Leukemia Classification - Report

Martina Cavallucci - Matr: 919588
martina.cavallucci@studio.unibo.it
Simone Gollinucci - Matr: 895245
simone.golinucci@studio.unibo.it

10 settembre 2021

Indice

1 Descrizione problema

Il problema che si vuole risolvere con tale progetto è la classificazione di linfociti nel sangue per diagnosticare la presenza della *leucemia linfoblastica acuta*.

La **leucemia linfoblastica acuta** (ALL) [6] è una malattia tumorale del sangue a rapida progressione che inizia quando i linfociti, un sottotipo di globuli bianchi, vanno incontro a livello del midollo osseo a trasformazione neoplastica, con conseguente moltiplicazione incontrollata e progressivo accumulo. I sintomi principali della *leucemia linfoblastica acuta* sono determinati dall'aumento di blasti (cellule malate) nel sangue e nel midollo osseo, con conseguente riduzione delle cellule sane.

E' una malattia relativamente rara: in Italia si registrano circa 1,6 casi ogni 100.000 maschi e 1,2 casi ogni 100.000 femmine, cioè circa 450 nuovi casi ogni anno tra gli uomini e 320 tra le donne [6].

La *leucemia linfoblastica acuta* è però **il tumore più frequente in età pediatrica**: rappresenta l'80% delle leucemie e circa il 25% di tutti i tumori diagnosticati tra 0 e 14 anni. L'incidenza raggiunge il picco tra i 2 e i 5 anni e poi cala con l'aumentare dell'età (il 50% di tutti i casi viene diagnosticato entro i 29 anni).

Per diagnosticare una *leucemia linfoblastica acuta* si utilizzano vari tipi di esami del sangue e del midollo osseo. Il sangue viene raccolto con un normale prelievo; il midollo osseo invece viene prelevato con una siringa dalle ossa del bacino, in anestesia locale. Tra le principali analisi per la diagnosi sono presenti [2]:

- **L'emocromo**, che misura i vari tipi di cellule presenti nel sangue;
- **L'esame al microscopio delle cellule del sangue e del midollo osseo**, che permette di contare le cellule leucemiche e di identificarne la struttura;
- **Le analisi del sangue** che misurano il funzionamento degli organi, per capire se e quali organi sono stati danneggiati dalla malattia;
- **Test che misurano il metabolismo del tumore**, e quindi la sua velocità di crescita;

Il progetto mira ad aiutare il processo di **analisi morfologica** che consiste nell'esaminare al microscopio ottico le caratteristiche morfologiche delle cellule midollari presenti in campioni di aspirato midollare e sangue periferico. Questa analisi fornisce importanti informazioni per la diagnosi e la definizione della fase di malattia.

2 Dataset

Il dataset utilizzato è reperibile tramite la piattaforma Kaggle [3]. Esso si compone in totale di 15.135 immagini di dimensioni 450x450 riferite a 118 pazienti. Il dataset è etichettato secondo le due classi: **Normal cell(Hem)** ovvero cellule sane e **Leukemia blast(All)** ovvero cellule malate.

Il dataset è suddiviso con la seguente struttura:

- `training_data/`
  - `fold_0/`
    - `all/`
    - `hem/`
  - `fold_1/`
    - `all/`
    - `hem/`
  - `fold_2/`
    - `all/`
    - `hem/`
- `validation_data/`
  - `C-NMC_test_prelim_phase_data/`
  - `C-NMC_test_prelim_phase_data_labels.csv`
- `testing_data/`
  - `C-NMC_test_final_phase_data/`

Le immagini sono distribuite nel seguente modo:

- **Training**: 10661 immagini di cui 7272 di cellule malate e 3389 sane. Le istanze del traning set sono sbilanciate rispetto alle classi.
- **Validation**: 1867 immagini di cui 1219 di cellule malate e 648 sane.
- **Test**: 2586 immagini **non etichettate**.

Siccome le immagini del Test set non sono etichettate e non si ha la possibilità di sottomettere i risultati alla challenge in quanto conclusa, abbiano deciso di eliminarle e di promuovere alcune del training set e del validation set come test set, assicurandoci che nelle fasi di training della rete questa non venga mai addestrata con tali immagini.

Di seguito si riportano degli esempi di immagini di entrambe le classi, Hem e All, presenti nel dataset.

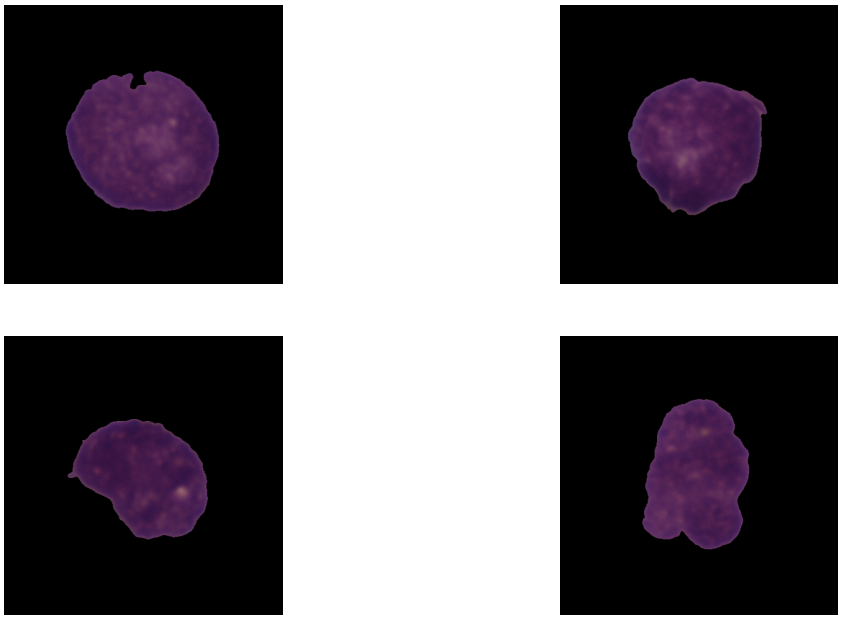


Figura 1: Esempi di immagini del dataset della classe Hem

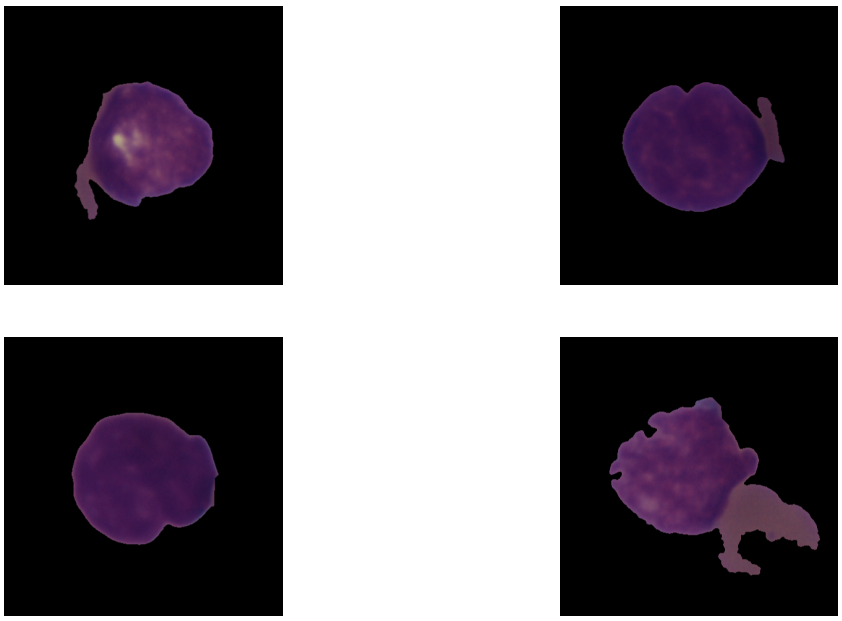


Figura 2: Esempi di immagini del dataset della classe All

3 Riferimento contesto applicativo

Per lo studio del contesto applicativo si è fatto riferimento al paper: *"Computer-Aided Acute Lymphoblastic Leukemia Diagnosis System Basedon Image Analysis"* [1]. In tale articolo si descrive un sistema di segmentazione dei linfociti e della loro classificazione in cellule sane o malate.

3.1 Diagnostica

I campioni di sangue possono essere osservati e diagnosticati per diverse malattie dai medici. Qualsiasi diagnosi basata sull'uomo soffre di una precisione non standard poiché dipende fondamentalmente dall'abilità del medico; inoltre è inaffidabile da un punto di vista statistico. Esistono attualmente vari sistemi in grado di contare il numero di cellule del sangue in base alla misurazione delle proprietà fisiche e chimiche. Tali sistemi utilizzano un rilevatore di luce che sfruttano la fluorescenza o l'impedenza elettrica per identificare i tipi di cellule. Sebbene i risultati della quantificazione siano precisi, costa molto denaro, inoltre non rileva le anomalie morfologiche delle cellule; pertanto, è necessaria un'analisi del sangue complementare basata sull'immagine microscopica.

