infatti si sono creati modelli senza i layer di data augmentation, i quali sono stati introdotti successivamente durante il processo di ottimizzazione degli iperparametri per verificare se potessero introdurre dei miglioramenti significativi.

3.4 Ottimizzazione degli iperparametri

In tutti gli approcci provati, si è scelto di utilizzare tecniche di hyperparameter optimization per individuare i parametri ottimali. Questo ha anche consentito di stabilire le migliori scelte architetturali sia nel caso delle reti preaddestrate, per decidere quanti layer fully connected aggiungere e quanti neuroni in ciascuno di essi, sia per la rete from scratch per definirne l'intera struttura. Per tutti gli approcci si è fatto ricorso all'algoritmo di ottimizzazione bayesiana, che nel caso specifico è stata definita per trovare gli iperparametri che minimizzassero la validation loss. L'algoritmo utilizza ad ogni iterazione un nuovo set di iperparametri, selezionato tenendo conto dei risultati ottenuti agli step precedenti tramite inferenza bayesiana evitando dunque di provare tutte le possibili combinazioni.

3.5 Fine tuning

Per l'approccio di fine tuning sono state selezionate tre diverse architetture caratterizzate da un diverso numero di parametri, di operazioni richieste e top-1 accuracy raggiunta sul task originale. Ciò ha permesso di studiarne le differenze e selezionare la rete migliore per il nuovo task. Ognuna di queste architetture è stata scaricata da Keras utilizzando i pesi definiti per il task di Imagenet. Inoltre tutti i modelli scaricati sono stati tagliati alla fine dei layer convoluzionali, i quali non verranno addestrati nuovamente per il nuovo task. Per ognuno dei modelli creati è stato inoltre applicato il preprocessing specifico, in modo che le immagini potessero essere correttamente trattate dai layer predefiniti. Per poter utilizzare questi modelli è stato necessario effettuare un reshaping dell'immagine ad una dimensione pari a 224x224px, mantenendo comunque i tre canali colore. Questa modifica è dovuta al fatto che le reti sono state preaddestrate su immagini di queste dimensioni, si potrebbero quindi generare errori con input diversi soprattutto nei layer convoluzionali.
Per quanto concerne il training di questi modelli si segnala che si è utilizzata come funzione di ottimizzazione Adam in quanto risultata la più performante rispetto alle altre possibilità.

3.5.1 VGG16

VGG16 è stata sviluppata e introdotta nel 2014 da Karen Simonyan et al. [10]. Il termine 'VGG' è un abbreviazione per 'Visual Geometry Group', ovvero il gruppo di ricercatori dell'Università di Oxford che ha sviluppato questa architettura, mentre il numero finale 16 riporta il numero di layers presenti nella rete. Questa è una delle prime architetture CNN proposte, nonché una delle più famose ed utilizzate. L'architettura della rete, riportata in Figura 2, è caratterizzata da stack di layer convoluzionali, con dimensione del kernel 3x3, applicati sequenzialmente. Questo modello è caratterizzato da un numero molto elevato di parametri e di operazioni, e permette di raggiungere una top-1 accuracy pari circa allo 0.73 sul task originale imagenet.
Come menzionato precedentemente, questa rete è stata tagliata alla fine dei blocchi convoluzionali, rimuovendo dunque i tre layer fully connected finali i quali sono stati sostituiti dai seguenti layer aggiuntivi:

- Fully connected con 64 neuroni.
- Fully connected con 32 neuroni e dropout rate 0.4.
- Fully connected con 8 neuroni.
- Fully connected finale con 2 neuroni che mappa alla dimensione dell'output.

Per tutti i layer aggiuntivi è stata usata come funzione di attivazione Relu. Durante la fase di addestramento soltanto i layer aggiuntivi sono stati settati come trainabili mentre i pesi dei layer relativi alla struttura base di VGG16 sono stati bloccati. Complessivamente si è dunque arrivati a realizzare una struttura con: 16.323.026 parametri totali di cui trainabili 1.608.338.

