UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA BIOMEDICA

"DATA AUGMENTATION PER LA CLASSIFICAZIONE DI CROMOSOMI"

Relatore: Prof. Loris Nanni

Laureando: Guglielmo Nicolas

ANNO ACCADEMICO 2022 – 2023

Data di laurea 16/03/2023

Abstract

La classificazione dei cromosomi è una sfida importante in molti campi, tra cui la medicina e la biologia. Per ottenere risultati accurati è fondamentale avere a disposizione una grande quantità di dati di addestramento la cui raccolta e annotazione possono essere costose e possono richiedere molto tempo.
In questa tesi, è stato affrontato il problema della carenza di dati di addestramento utilizzando tecniche di data augmentation per migliorare la classificazione di cromosomi utilizzando le reti neurali convoluzionali AlexNet e ResNet50. È stato utilizzato un dataset di immagini di cromosomi umani e sono state applicate diverse tecniche di data augmentation, tra cui rotazione, traslazione, straightening, FT e image enhancement.
Le reti, AlexNet e ResNet50, sono state addestrate sia con dati originali sia con dati di addestramento aumentati e sono state confrontate le diverse prestazioni ottenute. In sintesi, i risultati della ricerca suggeriscono che l'uso di tecniche di data augmentation possono essere un'ottima soluzione per migliorare l'accuratezza della classificazione di cromosomi, rispetto all'uso di dati originali, riducendo la necessità di raccogliere grandi quantità di dati di addestramento.

Introduzione

La classificazione di cromosomi è un'importante attività di analisi delle immagini utilizzata in molte applicazioni mediche e di ricerca biologica. L'accurata identificazione e classificazione dei cromosomi può aiutare a diagnosticare malattie genetiche e a comprendere meglio la struttura del genoma umano. Tuttavia, la raccolta e l'annotazione di grandi quantità di dati di addestramento può essere costosa e richiedere molto tempo.
L'apprendimento automatico, in particolare le reti neurali convoluzionali, rappresenta una soluzione promettente per migliorare l'accuratezza della classificazione di cromosomi. Però, l'utilizzo di queste tecniche richiede la disponibilità di un dataset molto ampio.
La tesi si concentra sulla risoluzione di questo problema attraverso l'uso di varie tecniche di data augmentation per migliorare la classificazione utilizzando le architetture AlexNet e ResNet50.
La data augmentation è una tecnica di elaborazione delle immagini che consente di aumentare il dataset di addestramento attraverso la manipolazione dei dati esistenti. Questa tecnica può aiutare a ridurre la necessità di raccogliere grandi quantità di dati di addestramento annotati e migliorare l'accuratezza della classificazione.
L'obiettivo di questa tesi è valutare l'efficacia della data augmentation nell'aumentare le prestazioni della classificazione di cromosomi, confrontando i risultati ottenuti dalla rete. I risultati ottenuti possono fornire importanti informazioni sulle strategie per migliorare la classificazione di cromosomi e possono essere utili per lo sviluppo di sistemi di diagnostica medica più precisi ed efficienti.
Nel primo capitolo verranno descritti i cromosomi prestando attenzione al loro ruolo e le loro proprietà, all'importanza del loro studio e l'ottenimento delle immagini; successivamente, nel secondo capitolo, verranno introdotte le reti usate e la loro struttura. Nel terzo capitolo verranno spiegate le diverse tecniche di data augmentation, dalle trasformazioni più basilari come rotazione e traslazione alle più complesse come, Fourier Transform e la tecnica di image enhancement sotto esame. Nel quarto e nel quinto capitolo verranno trattate le diverse combinazioni di trasformazioni applicate per aumentare il dataset. Dallo studio dei risultati ottenuti durante i test si potrà notare quale combinazione porterà ad una accuratezza maggiore. Infine, nell'ultimo capitolo verranno riportate le considerazioni ottenute dai risultati.