3.2 Componenti dell'immagine

Ci sono tre componenti principali dell'immagine del sangue: *globuli rossi, piastrine e globuli bianchi (o leucociti)*. I globuli bianchi svolgono un ruolo importante nel sistema immunitario poiché ci difendono da infezioni e da malattie.

I globuli bianchi possono essere classificati in 2 gruppi: *granulociti* e *agranulociti*. I secondi comprendono i **linfociti** che sono un tipo di cellula prodotta dal tessuto linfoide che passa nel sangue, caratterizzata da un citoplasma basofilo e da un grande nucleo rotondo che occupa quasi tutto il protoplasma. Un esempio di cellula è mostrata in figura 3.

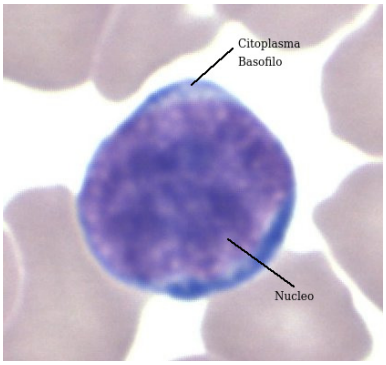


Figura 3: Esempio di componente linfocitica nel sangue

I linfociti sono di forma regolare e hanno un nucleo compatto con bordi regolari e continui. Al contrario, i linfociti che soffrono di ALL sono chiamati **linfoblasti**. Sono

di forma irregolare e contengono piccole cavità nel citoplasma (detti vacuoli) e particelle sferiche all'interno del nucleo (detti nucleoli), tanto maggiori sono i cambiamenti morfologici tanto maggiore è la gravità della malattia indicata.

Sebbene i leucociti possano essere facilmente identificati (poiché appaiono più scuri "viola" rispetto allo sfondo), ma a causa delle ampie variazioni nelle loro forme, dimensioni e bordi, l'analisi e l'elaborazione diventano molto complicate.

4 Soluzioni esistenti

Di seguito si riportano le principali soluzioni già implementate in letteratura, sulla pagina Kaggle del dataset Leukemia Classification e su Github. Tale studio ci permette di capire dove poter migliorare e conoscere le tecniche utilizzate da altri e i risultati ottenuti.

4.1 Soluzione in letteratura

In riferimento al paper *"Classification of Normal versus Leukemic Cells with Data Augmentation and Convolutional Neural Networks"* [4] si è approfondita la metodologia di Data Augmentation e di vari modelli che hanno apportato i migliori risultati.

In tale paper vengono addestrare diverse reti come: VGG16, VGG19 e Xception e vengono utilizzate diverse tecniche di Data augmentation per bilanciare i dati del training set e validation set. Il dataset utilizzato *C-NMC 2019 Dataset* coincide con quello usato per tale progetto che consiste in 15135 immagini raccolte da 118 pazienti di dimensioni 450 x 450. Siccome le immagini appartengono alla challange *"Classifi cation of Normal versus Malignant Cells in B-ALL White Blood Cancer Microscopic Images"* organizzata da *SBILab*. Nella soluzione proposta nel paper sono stati utilizzati soltanto i dati del `training_data/` e `validation_data/`, suddivisi in train set, valid set e test set con una rapporto 6:3:1. La parte fondamentale che abbiamo appreso da tale paper è quella di Data augmentation descritta di seguito.

Data Augmentation

Il set di dati originale era sbilanciato e, per questo motivo, è stato impiegato l'aumento dei dati per bilanciare i set. Tale tecnica non è utilizzata sui dati del test set. Le trasformazioni applicate sono:

- **Mirroring**: duplicazione riflessa di un oggetto che appare quasi identico, ma è invertito nella direzione perpendicolare alla superficie dello specchio;
- **Rotazione**: rotazione dell'immagine con scelta casuale tra 20 e 340 gradi;
- **Sfocatura gaussiana**: applicazione di un filtro con pesi che seguono una distribuzione gaussiana utilizzata per ridurre rumore nell' immagine;
- **Shearing**: trasformazione affine dell'immagine che porta ad una variazione geometrica dell'oggetto sugli assi;
- **Salt and pepper noise**: all'immagini vengono inseriti in modo casuale dei pixel bianchi e dei pixel neri.

Preprocessing applicato

Dato che i linfociti presenti nelle immagini hanno una asse maggiore di 223 $\pm$ 43 pixel in media, hanno ritagliato da ogni lato 100 pixel, riducendo la dimensione dell'immagine originale da 450×450 a 250×250.

Eseguendo questa operazione, è stato possibile ridurre le dimensioni dell'immagine senza perdere un'area sostanziale delle celle. Dopo il ritaglio, i pixel sono stati convertiti in float dividendo i loro valori per 255 e sottraendo la media del canale.

Training soluzione in letteratura

Durante il training si è usato come layer di output una *softmax*, come loss la *cross-entropy* che restituisce una probabilità di appartenenza alle due classi. La loss è stata ottimizzata con l'algoritmo Adam, inoltre si è applicata una regolarizzazione L1 e L2.

La regolarizzazione L1 è la somma dei valori assoluti della matrice dei pesi e la regolarizzazione L2 è la somma di tutti i valori dei pesi al quadrato della stessa matrice. In questo caso si combinano queste due norme in un'unica regolarizzazione con λ. Perciò la loss può essere riscritta come in figura 4 dove W_i è la matrice dei pesi con coefficienti i, λ_{L1}, λ_{L2} e si considera $\lambda = \lambda_{L1} = \lambda_{L2}$. Il termine di regolarizzazione λ è stato adeguato per ottenere la più bassa loss sul validation.

$$\text{Loss} = \text{Cross-Entropy} + \lambda_{L_1} \sum_i |W_i| + \lambda_{L_2} \sum_i W_i^2$$

$$\text{Loss} = \text{Cross-Entropy} + \lambda \left(\sum_i |W_i| + \sum_i W_i^2 \right)$$

Figura 4: Funzione di loss con regolarizzazione L1 e L2

Risulati

I migliori risultati si sono ottenuti addestrando completamente, senza aver inizializzato i pesi della rete VGG16 ottenendo un F1-score di 92.60%, precision di 91.14%, recall di 94.10%. Il grafico dell'andamento dell'accuracy si può osservare in figura 5.

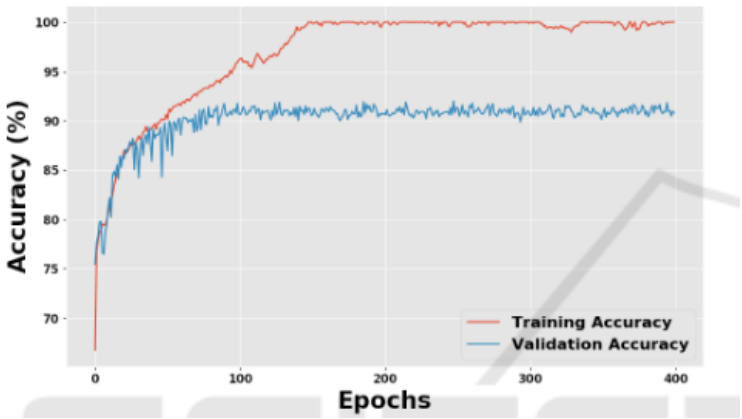


Figura 5: Andamento accuratezza VGG16

4.2 Soluzione su Kaggle

- **VGG19**

Link	https://www.kaggle.com/gauravrajpal /leukemia-classification-v1-1-vgg19-65-45
Accuratezza:	65,45%
Ottimizzatore	Adam(1e-5)
Loss	binary_crossentropy
Epoche	500 con Early Stopping

In particolare in tale progetto si sono utilizzate le seguenti tecniche:

- **Data Augmentation** tramite flip orizzontatale, verticale e uno zoom di 0,2%;
- Le immagini vengono ridimensionate ad una dimensione di 128x128.
- Il modello VGG19 viene creato caricando i pesi di **Imagenet**. **VGG19** è una variante di VGG che consiste in 19 layer (16 convoluzionali, 3 layer fully connected, 5 layer Max Pool). Tali layer non vengono nuovamente riaddestrati ma viene aggiunto un layer Flatten, due layer fully connected e un layer di output con funzione di attivazione Sigmoide per effettuare la classificazione binaria.

- **Xception**

Link	https://www.kaggle.com/gauravrajpal/ leukemia-classification-v1-2-xception-66-52
Accuratezza:	66.52%
Ottimizzatore	Adam(1e-5)
Loss	binary_crossentropy
Epoche	500 con Early Stopping

In particolare in tale progetto si sono utilizzate le seguenti tecniche:

- Vengono effettuate le stesse operazioni di Data Augmentation del modello precedente;
- Le immagini vengono ridimensionate ad una dimensione di 128x128.
- Il modello **Xception** viene creato caricando i pesi di **Imagenet**.
- Xception è un'estensione di Inception dove l'architettura di reti convoluzionali è basata interamente su strati di convoluzione separabili in profondità. E' composta da 36 layer convoluzionali ed è composta da 14 moduli ognuno dei quali ha intorno connessioni lineari residue tranne il primo e l'ultimo. Tali layer non vengono nuovamente riaddestrati ma vengono aggiunti i layer come nella soluzione precedente.