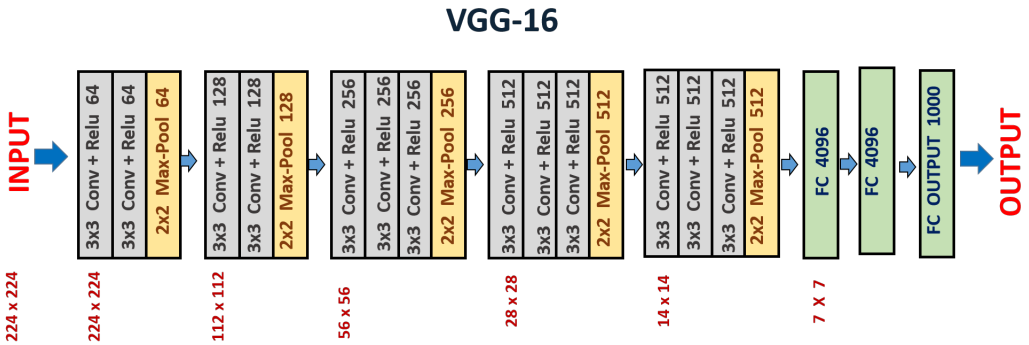

Figura 2: Architettura standard di VGG16

Fonte: `https://www.kaggle.com/thepinokyo/fruit-detection-with-vgg-16`

Questa rete ha ottenuto dei risultati, sul validation set, comparabili con quanto ci si aspettava date le caratteristiche dell'architettura. Nello specifico si è raggiunta una loss di 0.529 e un'accuracy di 0.42. Inoltre il modello ha richiesto solo 10 epoche di training, di cui le ultime tre eliminate per via dell'EarlyStopping che ripristina i pesi dell'epoca migliore. L'andamento del training è riportato in Figura 3, dalla quale si nota la presenza di un underfitting abbastanza accentuato che non si è riusciti a risolvere.

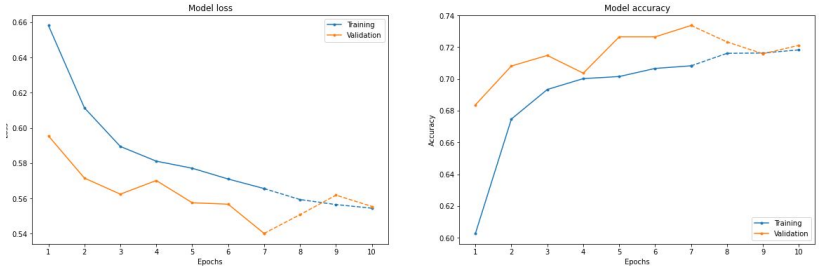

Figura 3: Andamento di loss e accuracy durante l'addestramento della rete VGG-16

3.5.2 DenseNet-121

DenseNet-121 è un'architettura più recente, introdotta nel 2018 da Gao Huang et al. [11]. In una CNN tradizionale ogni strato convoluzionale, tranne il primo che riceve l'input, riceve l'output dello strato convoluzionale precedente. Pertanto una rete con L layer avrà esattamente L connessioni dirette e sequenziali tra essi. Tuttavia con l'aumentare della complessità delle reti, e di conseguenza delle profondità, questa struttura ha evidenziato alcuni limiti tra cui il vanishing gradient. Il gradiente non riusciva infatti a raggiungere i primi layer dell'architettura riducendo quindi l'abilità della rete di apprendere i pesi correttamente. DenseNet risolve questo problema semplificando le connettività tra i diversi layer, ogni livello è infatti connesso con ogni altro livello, quindi avendo L layer si avranno $\frac{L(L+1)}{2}$ connessioni. Nello specifico l'input di un layer è dato dalla concatenazione delle feature di output degli altri layer. Per ottenere questo risultato sono stati introdotti dei layer specifici:

- *DenseBlock*. La concatenazione è possibile soltanto se il numero di features è il medesimo. Per questo motivo DenseNet utilizza dei DenseBlock, all'interno di ogni blocco il numero di feature rimane costante quello che cambia sono il numero di filtri convoluzionali applicati.

- *Transition*. Sono i layer che permettono di collegare i diversi DenseBlock, riducono il numero di canali alla metà rispetto al numero di canali esistenti. Per farlo applicano una batch normalization, una convoluzione 1x1 e un average pooling 2x2.

- *Bottleneck layer*. Si introduce una convoluzione con dimensione 1x1 prima delle convoluzioni 3x3 questo riduce il numero di feature e migliora l'efficienza della rete.

La figura 4 riporta, in versione semplificata, la struttura della rete focalizzandosi sui nuovi blocchi introdotti. Questa struttura porta diversi vantaggi tra cui la risoluzione del vanishing gradient, rafforza la propagazione e il riutilizzo delle feature. Un'altra sostanziale differenza, rispetto alla rete VGG16, è la presenza di un solo layer fully connected finale che permette di ridurre sensibilmente il numero di parametri. Questa architettura, oltre ad utilizzare pochi parametri, richiede poche operazioni. Tutto ciò senza inficiare le previsioni, sul task originale infatti la top-1 risulta essere pari a circa 0.75 migliorando quindi quanto fatto da VGG16.
Come per la rete precedente tale architettura è stata tagliata dopo il layer di flatten, e i layer finali aggiunti rispecchiano la configurazione proposta per VGG16. In totale la struttura è risultata composta da: 7.105.466 parametri totali di cui solo 67.962 trainabili.