Indice

Abstract

Introduzione

1 I cromosomi

1.1 Introduzione ai cromosomi
1.2 Ruolo e proprietà dei cromosomi
1.3 Perché studiarli

2 Reti neurali convoluzionali

2.1 Introduzione alle CNN
2.2 AlexNet
2.3 ResNet50
2.4 Fine-Tuning

3 Tecniche di data augmentation

3.1 Chromosome Data Augmentation – CDA
3.2 Straightening
3.3 Trasformazioni di base
3.4 Trasformata di Fourier
3.5 Sharpening
3.6 Image Enhancement

4 Fase sperimentale

4.1 Aspetti generali
4.2 Dataset

5 Risultati sperimentali

6 Conclusioni

Riferimenti bibliografici

Capitolo 1

I cromosomi

1.1 Introduzione ai cromosomi

I cromosomi sono strutture cellulari che contengono il materiale genetico di un organismo. Ogni specie ha un numero specifico di cromosomi che si trovano nel nucleo delle sue cellule. I cromosomi sono costituiti da lunghe molecole di DNA, che sono avvolte intorno a proteine chiamate istoni per formare una struttura a forma di bastoncino.
Durante la divisione cellulare, i cromosomi si duplicano e si separano, assicurando che ogni cellula figlia abbia lo stesso set di cromosomi della cellula madre. Ciò è essenziale per la riproduzione e la crescita dell'organismo.
Le informazioni genetiche contenute nei cromosomi sono fondamentali per la determinazione delle caratteristiche fisiche e comportamentali di un organismo. Studiare i cromosomi e il loro contenuto genetico può aiutare a comprendere le cause di alcune malattie genetiche e sviluppare nuove terapie per trattarle.