– Il batch size utilizzato è di 512.

- **InceptionV3**

Link	https://www.kaggle.com/gauravrajpal/ leukemia-classification-v1-3-inceptionv3-65-29
Accuratezza:	65.29%
Ottimizzatore	Adam(1e-5)
Loss	binary_crossentropy
Epoche	500 con Early Stopping

In particolare in tale progetto si sono utilizzate le seguenti tecniche:

– Vengono effettuate le stesse operazioni di Data Augmentation del modello precedente;

– Le immagini vengono ridimensionate ad una dimensione di 128x128.

– Il modello **InceptionV3** viene creato caricando i pesi di **Imagenet**, come nel caso precedente viene effettuata la scelta di non riaddestrare alcun layer del modello utilizzato.

– Il batch size utilizzato è di 512.

4.3 Soluzioni da Github

- **VGG16 e Inception V3**

Link	https://github.com/egeoguzman/ Leukemia-Classification
Accuratezza:	82% (VGG19) - 87%(InceptionV3)
Ottimizzatore	Adam
Loss	categorical_crossentropy
Epoche	100 con Early Stopping

In particolare in tale progetto si sono utilizzate le seguenti tecniche:

– Non vengono fatte operazioni di Data Augmentation

– Per gestire le classi sbilanciate vengono inseriti dei pesi che danno maggior importanza alla classe con minor istanze, tale soluzione però commette ancora molti errori sulla classe Hem.

- **CNN**

Link	https://github.com/rohan1198/Leukemia-Classification-using-Deep-Learning
Accuratezza:	93%
Ottimizzatore	Adam
Loss	categorical_crossentropy
Epoche	50 con Early Stopping

In particolare in tale progetto si sono utilizzate le seguenti tecniche:

- I dati vengono bilanciati effettuando Data Augmentation prima e durante l'addestramento tramite i generatori;
- Il modello si compone di livelli convoluzionali ognuno dei quali seguito da un livello di MaxPooling, si fa uso di Batch Normalization.
- Non viene utilizzato il test set del dataset originale, in quanto le etichette non sono presenti.

4.4 Considerazioni su soluzioni esistenti

Riteniamo che le soluzioni disponibili in Kaggle non siano ottimali in quanto viene effettuato un trasfer-learning nella quale vengono "congelati" i layer di feature extraction della rete con i pesi di ImageNet, ciò porta ad avere meno accuratezza data dal fatto che la rete è addestrata su un altro dominio. Altro problema è dato dal forte sbilanciamento del dataset che molte delle soluzioni non trattano. Abbiamo perciò dato maggior peso alle informazioni apprese dalle soluzioni in letteratura cioè abbiamo pensato di effettuare un modello costruito da zero con l'applicazione di Data Augmentation per cercare di risolvere il problema di sbilanciamento e in modo da "aiutare" la rete nell'apprendimento di feature. Inoltre abbiamo voluto sperimentare un modello recente con EfficentNet e studiare il suo comportamento adottando un addestramento senza caricamento dei pesi come viene effettuato nel lavoro di J.E.M. Oliveira et al. [4]. Dalle soluzioni di Github e Kaggle abbiamo acquisito comunque tecniche per l'augmentation e preprocessig e inoltre ci sono servite per capire quale modello creare.

5 Soluzione realizzata

Le soluzioni realizzate sono state 2: la prima soluzione, prendendo come riferimento i lavori in letteratura, è una rete a convoluzione interamente creata e addestrata da zero. Tale soluzione è stata perseguita sulla base di modelli analoghi riscontrati in letteratura che hanno ottenuto buone performance. La seconda soluzione, basata sulla la rete EfficientNet B2, è stata realizzata per verificare le sue prestazioni nel nostro contesto applicativo.

In questa sezione verranno esposte tutte quelle operazioni che sono state effettuate che sono in comune tra le soluzioni. Successivamente verranno presentate le due soluzioni in sezioni dedicate.

5.1 Pre-processing

In questa prima fase, le immagini vengono elaborate per essere compatibili in dimensioni e formato per il classificatore.

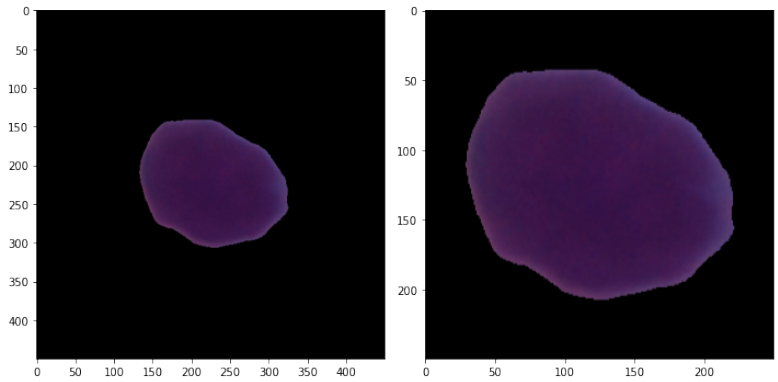


Figura 6: Esempio di immagine tagliata mediante cropping.

Preparazione immagini: Le immagini presenti nel dataset hanno una dimensione di 450 x 450 pixel e tre canali colore. La fase di pre-processing porta la dimensione dell'immagine da quella originaria a quella specificata come input per il classificatore. La riduzione della dimensione viene effettuata mediante **cropping** dell'immagine. Come descritto da J.E.M. de Olivier et al. [4] nel loro lavoro, considerando che l'asse maggiore della cellula nell'immagine ha dimensioni di 223 ± 43 pixel si è scelto di tagliare le immagini fino ad un massimo di 100 pixel per lato.

Il limite di 100 pixel per lato è stato scelto per evitare l'eliminazione di troppe informazioni relative alla cellula, nel caso in cui l'asse maggiore della cellula sia superiore alla dimensione desiderata per l'immagine. Nella figura 6 è presente un esempio di immagine prima e dopo il cropping.

	All	Hem
Train	5089	2427
Validation	2553	1206
Test	849	404

Tabella 1: Numero di immagini nei set dopo aver suddiviso l'intero dataset

Le operazioni di questa fase vengono eseguite su tutte le immagini del dataset. Ovviamente questo passaggio deve essere effettuato anche sulle immagini su cui vogliamo eseguire la classificazione.

Dataset split: Come descritto nella sezione 2 le immagini sono suddivise in 3 set, sebbene il terzo sia inutilizzabile in quanto non etichettato.

Abbiamo quindi deciso di unire tutte le immagini etichettate a nostra disposizione e ripartirle in Train, Validation, Test set con la proporzione a 6:3:1, come è stato fatto J.E.M. de Oliveira et al. in [4]. Nella tabella 1 sono indicati il numero di immagini nei vari set dopo averle divise mediante la proporzione precedentemente definita.

5.2 Data augmentation

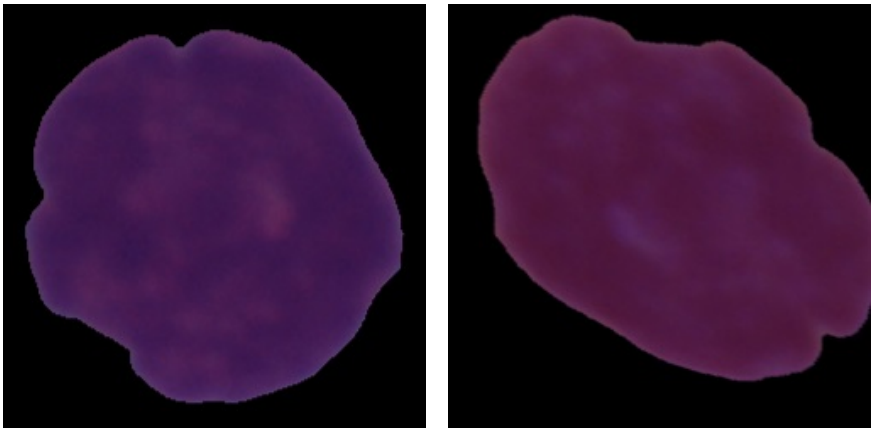
Il dataset a nostra disposizione possiede poche immagini ed è sbilanciato nel numero di istanze tra le due classi, avendo molti più esempi di cellule malate rispetto a quelle sane.

Per questo motivo è stata sfruttata la tecnica di data augmentation. Con la data augmentation abbiamo la possibilità di modificare le immagini in modo tale da generare nuove istanze partendo da quelle disponibili. Come avviene in [4], soltanto il training set e il validation set vengono aumentati, mentre il test set viene lasciato così com'è. Prima di effettuare la data augmentation si creano due liste di immagini divise per classi. Durante la data augmentation si campiona un elemento della lista e lo si aumenta. Questo passaggio viene effettuato fino a raggiungere il numero di istanze desiderate per classe dentro al dataset. Di base nel training set è composto da 10000 immagini per ogni classe, mentre il validation set ha un numero prestabilito di immagini per classe di 2000.