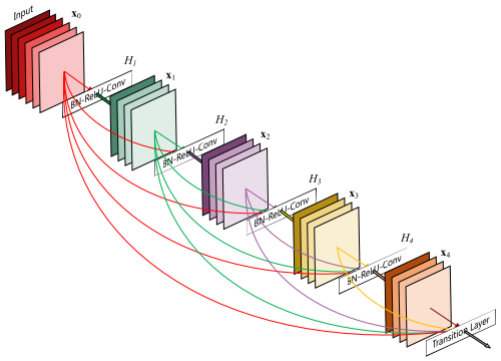

Figura 4: Architettura standard di DenseNet-121. Focus sui nuovi layer introdotti
Fonte: `https://arxiv.org/pdf/1608.06993.pdf`

Questa rete è risultata essere la migliore, rispetto alle altre architetture di fine-tuning proposte, ed è stata dunque addestrata sia sul dataset piccolo che sul dataset medio. Su quest'ultimo si è raggiunta una validation loss pari a 0.432 e una validation accuracy di 0.80, terminando dopo 16 epoche di cui le ultime tre scartate per via dell'EarlyStopping. Di seguito, in figura 5, si riporta l'andamento di loss e accuracy durante l'addestramento della rete. Anche per questa rete è possibile verificare la presenza di underfitting, anche se leggermente meno accentuato rispetto al precedente.

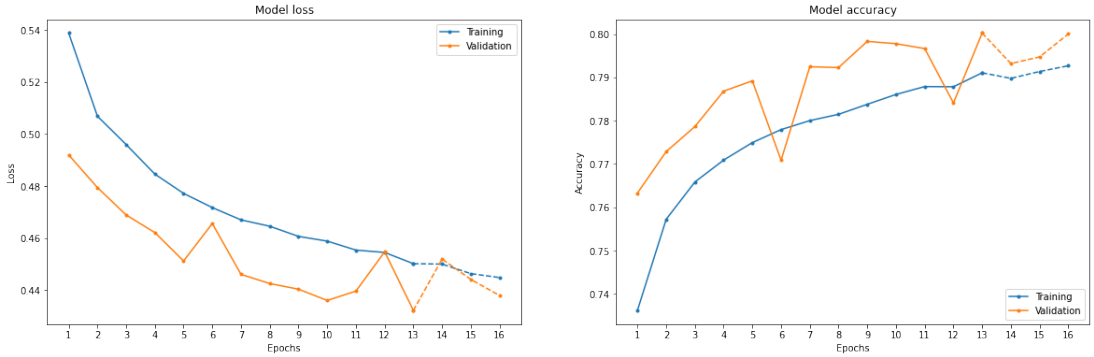

Figura 5: Andamento di loss e accuracy durante l'addestramento della rete DenseNet-121

3.5.3 MobileNet-V2

MobileNet-V2 è l'ultima architettura considerata per il fine tuning. Questa architettura è stata sviluppata nel 2017 da un gruppo di ricercatori di Google [12]. Questa architettura è fortemente ottimizzata per essere utilizzata su dispositivi mobili, il numero di parametri e operazioni richieste è infatti minimo e, inoltre, anche il peso del modello, 14MB, è ideale per essere utilizzato su dispositivi con poca memoria. Nonostante questo, sul task originale Imagenet, il modello riesce a raggiungere quasi la stessa accuracy di altri modelli più complessi come VGG. L'idea alla base di questa architettura, riportata in Figura 6, è utilizzare dei *BottleNeck residual block* che sono costituiti da tre layer convoluzionali. Il primo layer è una convoluzione 1x1 che ha lo scopo di espandere il numero di canali, successivamente viene applicata una convoluzione in profondità che applica la convoluzione sui singoli canali separatamente e infine viene applicata una convoluzione pointwise che utilizza dei kernel 1x1 con lo scopo di ridurre il numero di canali. Vi sono altri due fattori molto importanti in questa architettura: vi è una connessione residuale tra l'input e l'output dopo la convoluzione pointwise e, inoltre, dato che i nuovi layer producono dati in una ridotta dimensione, vengono

utilizzate funzioni di attivazione lineari per evitare di distruggere troppe informazioni.
Come nei casi precedenti la rete preaddestrata è stata tagliata prima dei blocchi fully connected, e i layer rimasti sono stati bloccati per evitare il loro addestramento. I layer finali sono stati sostituiti dagli stessi tre blocchi fully connected descritti per la rete VGG, l'unica differenza è che in questo caso il terzo fully connected layer utilizza 16 neuroni invece di 8. Il modello finale è quindi composto da: 2.342.610 parametri totali e 84.626 parametri trainabili.