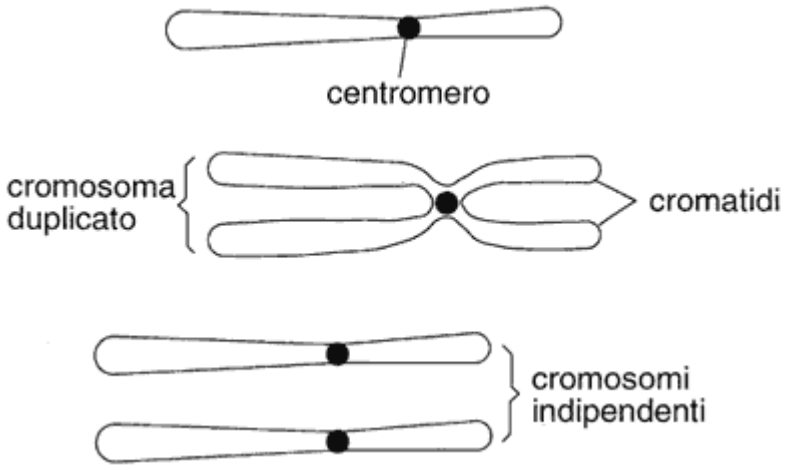
1.2 Ruolo e proprietà dei cromosomi

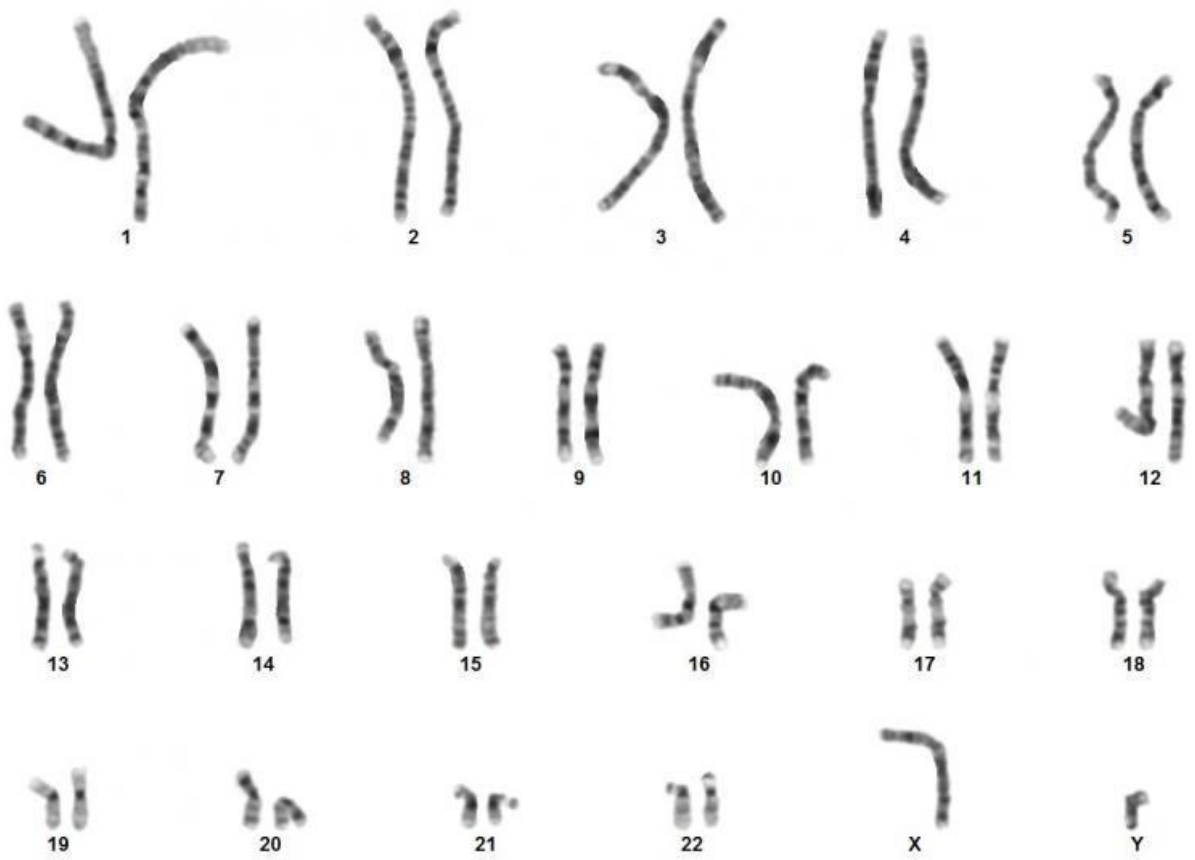
I cromosomi, strutture complesse che si trovano all'interno delle cellule, svolgono diversi ruoli cruciali per il corretto funzionamento dell'organismo nel suo insieme. Uno dei ruoli principali dei cromosomi è quello di trasmettere i geni, le unità fondamentali dell'ereditarietà, alle cellule figlie durante la divisione cellulare. In questo modo, le informazioni genetiche vengono trasmesse di generazione in generazione ma i cromosomi non si limitano semplicemente a trasmetterle, infatti svolgono anche un ruolo attivo nella regolazione dell'espressione dei geni, controllando quando e come i geni vengono attivati o disattivati. Inoltre, i cromosomi proteggono il DNA da possibili danni causati da fattori ambientali come la radiazione e le sostanze chimiche, avvolgendolo intorno alle proteine istoniche.
La diversità genetica tra gli individui di una stessa specie è il risultato di variazioni nel numero o nella struttura dei cromosomi. Le mutazioni cromosomiche possono causare la duplicazione o la perdita di parti dei cromosomi, portando a variazioni nel fenotipo dell'individuo. Questa variazione genetica può anche portare alla formazione di nuove specie o alla diversificazione delle popolazioni esistenti, contribuendo così alla diversità biologica dell'ecosistema.
In sintesi, i cromosomi svolgono un ruolo fondamentale nella trasmissione delle informazioni genetiche, nella regolazione dell'espressione dei geni, nella protezione del DNA e nell'evoluzione delle specie.