Le trasformazioni applicate all'immagine sono un sottoinsieme di quelle utilizzate in [4]:

1. Rotazione random + flip (orizzontale o verticale);
2. Shearing dell'immagine;
3. Sfocatura con filtro gaussiano.

Utilizzando un generatore random le immagini sono aumentate con probabilità $\frac{1}{3}$: con solo la prima trasformazione o con una combinazione della prima trasformazione con una delle due successive. Di seguito si fornisce un esempio di data augmentation in figura 7, dove 7a è l'immagine originale del dataset mentre 7b è l'immagine aumentata utilizzando flip, rotazione e shearing.



(a) Immagine originale (b) Immagine aumentata

Figura 7: Esempio di data Augmentation

5.3 Funzioni di utilità

Durante l'addestramento si è fatto uso di alcune tecniche come:

- **EarlyStopping**: permette di terminare il training nel caso l'accuratezza sul validation non migliori ulteriormente, evitando situazioni di overfitting;
- **ReduceLROnPlateau**: riduce il learning_rate quando l'accuratezza sul validation set non migliora per 2 epoche consecutive. La riduzione del learning_rate può aiutare il modello a convergere;
- **ModelCheckpoint**: permette di salvare i pesi che raggiungono la miglior accuratezza sul validation set durante l'addestramento;
- **Kernel Regularizers**: i regolarizzatori consentono di applicare penalità sui parametri del layer durante l'ottimizzazione. Queste penalità sono sommate alla loss che la rete ottimizza. In entrambe le soluzioni è stata utilizzata la funzione l1 e l2 che utilizza sia la penalità L1 che L2, come descritto in sezione *Training soluzione in letteratura*.

5.4 Soluzione A

Questa soluzione è stata realizzata partendo dalla proposta dall'utente Github Rohan1198. Il modello presentato da Rohan è molto semplice, basato su convoluzioni per l'estrazione di feature e di Max pooling per la loro successiva riduzione di dimensione. Il problema di questo modello sono il numero di parametri addestrabili che risultano essere più di 95 milioni.

Questa enorme quantità di parametri porta ad un tempo di esecuzione molto elevato per ogni epoca, anche utilizzando un runtime su GPU. Rohan stesso nel suo progetto riconosce tale problematica in quanto Colab dopo 12 ore di esecuzione su GPU scollega automaticamente dal runtime e ne nega l'accesso per circa 24 ore.

Il modello creato vuole quindi cercare di ottenere le medesime prestazioni del modello presentato da Rohan ma sfruttando una minore quantità di parametri addestrabili, riducendo il tempo di training.

5.4.1 Architettura

Partendo dal lavoro di Rohan, abbiamo sviluppato l'architettura del modello in modo iterativo integrando modifiche e ri-addestrando il modello per verificarne l'efficacia. Parte delle modifiche effettuate, sono state attuate grazie al lavoro presentato da Rehman Amjad et al. [5].

Rehman Amjad et al. hanno presentato un classificatore sempre basato su convoluzione con il 97.78% di accuratezza [5]. Il loro classificatore era pensato per riconoscere le cellule malate, divise in tre sottotipi di All, da quelle sane. Pertanto l'output del classificatore è formato da 4 neuroni.

Non essendo di nostro interesse definire il sottotipo della cellula malata il problema viene semplificato, pertanto ci siamo permessi di partire con una versione alleggerita di questo classificatore, che ha portato ad ottimi risultati.

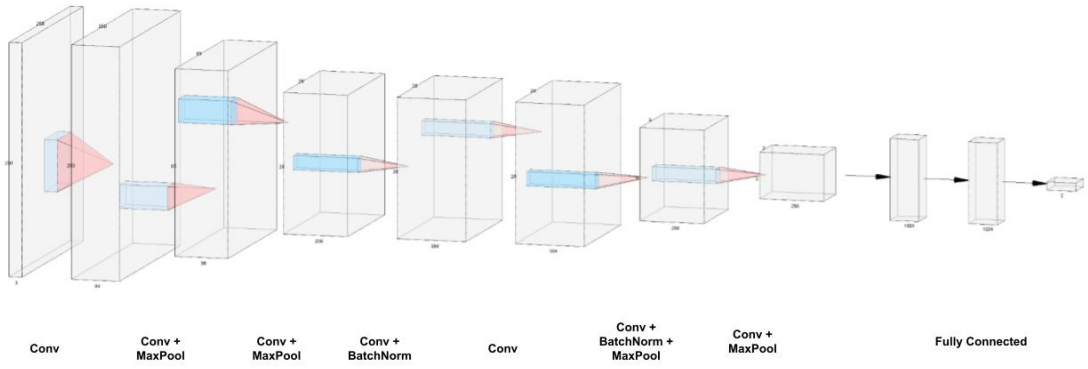


Figura 8: Architettura modello Soluzione A

La rete proposta, mostrata in figura 8, presenta una serie di convoluzioni per l'estrazione di feature, facendo uso di filtri 3x3. Queste convoluzioni sono state divise in moduli come segue:

- *Modulo 1*: Convoluzione e Max pooling;
- *Modulo 2*: Convoluzione e Batch Normalization;
- *Modulo 3*: 2 Convoluzioni seguite da Batch Normalization e da max pooling;
- *Modulo 4*: Convoluzione seguita da Max pooling e da un layer di dropout;
- *Modulo 5*: contiene i layer di classificazione veri e propri. Formato da 3 layer Densi intervallati da layer di Dropout.

Modulo 1: Il modulo tramite convoluzione estrae le feature e tramite il Max Pooling permette la riduzione della loro dimensione preservando l'informazione di interesse.

Modulo 2: Permette un'estrazione di feature apportando poi una normalizzazione dei risultati in modo da ridurre il problema del *Internal Covariate Shift*, che potrebbe sorgere con l'uso di mini-batch.

Modulo 3: Questo modulo è pensato per l'estrazione di feature specifiche per il nostro contesto applicativo. Infatti questo è lo scopo delle due convoluzioni affiancate l'una all'altra. Successivamente viene effettuata una Batch Normalization e un pooling per ridurre la dimensione dei filtri ma mantenendo intatta l'informazione di maggiore rilievo.

Modulo 4: Questo modulo è analogo al primo, ma con l'aggiunta di un layer di Dropout che ha il compito di disabilitare alcuni neuroni, ignorandoli durante il *forward* e *backward*, permettendo di ridurre il rischio di l'overfitting.

Modulo 5: Questo modulo rappresenta il classificatore finale che ha lo scopo di definire se la cellula è malata o meno. Questo modulo è formato da layer fully-connected, dove ogni neurone di un layer è collegato con ogni neurone del layer successivo e viceversa.

Il modulo è composto da 3 layer Densi, alternati con 2 layer di Dropout. I layer Densi sono sono rispettivamente della forma di 1024 neuroni per layer mentre l'ultimo è formato solo da 2 neuroni per la classificazione dell'immagine. Nei layer densi con 1024 neuroni è stato utilizzato un kernel regulizer.

5.4.2 Motivazioni

La creazione di un nuovo modello ci ha permesso una facile sperimentazione, grazie anche al basso numero di parametri addestrabili e ai ridotti tempi richiesti per l'addestramento. Inoltre si voleva verificare una rete che non fosse una delle solite VGG, InceptionV3 o Xception sebbene nel corso delle sperimentazioni anche quelle sono state prese in considerazione.

5.4.3 Training

Per il training sono state prese tutte le immagini del training set e sono state sottoposte al classificatore in batch da 16 istanze. Alla fine di ogni epoca il classificatore è stato valutato utilizzando le immagini del Validation set.

Sulla base del risultato della valutazione della epoca, veniva deciso se abbassare o meno il valore del learning rate. L'idea è partire con un learning rate alto per convergere più velocemente a soluzione, e di ridurlo progressivamente fino a raggiungere una migliore convergenza. La riduzione del learning rate è effettuata per mezzo della funzione di utilità *ReduceLROnPlateau*.

Per il modello è stato scelto di utilizzare come optimizer l'algoritmo **Adam** e come loss function è stata selezionata la funzione *Categorical crossentropy* che permette di calcolare la loss in classificatori multi classe. Infatti sebbene il nostro problema fosse binario, e pertanto sufficiente un solo neurone nel layer di output, è stato deciso di utilizzare 2 neuroni, uno per classe, che fossero in grado di restituire anche la confidenza della classificazione.