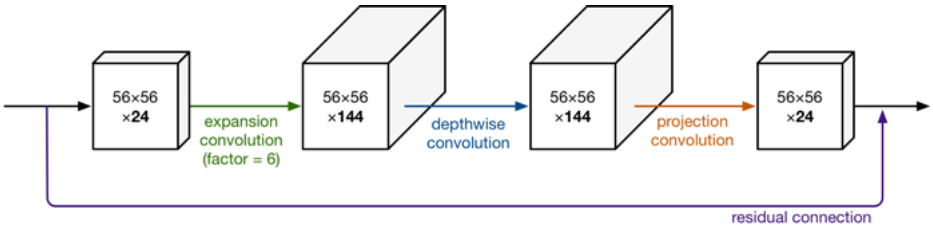

Figura 6: Architettura standard di MobileNet-V2. Focus sui bottleneck residual block
Fonte: `https://machinethink.net/blog/mobilenet-v2/`

La rete ottiene risultati molto simili alla rete VGG, con una validation loss pari a 0.51 e una validation accuracy pari a 0.75. Anche in questo caso il numero di epoche complessive richieste, considerando le tre eliminate successivamente dall'EarlyStopping, sono 10. L'andamento globale del training è riportato in figura 7, si può notare la presenza di un underfitting abbastanza accentuato come in precedenza.

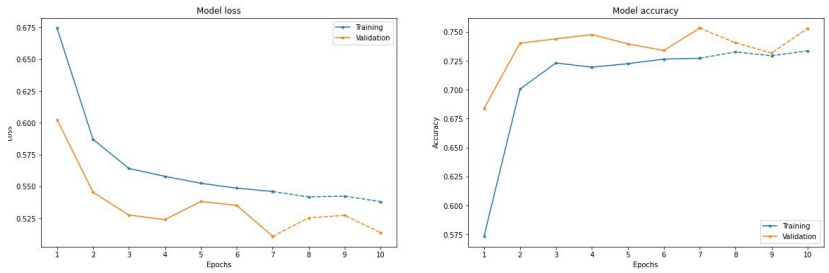

Figura 7: Andamento di loss e accuracy durante l'addestramento della rete MobileNet-V2

3.5.4 Scelta degli iperparametri

Con lo scopo di migliorare i risultati ottenuti dal fine tuning, è stato applicato l'approccio di ottimizzazione degli iperparametri. Durante questa fase sono stati introdotti anche i layer di data augmentation, sottoposti anch'essi all'ottimizzazione. In particolare i parametri sottoposti all'ottimizzazione, con le rispettive possibilità, sono i seguenti:

- Layer di random flip: [True; False]
- Layer di random rotation con range di rotazione compreso tra $[-0.5, 0.5]$ e step pari a 0.05.
- Layer di random contrast con range di rotazione compreso tra $[-0.7, 0.7]$ e step pari a 0.1.
- Learning Rate: $[1e-2;\ 1e-3;\ 1e-4]$
- Layer Fully Connected (FC):
  - Dimensione del primo layer: $[32, 64, 96, 128]$

- Dropout per il primo layer: $[0.0, 0.2, 0.4, 0.5]$
- Dimensione del secondo layer: $[16, 32]$
- Dropout per il secondo layer: $[0.0, 0.2, 0.4, 0.5]$
- Dimensione del terzo layer: $[8, 16]$

In totale si è dunque ottenuto uno spazio di ricerca molto ampio, dunque è stato utilizzato l'algoritmo di ottimizzazione bayesiana per diminuire il numero di combinazioni trovate secondo regole probabilistiche. Nel caso della rete DenseNet-121 si è provato anche ad aggiungere un quarto layer finale fully connected il quale però non ha portato benefici ed è stato scartato dall'ottimizzatore. La tabella 1 riporta i migliori parametri trovati per le diverse architetture:

Tabella 1: Migliori iperparametri per le reti di fine tuning

Parametro	**VGG16**	**DenseNet-121**	**MobileNet-V2**
Num. neuroni primo layer FC	32	128	128
Num. neuroni secondo layer FC	32	16	32
Num. neuroni terzo layer FC	8	16	8
Dropout dopo il primo layer FC	0.0	0.0	0.2
Dropout dopo il secondo layer FC	0.0	0.0	0.5
Learning rate	0.0001	0.01	0.0001
Random flip	False	False	False
Random rotation	$[-0.05, 0.05]$	0.0	0.0
Random contrast	0.0	0.0	$[-0.02, 0.02]$

A seguito di questa ottimizzazione tutti i modelli sono stati addestrati nuovamente con i parametri migliori individuati. I nuovi modelli non hanno evidenziato alcuna differenza, se non nell'architettura, rispetto ai modelli base inizialmente creati. In alcuni casi i risultati sono stati anche lievemente inferiori.