I cromosomi possono anche causare problematiche dal momento che mutazioni genetiche, ovvero cambiamenti nel DNA dei cromosomi, possono portare a malattie genetiche come la Sindrome di Down, la fibrosi cistica e il cancro.

1.3 L'importanza dello studio dei cromosomi

La classificazione manuale dei cromosomi è un processo lungo e costoso che richiede competenze specialistiche. Pertanto, l'utilizzo di tecniche di apprendimento automatico, come le reti neurali convoluzionali, per la classificazione di cromosomi può rappresentare una soluzione promettente per migliorare l'efficienza e l'accuratezza di questa attività.
La classificazione dei cromosomi è un'attività importante che offre molteplici benefici in diversi ambiti. In primo luogo, è essenziale per identificare eventuali anomalie cromosomiche che possono causare malattie genetiche. Grazie alla conoscenza delle normali caratteristiche dei cromosomi, gli specialisti possono individuare eventuali difetti che possono causare ad esempio la sindrome di Down, la sindrome di Turner o la sindrome di Klinefelter.
Inoltre, la classificazione dei cromosomi è fondamentale per la ricerca scientifica in biologia e medicina. Questa attività permette di studiare le relazioni tra i geni e le malattie, fornendo importanti informazioni sulle caratteristiche genetiche degli organismi che sono poi utilizzate per sviluppare nuove terapie e trattamenti per le malattie genetiche.
La classificazione dei cromosomi può anche aiutare nella diagnosi di malattie genetiche, in quanto permette di identificare eventuali anomalie cromosomiche correlate alla malattia. Questa attività è particolarmente importante per malattie rare, che possono essere difficili da diagnosticare.
Inoltre, la classificazione dei cromosomi può influenzare la scelta del trattamento per alcune malattie genetiche, come ad esempio, la presenza di una particolare anomalia cromosomica può influire sulla scelta della terapia per il cancro.