Iperparametri

Gli iperparametri sono dei valori assegnati alle varie funzioni e parti del modello che sono definiti a tempo di compilazione e non vengono modificati a tempo di esecuzione. Nel nostro modello ci sono diversi iperparametri che dovranno essere considerati. I più importanti sono:

- Loss function: Categorical crossentropy;
- Ottimizzatore: Adam;
- Batch size: 16;
- Numero di epoche: 50;
- Learning rate: inizializzato a $1E^{-4}$ e ridotto tramite la funzione di utilità *ReduceLROnPlateau* con i seguenti parametri:
  - factor: fattore di riduzione, impostato a 0.5;
  - min_lr: valore minimo che può assumere il learning rate, impostato a $1E^{-6}$.
- Metrics: Accuracy;
- Kernel regularizer: impostata con la funzione l1_l2 con peso $7E^{-4}$ sia per la componente L1 sia per la componente L2.

5.5 Soluzione B (EfficientNetB2)

Il modello EfficientNet è stato proposto nel 2019 da Google AI [7], con tale tecnica si propone un metodo allo stato dell'arte ed efficiente. In particolare questa tecnica ripensa al modo in cui ridimensionare le reti neurali convoluzionali in termini di larghezza, profondità e risoluzione delle immagini. L'aumento di queste caratteristiche porta a benefici ma il modello presenta molti parametri e quindi non è efficiente. In EfficientNet i modelli vengono ridimensionati gradualmente.

Lo studio che sta dietro a tale tecnica si basa sul bilanciare le dimensioni di larghezz,/profondità e risoluzione della rete e si afferma che tale equilibrio può essere raggiunto ridimensionando ciascuno con un rapporto costante. La differenza tra la pratica convenzionale nello *scaling up* delle CNN è che quest'ultima scala arbitrariamente i 3 fattori.

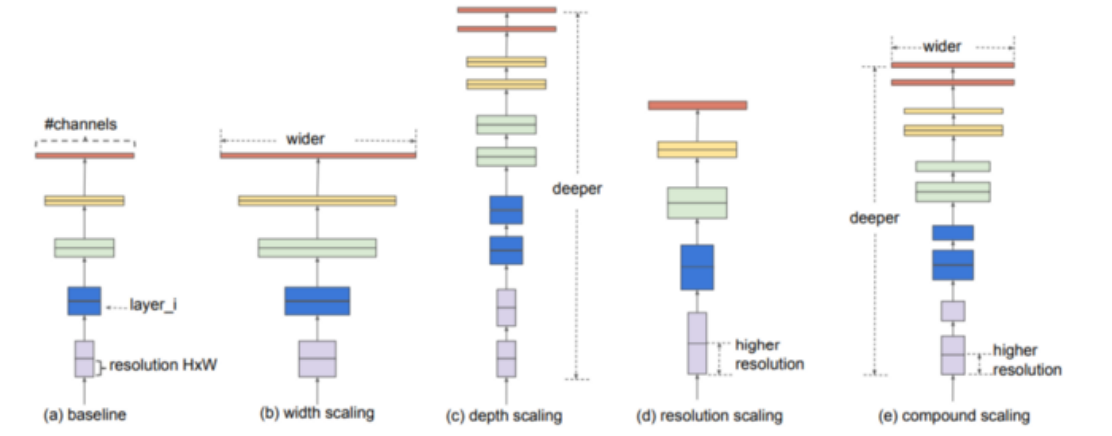


Figura 9: Model Scaling

In figura 9 si mostra la differenza tra il metodo di scala di EfficientNet e quello utilizzato nei metodi convenzionali. In particolare in figura a viene mostrato un esempio base di rete neurale. Nelle figure b,c,d viene applicato un *Compound Scaling* nella quale si aumenta rispettivamente la larghezza, la profondità e la risoluzione, infine in figura e viene mostrato un esempio che scala uniformemente sulle tre dimensioni con un rapporto costante.

Compound scaling

Il metodo di ridimensionamento utilizza un coefficiente ø per ridimensionare insieme lunghezza, profondità e risoluzione. Di seguito in figura 10 si riporta la formula per il ridimensionamento.

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

Figura 10: Compount scaling formula

Dove alfa, beta e gamma sono moltiplicatori di scala e possono essere ottenuti utilizzando una *Grid Search*. ø è un coefficiente espresso dall'utente che controlla le risorse.

5.5.1 Performance su ImageNet

Come si mostra in figura 11 i modelli EfficientNet hanno prestazioni superiori alle altre reti convoluzionali sulla classificazione ImageNet. In particolare EfficientNet B7 supera la miglior accuratezza esistente con Gpipe del 2018 ma utilizzando 8,4 volte parametri in meno ed eseguendo 6,1 volte più veloce.

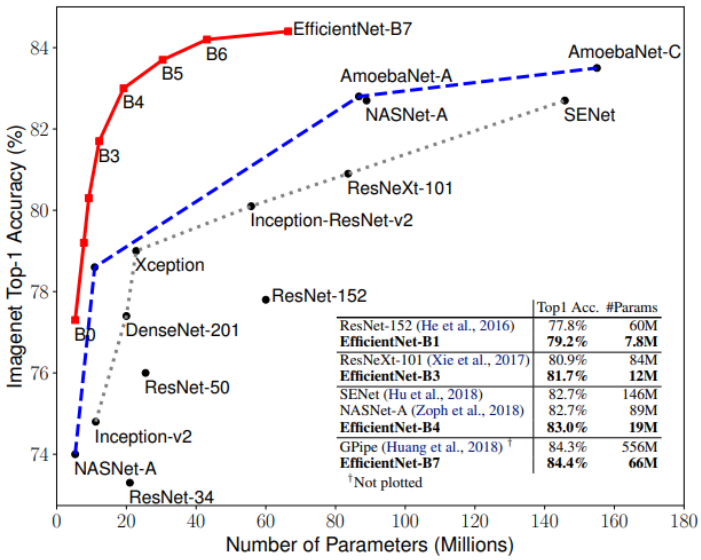


Figura 11: Grandezza del modello vs Accuratezza su ImageNet

5.5.2 Architettura

Esistono 8 modelli EfficientNet da B0 a B7 che differiscono per dimensioni e quindi per numero di parametri. L'architettura del modello EfficientNetB2 segue la struttura mostrata in figura 12. Si precisa che i rettangoli in figura per errore vengono chiamati moduli ma si riferiscono ai sotto-blocchi richiamati dallo stesso colore dei sotto-blocchi descritti in figura 14.

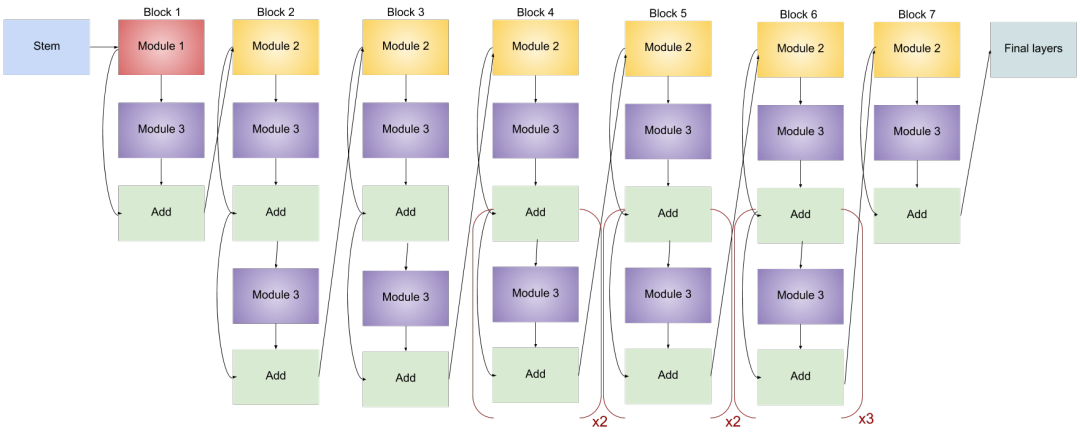


Figura 12: Architettura EfficientNetB2

In particolare il primo blocco è chiamato *Stem* ed è presente in tutti e gli 8 modelli.
Il blocco *Stem* si compone dei seguenti layer mostrati in figura 16. Dopo tale blocco tutti i modelli contengono 7 blocchi, essi hanno un numero variabile di sotto-blocchi in cui il numero aumenta man mano che si passi da EfficentNetB0 ad EfficientNetB7.

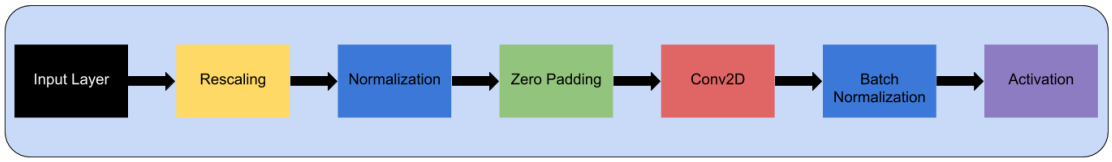


Figura 13: Stem Block

I 7 sotto-blocchi si compongono in tali moduli mostrati in figura 15, nella quale:

- Il **Modulo 1**: è usato come punto di inizio per i successivi sotto-blocchi;
- Il **Modulo 2**: è usato come punto di partenza per tutti i sotto-blocchi eccetto il primo;
- Il **Modulo 3**: esso è connesso a tutti i sotto-blocchi come skip-connection;
- Il **Modulo 4**: esso è utilizzato per combinare la skip connection nel primo sotto-blocco;

- Il **Modulo 5**: ogni sottoblocco è collegato al suo sottoblocco precedente in una skip-connection e vengono combinati usando questo modulo.