3.6 Rete from scratch

Per la realizzazione della rete from scratch si è deciso di mantenere il formato originale previsto dal dataset, ovvero immagini RGB con dimensione 96x96px. Allo scopo di selezionare l'architettura migliore si è definito uno spazio di possibili iperparametri che definissero le caratteristiche dei layer convoluzionali e dei layer fully connected. Tramite il processo di ottimizzazione bayesiana si è dunque proceduto a individuare la struttura ottimale. Oltre ai parametri appena descritti, il medesimo approccio è stato adoperato anche per stabilire il miglior set di funzione di attivazione, algoritmo di ottimizzazione e learning rate. Di seguito l'elenco dei parametri sottoposti al processo di ottimizzazione e dei rispettivi possibili valori:

- Algoritmo di ottimizzazione: [SGD; RmsProp; Adam]
- Learning Rate: $[1e-2;\ 1e-3;\ 1e-4]$
- Funzione di attivazione: [ReLu; Leaky-ReLu]
- Layer Convoluzionali:
  - Numero di blocchi convoluzionali (layer Conv2D + MaxPool2D): $[1; 2; 3]$
  - Dimensione del kernel (diversa per ciascun layer): $[(3, 3); (5, 5)]$
  - Numero di filtri (diverso per ciascun layer): $[32; 64; 128; 256]$

  - Probabilità dropout alla fine dei blocchi convoluzionali: $[0; 0.2; 0.5]$

- Layer Fully Connected:
  - Aggiunta di un layer fully connected prima dell'output layer: [True; False]
  - Dimensione del layer: $[64; 128; 256]$
  - Probabilità dropout prima dell'output layer: $[0; 0.2; 0.5]$

Si ottiene quindi uno spazio di 105.120 possibili configurazioni, un numero decisamente troppo elevato per essere approcciato con tecniche come GridSearch. L'algoritmo di ottimizzazione utilizzato invece consente di esplorare lo spazio dei parametri senza dover testare tutte le possibilità. È stata dunque avviata una ricerca del set di iperparametri migliore, limitando a 50 il numero massimo di configurazioni da valutare. L'architettura risultante dal processo di ottimizzazione, mostrata in figura 8, ha consentito di ottenere una validation loss pari a 0.276, e una validation accuracy pari a 0.886, usando il trainset *'small'*.

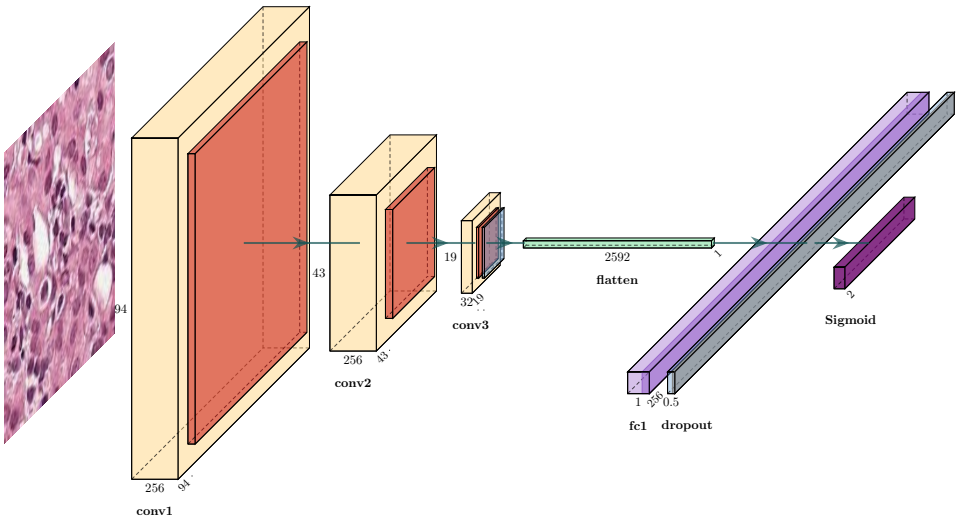

Figura 8: Architettura ottimale della rete from scratch

Il modello consiste di tre blocchi convoluzionali e un layer fully connected, definiti come segue:

- Primo blocco: un layer Conv2D con filter bank di `dimensione=(3,3,256)` e nessun padding, seguito da un layer di MaxPooling di `dimensione=2`
- Secondo blocco: un layer Conv2D con filter bank di `dimensione=(5,5,256)` e nessun padding, seguito da un layer di MaxPooling di `dimensione=2`
- Terzo blocco: un layer Conv2D con filter bank di `dimensione=(3,3,32)` e nessun padding, seguito da un layer di MaxPooling di `dimensione=2`
- Layer di dropout con `rate=0.5`
- Layer fully connected con 256 neuroni, preceduto da un layer di flattening per adattare la dimensione dell'output dei blocchi convoluzionali all'input dei fully connected.
- Layer di dropout con `rate=0.5`
- Layer fully connected finale con 2 neuroni che mappa alla dimensione dell'output