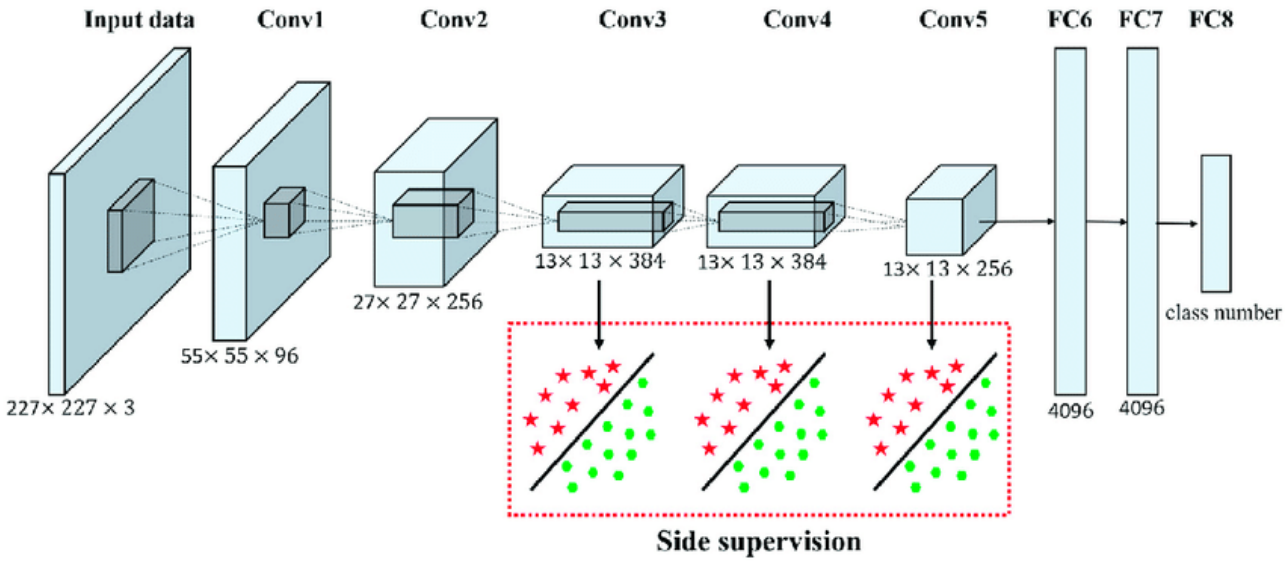
RESULT: 46,XY

Capitolo 2

Reti neurali convoluzionali

2.1 Introduzione alle CNN

Le Convolutional Neural Networks (CNN), o Reti Neurali Convoluzionali, sono un tipo di algoritmo di deep learning utilizzato soprattutto nel campo della computer vision e del riconoscimento di immagini.
Le CNN sono state sviluppate specificamente per lavorare con dati bidimensionali come le immagini e sono in grado di riconoscere automaticamente i modelli all'interno di esse. Utilizzano una combinazione di strati convoluzionali, strati di pooling e strati completamente connessi per estrarre e analizzare le caratteristiche delle immagini, costruendo un modello di classificazione degli oggetti.
Le CNN hanno dimostrato di avere una grande efficacia in molti compiti di computer vision, come il riconoscimento facciale, il rilevamento di oggetti, la classificazione di immagini.



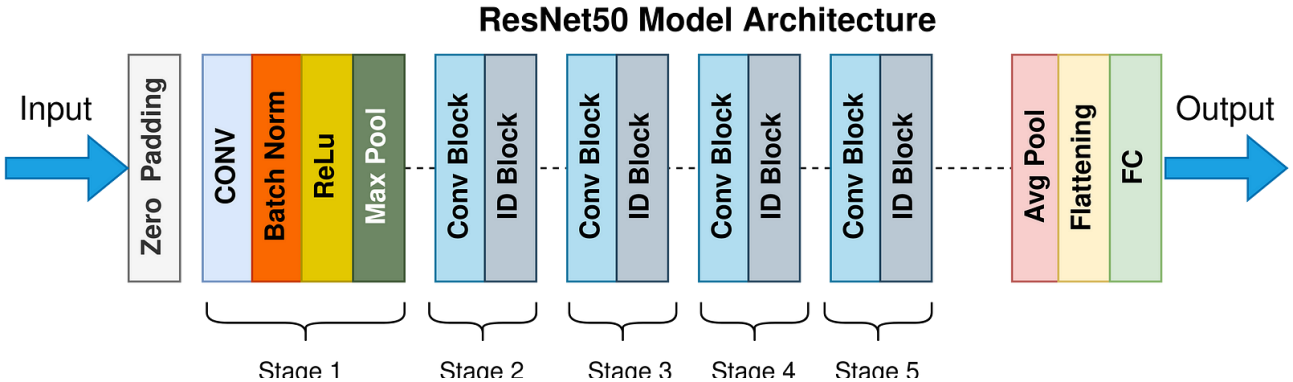
2.2 AlexNet

AlexNet [2] è una rete neurale convoluzionale (CNN) sviluppata nel 2012 da Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton. È stata la prima CNN a vincere il concorso ImageNet Large Scale Visual Recognition Challenge (ILSVRC) nel 2012, segnando un importante passo avanti nell'ambito della visione artificiale.
La caratteristica chiave di questo modello è la profondità che da un lato rende la rete molto pesante, dall'altro la rende molto performante e grazie all'utilizzo di GPU per il training è possibile allenarla. La rete AlexNet è composta da 5 strati convoluzionali e 3 strati fully connected, per un totale di 8 strati. Inoltre, utilizza tecniche come la