Tali moduli vengono combinati per formare sotto-blocchi nel modo graficato in figura 14.

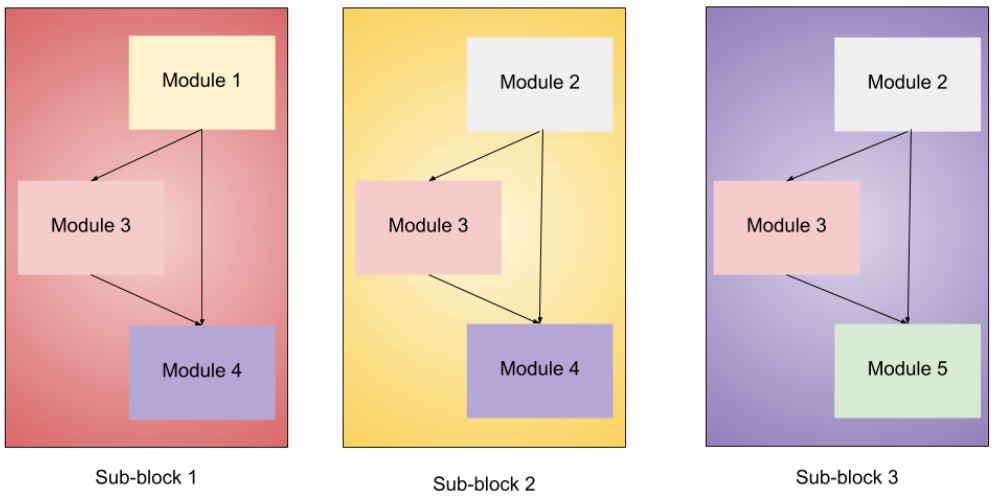


Figura 14: Struttura sotto-blocchi

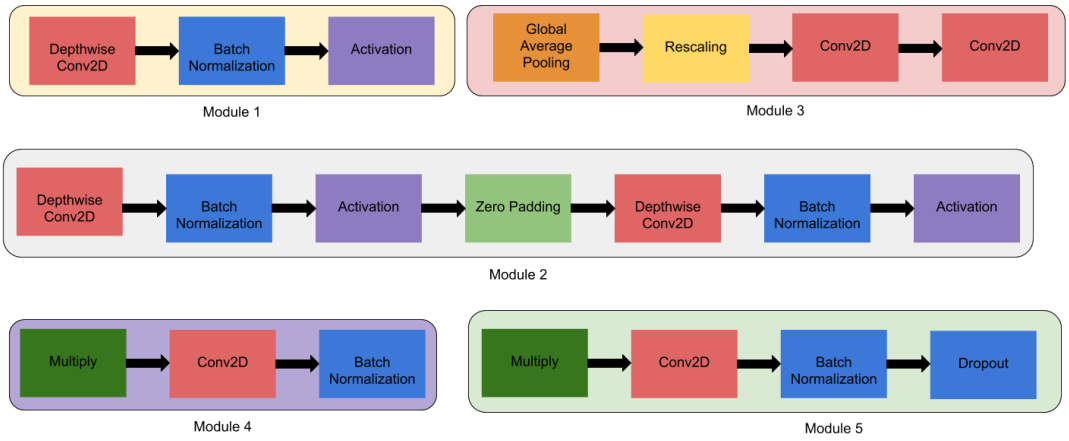


Figura 15: Moduli che formano i sottoblocchi

5.5.3 Motivazioni

Si è scelto di utilizzare tale modello per poter aprofondire un'architettura che ha raggiunto risultati allo stato dell'arte e considerando che nel paper [4] tale architettura

non era stata verificata. La scelta di EfficientNet B2 è stata presa dopo aver provato altre reti con minor numero di parametri ma anche superiori e tale scelta ha portato ad un trade-off tra tempi di training e numero di parametri, considerando come macchina utilizzata quelle messe a disposizione su Colab.

Approfondendo le soluzioni presenti nei Benchmark Top 1 Accuracy su ImageNet sono presenti modelli basati su Transformer o su EfficentNet perciò abbiamo voluto studiare tale architettura.

5.5.4 Creazione modello

Il modello base della rete viene creato partendo da EfficientNetB2 senza alcun caricamentro di pesi, si specifica il parametro *include-top=False* per escludere i livelli fully connected alla fine della rete. Vengono aggiunti in sequenza i seguenti livelli:

- **Global Average Pooling**: livello che riduce le dimensioni delle feature map mantenendo le informazioni più importanti applicando la media delle misure;
- Due livelli **Fully connected** con funzione di attivazione Relu e Kernel Regulizer;
- Due livelli di **dropout** 50%, che permettono di ridurre l'overfitting disattivando casualmente un certo numero di neuroni della rete.

Iperaparametri

Gli iperparametri scelti per l'addestramento sono:

- Batch size:32;
- Loss functions: Categorical crossentropy;
- Ottimizzatore: Adam;
- Epoche: 30;
- Learning rate: inizializzato a $1E^{-4}$ e ridotto tramite la funzione di utilità *ReduceLROnPlateau* con i seguenti parametri:
  - factor: fattore di riduzione, impostato a 0.5;
  - min_lr: valore minimo che può assumere il learning rate, impostato a $1E^{-6}$.

6 Risultati e prestazioni

6.0.1 Soluzione A

Il modello realizzato si è dimostrato molto buono sia dalla velocità di addestramento.

Nella figura possiamo vedere come il sistema, in fase di training, tenda ad aumentare la sua accuratezza fino a fermarsi a 89% su validation set.

Interessante la curva della accuracy sul training set. Tale forma è data dall'azione della funzione di utilità *ReduceLROnPlateau*, che riduce gradualmente il learning rate.

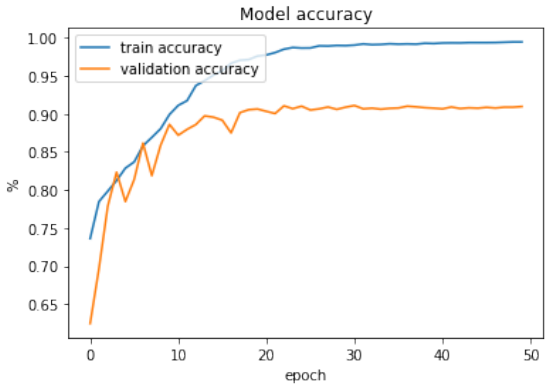
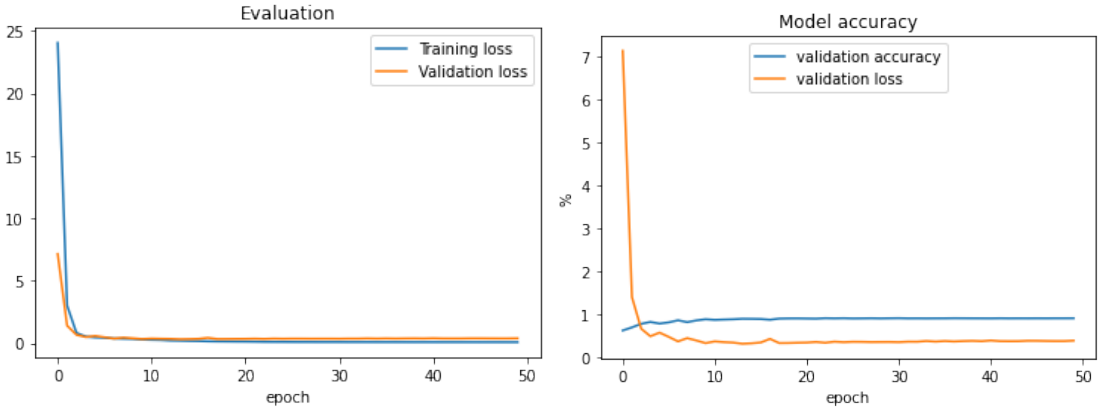


Figura 16: Andamento della accuratezza sul train e validation set durante l'addestramento



(a) (b)

Figura 17: Grafici su Accuracy e Loss sul validation set

Un'altra informazione che possiamo dedurre dal grafico, è che il sistema si adatta perfettamente alle immagini del training set, infatti l'accuracy arriva a toccare il 99%.

La differenza del 10% tra training set e validation set, mi fa pensare che la rete sia finita in una situazione di overfitting, dove le feature estratte applicate sul validation set non generalizzano al meglio il problema, nonostante l'ottimo risultato.

In figura 22a e 22b sono anche presentati i grafici rispettivamente tra il training loss e la validation loss nel primo e il grafico relativo alla validation accuracy e validation loss nel secondo.