L'architettura così definita è quindi composta da 2.383.906 parametri, tutti trainabili. Per tutti i layer è stata usata ReLu come funzione di attivazione, con `learning rate=`$1e-4$. Il miglior algoritmo di ottimizzazione si è invece dimostrato essere Adam. Il modello ha permesso di ottenere, una volta addestrato usando il trainset *'medium'*, una validation loss pari a 0.163 e una validation accuracy del 93.6%, arrestandosi dopo 27 epoche di cui le tre finali eliminate automaticamente dall'EarlyStopping, che provvede a ripristinare i pesi migliori. In figura 9 si riportano i grafici dell'andamento di loss e accuracy durante l'addestramento, le epoche eliminate dall'EarlyStopping sono riportate da linee tratteggiate per facilitare il distinguo.
Una volta individuata l'architettura migliore, si è sperimentata l'aggiunta dei layer di Data Augmentation descritti nelle sezioni 3.3 e 3.5.4. È stata avviata quindi un'ulteriore ricerca basata su ottimizzatore Bayesiano, che dopo 10 configurazioni diverse testate si è arrestata senza apportare alcun miglioramento alle performance della rete. Si è dunque deciso di tenere come modello from scratch definitivo quello addestrato senza i suddetti nuovi layer.

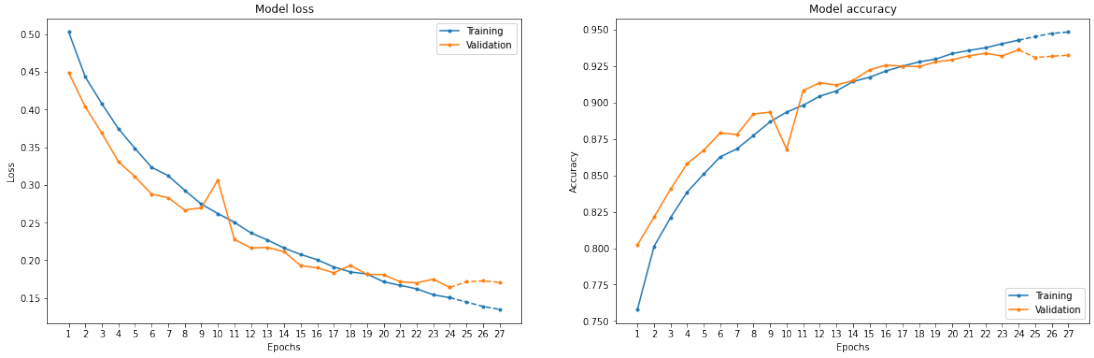

Figura 9: Andamento di loss e accuracy durante l'addestramento della rete from scratch

4 Risultati e valutazione

Per valutare le capacità predittive dei diversi modelli creati si sono utilizzati gli appositi test set creati in fase di ridimensionamento del dataset. Tali test set permettono di verificare e confrontare le performance delle varie architetture create. Nel valutarne la bontà, oltre alle classifiche metriche di *loss* e *accuracy*, si è deciso di considerare anche la *recall* della classe positiva 1, quella delle immagini di tumori. A parità delle altre metriche viene dunque preferito il modello con recall più alta. Difatti, si è scelto di preferire la presenza di possibili falsi positivi, che richiederanno analisi più approfondite, piuttosto che dei falsi negativi che potrebbero avere ripercussioni più gravi su cure e guarigione della persona interessata. Inoltre si è valutato anche il risultato ottenuto per l'area sottesa alla curva (AUC), in quanto è la metrica valutata nella challenge Kaggle.
Il modello migliore è risultato essere quello from scratch, implementato ad hoc per questo problema. Tale modello permette di ottenere score molto validi, con un numero limitato di parametri e con predizioni molto veloci. Dalla figura 10, riportante i risultati ottenuti dal suddetto modello per il dataset medio, si nota come i valori di accuracy e di recall siano pari a 0.93 e dunque soddisfacenti. La stessa metrica di AUC restituisce un risultato buono, tuttavia si nota la presenta di 595 casi su 9000 di falsi negativi. Il modello è stato utilizzato sul test set non etichettato fornito da Kaggle ai fini della challenge e ne è risultata una AUC pari a 0.88.