regolarizzazione dropout, per disattivare casualmente un certo numero di neuroni durante la fase di addestramento, e la batch normalization per evitare l'overfitting, migliorare la performance della rete e ridurre il tempo di addestramento.
Durante l'addestramento, AlexNet utilizza la cross-entropy come funzione di costo, metrica usata per misurare quanto è performante un modello di classificazione, e la Stochastic Gradient Descent (SGD) come algoritmo di ottimizzazione. La rete viene addestrata su un dataset di immagini etichettate, in cui la rete impara ad associare le caratteristiche delle immagini alle rispettive etichette di classe.
In sintesi, AlexNet è una rete neurale convoluzionale complessa ma altamente efficace per la classificazione di immagini, in grado di apprendere in modo autonomo e di riconoscere le caratteristiche delle immagini con una precisione elevata.

2.3 ResNet50

ResNet50 [3] è un'architettura di rete neurale convoluzionale (CNN) introdotta nel 2015 da Microsoft Research. Fa parte della famiglia di reti neurali ResNet, che sono note per l'utilizzo di connessioni residue che aiutano a risolvere il problema dell'annullamento del gradiente, che può verificarsi durante l'allenamento di reti neurali molto profonde.
ResNet50 è composto da 50 strati, tra cui strati convoluzionali, strati di batch normalization e funzione di attivazione ReLU.
L'input alla rete ResNet50 è un'immagine RGB di dimensioni 224x224. La rete restituisce un vettore di probabilità che rappresenta la probabilità di ciascuna delle 1000 possibili classi nel dataset ImageNet.
ResNet50 ha raggiunto risultati all'avanguardia in una varietà di compiti di visione artificiale, inclusa la classificazione delle immagini, la rilevazione degli oggetti e la segmentazione delle immagini.



2.4 Fine-Tuning

Il fine-tuning è un metodo di transfer learning utilizzato per migliorare l'addestramento di una rete neurale quando la quantità di dati a disposizione è limitata. In questo approccio la rete neurale viene pre-addestrata utilizzando ImageNet, un dataset che comprende oltre un milione di immagini appartenenti a

mille categorie diverse. Per applicare il fine-tuning, si rimuovono gli ultimi tre livelli della rete, che sono responsabili della classificazione dell'immagine, e vengono sostituiti con nuovi livelli completamente connessi. Questi livelli includono un livello SoftMax finale che contiene un numero di neuroni pari al numero di classi presenti nel dataset di immagini che si vuole identificare.

Capitolo 3

Tecniche di data augmentation

3.1 Chromosome Data Augmentation (CDA)

La tecnica basata su CDA [4] consiste nella rotazione casuale delle immagini e della divisione del dataset iniziale in tre subset: train set, validation set e test set, regolando questa divisione attraverso due parametri di offset.
Oltre al dataset delle immagini X, viene fornito il vettore Y delle etichette corrispondenti alle immagini.
Inizialmente il dataset in input e le etichette corrispondenti vengono divise tra test set e un set temporaneo, con i rispettivi label, in base al primo valore di offset; successivamente le immagini nel set temporaneo vengono ruotate casualmente e, in base al secondo offset, assegnate al train set o validation set con le rispettive etichette delle immagini iniziali.
Oltre alle immagini ruotate, nei vari set sono state aggiunte le immagini originali, sempre divise in base agli offset, così da creare dei set più grandi.

3.2 Straightening

Lo straightening è un'operazione di preprocessing che è stata usata in diversi studi durante la classificazione dei cromosomi che, quasi sempre, sono piegati.
Esistono diversi approcci per raddrizzare le immagini dei cromosomi, quello usato in questa tesi è basato sulla funzione straighten [5], la quale lavora con le spline e l'interpolazione che, dato un insieme di punti, cerca di trovare una funzione che passi per la maggior parte dei punti.
Per applicare questa funzione è necessario avere due vettori, x e y, che indicano le coordinate della linea centrale, o scheletro, del cromosoma. Per ottenere questi vettori è stata usata la funzione built-in di Matlab bwskel() che però non forniva i punti nell'ordine richiesto e tagliava coda e testa dei cromosomi.
Per ovviare a questi problemi è stato applicato un algoritmo per riordinare i punti e un altro per crearne in aggiunta. Prima di eseguire la funzione di straighten viene controllata la vicinanza delle coordinate ai bordi dell'immagine che, in caso, porterebbero a errori durante la fase di creazione di pose aggiuntive.
Un ulteriore problema è dato sempre dalla funzione bwskel() che per alcuni cromosomi forniva coordinate per le quali il riordinamento richiedeva un'analisi dei punti più complessa.