Nella figura 18 è presentata la confusion matrix della soluzione. Come si nota dall'immagine, nel test set ci sono diverse cellule etichettate come sane, che vengono predette come malate. Infatti possiamo notare dai dati presenti nella tabella 2 che f1-score delle cellule sane è 84% mentre quello delle cellule malate è del 93%.

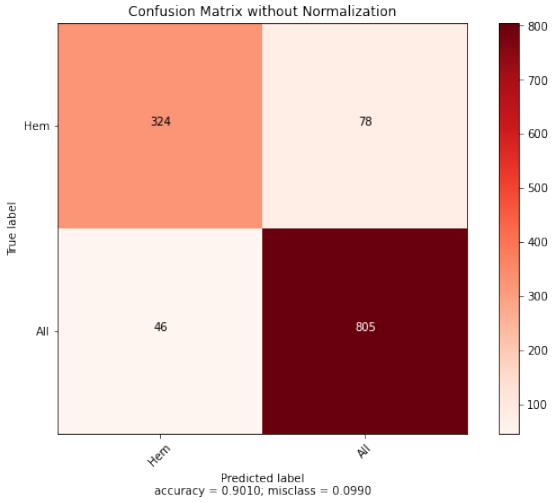


Figura 18: Confusion matrix della soluzione A

	precision	**recall**	**f1-score**	**support**
Hem	0.88	0.81	0.84	402
All	0.91	0.95	0.93	851
accuracy			0.90	1253
macro avg	0.89	0.88	0.88	1253
weighted avg	0.90	0.90	0.90	1253

Tabella 2: Statistiche sul test-set della soluzione A

6.0.2 Visualizzazione errori su test set

Nella figura 19 e 20 sono mostrate alcune immagini che il il modello erroneamente classifica nella classi sbagliate. Nella prima saranno mostrate le immagini etichettate come Hem

ma classificate come All (Falsi positivi), mentre nella seconda sono immagini etichettate come All ma classificate come Hem(Falsi Negativi).

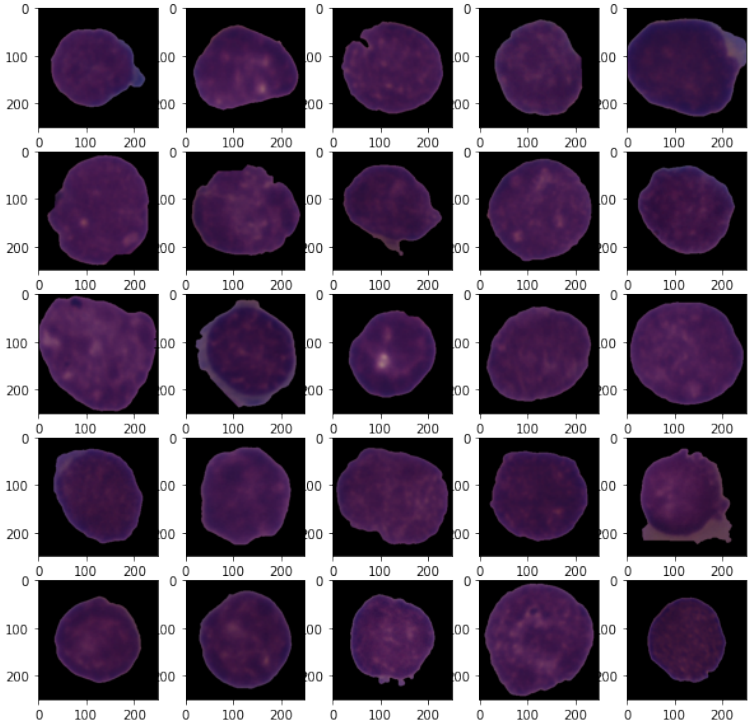


Figura 19: False positive su test set nella Soluzione A

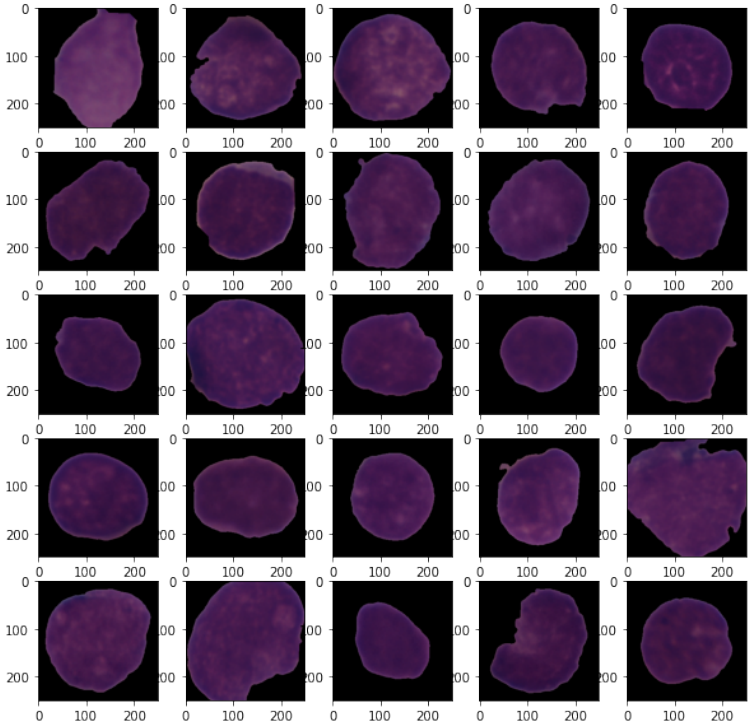
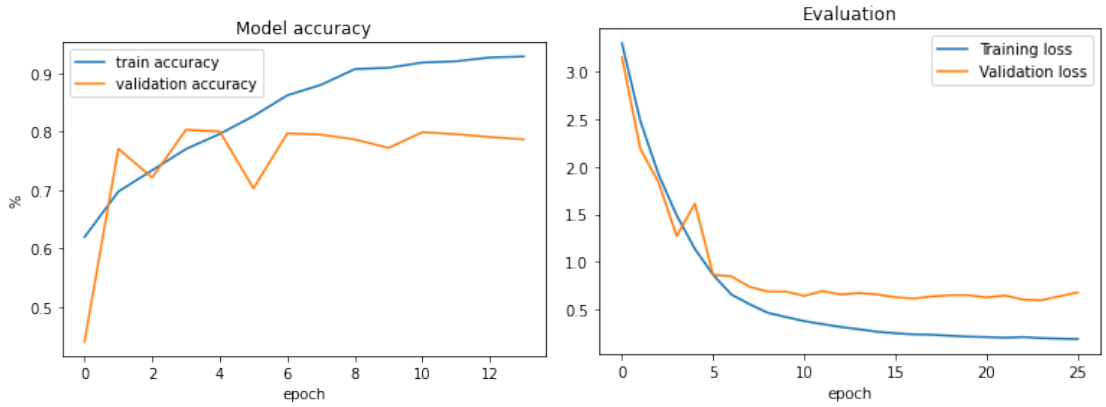


Figura 20: False negative su test set nella Soluzione A

6.0.3 Soluzione B

Prime 30 epoche - con earlystop

L'andamento delle accuratezze e della loss in funzione delle epoche sono graficati in Figura 21. Notiamo come l'accuratezza di training set e validation set aumentino gradualmente durante le epoche per assestarsi circa al 83% verso la fine dell'addestramento.

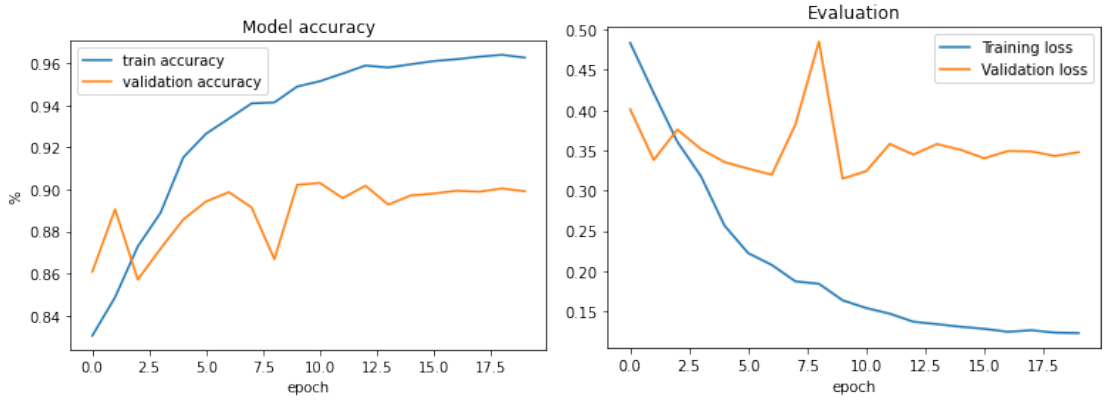


(a) (b)

Figura 21: Grafici

30 + 20 epoche - senza earlystop

Effettuando un addestramento senza early stopping è possibile migliore l'accuratezza sul validation set, in particolare l'andamento delle accuratezze e della loss in funzione delle epoche sono graficati in Figura 22. Notiamo come l'accuratezza di training set e validation set aumentino gradualmente durante le epoche per assestarsi circa al 90% verso la fine dell'addestramento.