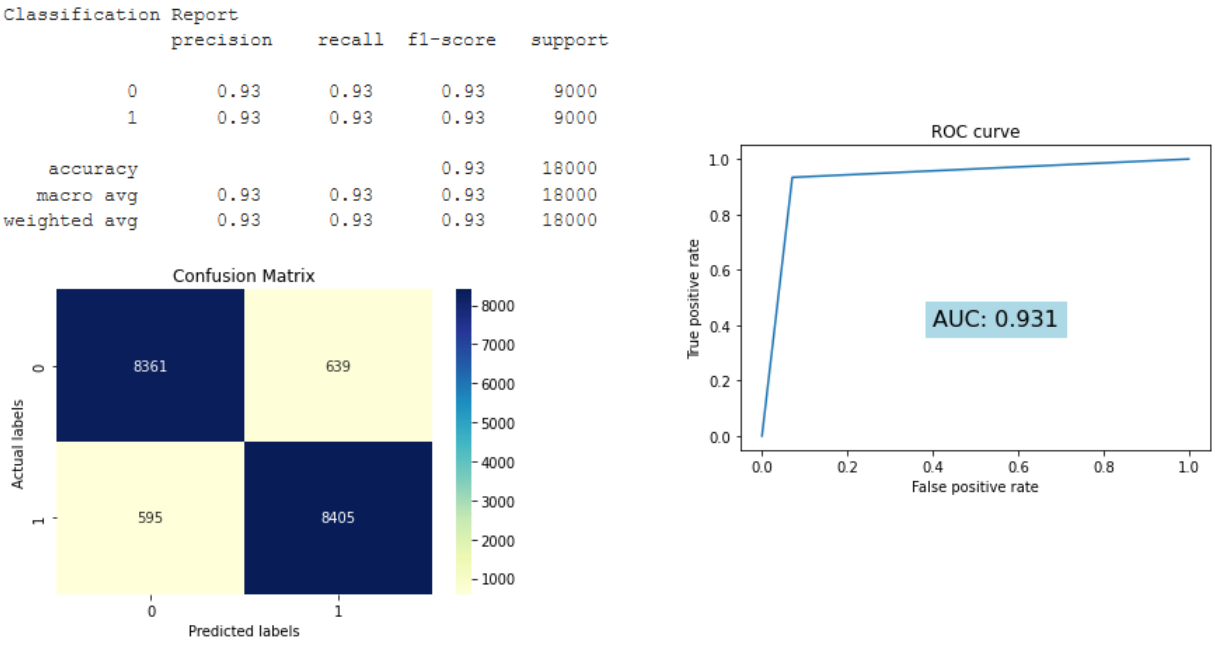

Figura 10: Risultati del modello from scratch sul test set 'medium'.

I modelli di fine tuning non hanno invece riportato performance così elevate, rivelandosi dunque meno efficaci per questo particolare scopo. Nel caso migliore, ottenuto con l'architettura DenseNet-121 le cui metriche sono riportate in figura 11, si arriva infatti a performance molto inferiori, con accuracy pari a 0.79, oltre 2000 casi di falsi negativi e recall per la classe positiva pari a 0.77.

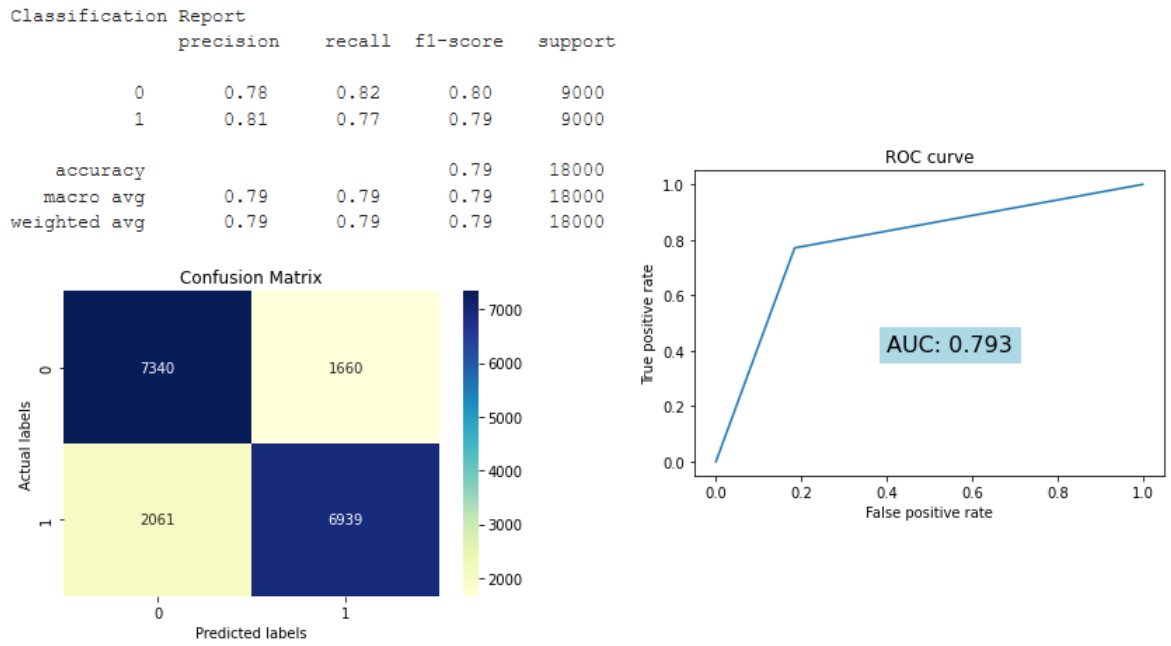

Figura 11: Risultati del modello DenseNet-121 sul test set 'medium'.