3.3 Trasformazioni di base

Le trasformazioni in questione sono le più basilari, ovvero: riflessione, scala e traslazione.
Più precisamente sono state applicate la riflessione sia lungo l'asse x, sia lungo l'asse y; per la trasformazione di scala è possibile la modifica dell'immagine originale lungo entrambi gli assi contemporaneamente e lungo gli assi separati, inserendo, nelle giuste opzioni di modifica, opportuni range, in questo caso da 0.7 a 1.3, per evitare di modificare troppo le informazioni sulle immagini. Come per il rescale, la traslazione viene effettuata lungo ogni asse, usando un range tra -30 e 30 pixel è possibile creare pose aggiuntive con un basso rischio di perdita di informazioni.
Questa funzione richiede in ingresso il dataset delle immagini con relative etichette e 6 parametri, con valore tra 0 e 1, che rappresentano la probabilità di attivare le rispettive trasformazioni: rx e ry controllano la funzione specchio, rx esegue il flip sottosopra dell'immagine e ry da destra a sinistra; sx e sy controllano il rescale dell'immagine lungo i due assi separatamente; tx e ty si occupano invece della traslazione, sempre separatamente.
Queste trasformazioni vengono eseguite in cascata e, nel caso una trasformazione avesse tutti i parametri nulli, l'apposita funzione, anche se richiamata, restituisce l'immagine originale.

3.4 Trasformata di Fourier

Oltre alle trasformazioni di base, traslazione, scala, rotazione e riflessione, la trasformata di Fourier [6] permette di creare pose aggiuntive aggiungendo rumore casuale o lavorando sullo spettro dell'immagine e sul dominio delle frequenze.
Questa funzione, oltre al dataset e le etichette, richiede la selezione di tre parametri: mask1, mask2, mask3, i quali vengono usati come controlli e servono ad attivare le diverse trasformazioni.
Nella prima trasformazione all'immagine viene applicata la trasformata di Fourier e centrate le basse frequenze. Un kernel viene inizializzato come un rettangolo della dimensione dell'immagine trasformata composto da 1 e, successivamente, viene calcolato il raggio minimo per il quale i valori della maschera all'interno dell'area delimitata da questo raggio andranno posti a zero.
Creato il kernel, viene eseguito il prodotto di ogni elemento tra l'immagine e la maschera. Infine, la funzione restituisce il modulo dell'inversa del prodotto precedente.
La seconda maschera è regolata dal parametro 'p', compreso tra 0 e 1, che serve a scegliere la percentuale di pixel da rimuovere dalla trasformata dell'immagine. Per fare questo viene eseguita la FT dell'immagine; dopo aver creato un vettore con elementi da 1 al numero di pixel dell'immagine viene creato un vettore contenente gli indici dei pixel 'indici_random', creato randomizzando il vettore precedente. Per

scegliere gli indici casualmente viene creato un ultimo vettore di indici random a cui vengono assegnati gli indici in funzione degli elementi casuali scelti.
Nell'ultima trasformazione, all'immagine in ingresso, viene applicata la DCT, successivamente vengono rimosse alcune componenti alle basse frequenze azzerandole e viene restituita la nuova immagine subito dopo aver eseguito l'inversa della DCT.