(a) (b)

Figura 22: Grafici

La soluzione riportata consente di ottenere un'accuratezza sul Test del del 88% che supera alcune soluzioni ma non quella in letteratura.

In figura 23 è riportata la matrice di confusione, ricordando che il test set è sbilanciato ossia presenta molte più immagini di pazienti malati rispetto a quelli sani, il classificatore rispetto ad altre soluzioni si comporta bene per entrambe le classi.

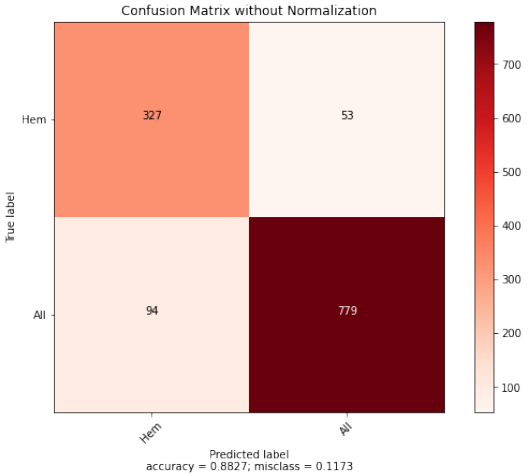


Figura 23: Matrice di confusione

In tabella 3 si nota come l'F1-Score della classe Hem(sane) sia leggermente inferiore a quello della classe All(Malate). Tale problema era già emerso dallo studio delle soluzioni esistenti su GitHub e Kaggle ma con tale soluzione la differenza non è significativa come in altre.

	precision	**recall**	**f1-score**	**support**
Hem	0.78	0.86	0.82	380
All	0.94	0.89	0.91	873
accuracy			0.88	1253
macro avg	0.86	0.88	0.87	1253
weighted avg	0.89	0.88	0.88	1253

Tabella 3: Statistiche sul test set Soluzione B

6.0.4 Visualizzazione errori su test set

Di seguito in figura 24 si mostrano gli errori commessi sul test set per i False positive cioè le immagini classificate come malate ma che in realtà sono sane. Mentre in figura 25 si mostrano i Falsi negativi cioè le immagini classificate come sane mentre in realtà sono malate. Come si può notare la differenza tra le due non è ben visibile ad occhio nudo, perciò è comprensibile che su tali immagini il classificatore commetta più errori.

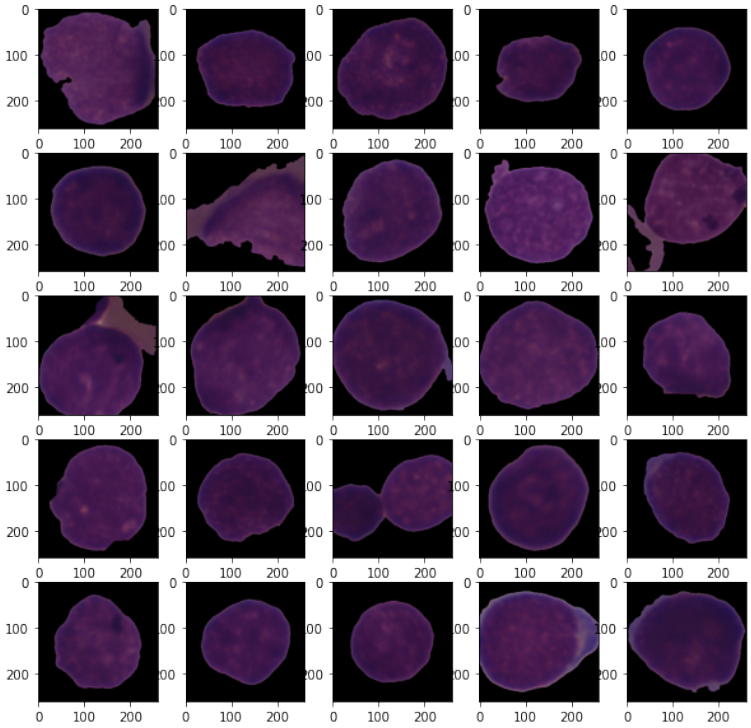


Figura 24: False positive su test set

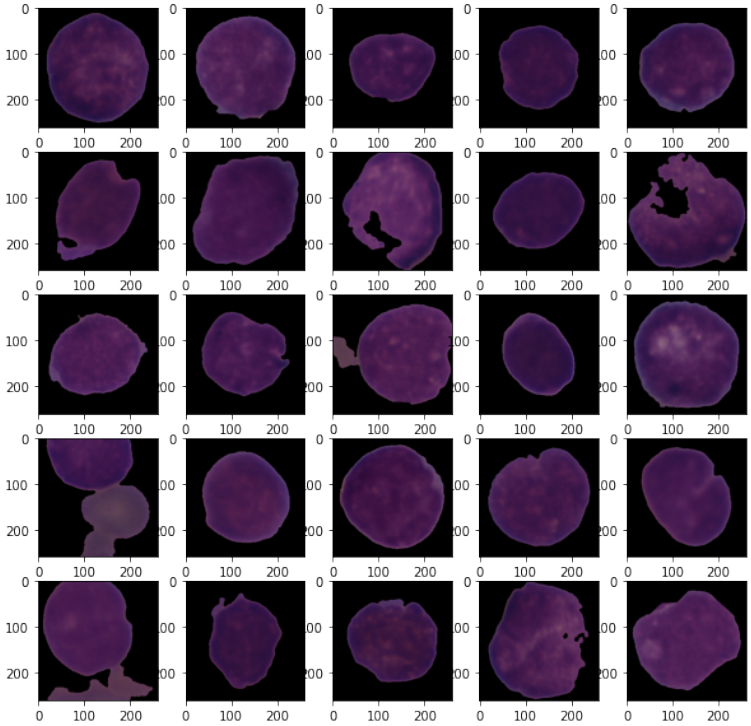


Figura 25: False negative su test set

7 Conclusioni e sviluppi futuri

Le soluzioni proposte ci hanno permesso di classificare con un considerevole livello di accuratezza, le cellule malate da quelle sane riferite alla Leucemia Linfoblastica Acuta.

I risultati ottenuti sono ancora lontani dallo stato dell'arte per questo dominio applicativo.

La realizzazione di queste soluzioni ci hanno permesso di studiare nuovi architetture e tecniche utili per progettare e implementare una rete neurale che permetta di risolvere un problema in un contesto reale.

7.1 Sviluppi futuri

I possibili sviluppi futuri per la soluzione A possono essere:

- studiare il comportamento di una rete più profonda aggiungendo una serie di moduli tra quelli definiti in 5.4.1;
- analizzare le immagini erroneamente classificate ed realizzare un classificatore in grado di discriminarle ed unirlo a quello già realizzato;
- introdurre nuove tecniche di pre-processing al fine di estrarre feature più robuste.

Gli sviluppi futuri per la soluzione B sono:

- reperimento di maggiori risorse computazionali al fine di valutare anche modelli di EfficentNet con più parametri (ad esempio fino a EfficentNetB7);
- unione di diversi modelli tramite tecniche di *Ensamble Learning* integrando quelli realizzati e/o sviluppandone di nuovi che vadano a coprire specifiche feature.

Riferimenti bibliografici

[1] Ahmed Abdeldaim et al. ≪Computer-Aided Acute Lymphoblastic Leukemia Diagnosis System Based on Image Analysis≫. In: gen. 2018, pp. 131–147. ISBN: 978-3-319-63753-2. DOI: 10.1007/978-3-319-63754-9_7.

[2] Humanitas. *Leucemia linfoblastica acuta*. https://www.humanitas.it/malattie/leucemia-linfoblastica-acuta/.

[3] Larxel. *Leukemia Classification*. https://www.kaggle.com/andrewmvd/leukemia-classification. 2020.

[4] Jose Elwyslan Maurıcio de Oliveira e Daniel Oliveira Dantas. ≪Classification of Normal versus Leukemic Cells with Data Augmentation and Convolutional Neural Networks≫. In: (2019). URL: https://www.scitepress.org/Papers/2021/102574/102574.pdf.

[5] Amjad Rehman et al. ≪Classification of acute lymphoblastic leukemia using deep learning≫. In: *Microscopy Research and Technique* 81.11 (2018), pp. 1310–1317.

[6] Associazione Italiana per la Ricerca sul Cancro. *Leucemia linfoblastica acuta*. https://www.airc.it/cancro/informazioni-tumori/guida-ai-tumori/leucemia-linfoblastica-acuta.

[7] Mingxing Tan e Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019. arXiv: 1905.11946 [cs.LG].