Mentre le altre due architetture, VGG16 e MobileNet-V2, ottengono performance comparabili con un accuracy finale di circa 0.75% sul test set più piccolo.
Un'altra importante metrica che si è deciso di valutare è stata il tempo di predizione per una singola immagine. Si è scelto di analizzare anche questo aspetto in quanto predizioni veloci

potrebbero rendere possibile il riconoscimento del cancro in real-time sfruttando le immagini che arrivano durante analisi come colonscopie [13]. In tal senso si è mossa da tempo anche l'Unione Europea con il progetto MagentiqEye [14]. In tabella 2 sono riportati i tempi di predizione dei diversi modelli, ancora una volta il modello from scratch è risultato essere il più performante rispetto a tutti gli altri testati, in virtù del minor numero di parametri. [1]

Tabella 2: Tempo di predizione di una singola immagine

Modello	**Tempo (secondi)**
From scratch	0.109
MobileNet-V2	0.157
VGG16	0.193
DenseNet-121	0.218

5 Conclusioni

Il lavoro svolto ha consentito di realizzare un modello per il riconoscimento di tumori con risultati che, considerando la complessità del task, possono considerarsi incoraggianti. Si può dunque concludere che soluzioni come quella descritta possano rappresentare non solo in futuro, ma già oggi un valido aiuto per i patologi. Questo non significa tuttavia che le performance ottenute non siano migliorabili. In particolar modo, anche nel caso del modello migliore ottenuto, è ancora presente un numero consistente di falsi negativi che, se fossero assenti, consentirebbero di affidarsi molto di più alle predizioni effettuate dalla rete. La presenza di questi errori si deve, tra le altre problematiche, alla presenza di alcune scansioni falsate (completamente bianche o nere) che potrebbero aver influenzato in negativo l'addestramento. Sarebbe anche opportuno affidarsi ad un esperto di dominio in modo da conoscere meglio le peculiarità delle immagini e sfruttare consapevolmente le loro caratteristiche.
Occorre inoltre ricordare che, ai fini del progetto, si è utilizzata soltanto circa la metà dei dati messi a disposizione. Utilizzare l'intero dataset, compatibilmente con le risorse hardware a disposizione, permetterebbe probabilmente di ottenere classificatori più performanti e robusti.
Uno dei possibili sviluppi futuri è quello di concentrarsi maggiormente sulla tematica del tempo di predizione, cercando di ottenere un sistema real-time che riduca al minimo la durata della valutazione sulla singola immagine e consenta così di applicare questi modelli in uno spettro più ampio di analisi. Per fare ciò si dovrebbero dunque considerare strategie di compressione della rete, a patto di non inficiare sulle sue performance che, considerando il dominio preso in esame, sono comunque da considerarsi prioritarie.

[1] Il tempo di predizione è stato calcolato come media rispetto al tempo necessario per predire un intero batch a parità di potenza computazionale. Inoltre è strettamente dipendente dalle risorse computazionali a disposizione.

Riferimenti bibliografici

[1] FDA (US Food and Drug Administration), "FDA allows marketing of first whole slide imaging system for digital pathology," 4 2017. [Online]. Available: https://www.fda.gov/news-events/press-announcements/fda-allows-marketing-first-whole-slide-imaging-system-digital-pathology

[2] U. Muehlematter, P. Daniore, and K. Vokinger, "Approval of artificial intelligence and machine learning-based medical devices in the usa and europe (2015–20): a comparative analysis," *The Lancet Digital Health*, vol. 3, 01 2021.

[3] "State of healthcare q1'21 report: Investment & sector trends to watch," 4 2021. [Online]. Available: https://www.cbinsights.com/research/report/healthcare-trends-q1-2021/

[4] B. S. Veeling, "Patchcamelyon (pcam)," https://github.com/basveeling/pcam, 2020.

[5] PCCam, "Histopathologic cancer detection," 2019. [Online]. Available: https://www.kaggle.com/c/histopathologic-cancer-detection/data

[6] P. C. Team, *Python: A dynamic, open source programming language*, Python Software Foundation, 2015. [Online]. Available: https://www.python.org/.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[8] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[9] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[13] K.-S. Lee, S.-H. Son, S.-H. Park, and E. Kim, "Automated detection of colorectal tumors based on artificial intelligence," *BMC Medical Informatics and Decision Making*, vol. 21, 02 2021.

[14] Magentiq Eye Ltd. (2015) Magentiq eye ltd. [Online]. Available: https://www.magentiq.com/

[15] V. Messina and S. Bianco, "Dispense e slide del corso advanced machine learning," 2021.