3.5 Sharpening

Per questa tecnica è stata usata la funzione deconvblind(), che deconvolve l'immagine, schiarendola usando il Point-Spread Function (PSF), funzione di diffusione del punto, ovvero un filtro che viene applicato all'immagine originale.
In questa tesi sono stati creati tre tipi di filtri, i primi due con il metodo della deconvoluzione: gaussiano, con dimensione 7x7 e deviazione standard uguale a 2; average, sempre con dimensione 7x7. L'ultimo filtro, come il primo, è un filtro gaussiano creato con la funzione built-in di matlab imsharpen() controllata da tre parametri: 'Radius', è la deviazione standard del filtro passa-basso gaussiano che verrà applicato, incrementa i valori dei pixel del cromosoma; 'Amout' per impostare la durezza dell'effetto smussamento, regola il contrasto dell'immagine; l'ultimo parametro, 'Thresold', serve a decidere il contrasto minimo richiesto per un pixel per essere considerato un pixel di bordo, aiuta a regolare l'immagine.
Questo tipo di elaborazione può essere usata efficientemente quando non ci sono informazioni riguardanti il tipo di distorsione, sfocamento o rumore.

3.6 Image Enhancement

Questa tecnica, usata in [1], serve a migliorare il contrasto e i contorni dell'immagine. Per ottenere questi risultati è stata usata inizialmente la convoluzione con un kernel 5x5 scalato di 1/16, successivamente, le intensità dei pixel della nuova immagine sono stati elevati a una potenza opportuna, creando il secondo tipo di immagine e infine è stata eseguita nuovamente una convoluzione tra la seconda immagine e il kernel scalato di 1/8.
In questa trasformazione l'immagine in ingresso viene trasformata da RGB a scala di grigi, successivamente viene eseguito uno stretch del contrasto moltiplicando ogni valore dell'immagine a scala di grigi per l'intensità massima tra tutti i pixel, 'M', diminuita di 30 e divisa per 'M'.
Dopo questa operazione vengono create le nuove immagini e, dopo un rescale per mantenere le dimensioni di partenza, vengono presi i coefficienti a parte reale.
Prima di aggiungere le immagini al dataset viene eseguito un controllo sui parametri che controllano questa funzione: p1,p2 e p3. Questi parametri, compresi tra 0 e 1, sono le probabilità con cui verranno assegnate le varie trasformazioni.

All’assegnazione dell’immagine viene assegnato anche il rispettivo label. Infine, dopo aver aggiunto il dataset iniziale, viene eseguito uno shuffle del dataset prima di restituirlo.

Successivamente seguiranno gli pseudocodici delle funzioni

PSEUDOCODICE CDA

```
REQUISITI:
X, dataset immagini
Y, etichette corrispondenti a X
r_test, test ratio
r_valid, valid ratio

FUNCTION CDA(X,Y,r_test,r_valid)
trainSet <- {}
validSet<- {}
testSet <- {}
temporarySet <- {}
// prima divisione immagini originali tra testSet e temporarySet
FOR (x,y) in (X,Y) DO
        IF random(0,1) < test_ratio
                testSet <- testSet U {(x,y)}
        ELSE
                temporarySet <- temporarySet U {(x,y)}
        END
END
% divisione immagini originali nel temporarySet tra validSet e trainSet
FOR (x,y) in temporarySet DO
        IF random(0,1) < valid_ratio
                validSet <- validSet U {(x,y)}
        ELSE
                trainSet <- trainSet U {(x,y)}
        END
END
// rotazione immagini e assegnazione a validSet o trainSet
FOR (x,y) in temporarySet DO
        θ <- 2π*rand(0,1)
        x(θ) <- A(θ)*x
        IF random(0,1) < valid_ratio
                validSet <- validSet U {(x(θ),y)}
        ELSE
                trainSet <- trainSet U {(x(θ),y)}
        END
END
// mescola set
trainSet,validSet,testSet <- shuffle(trainSet,validSet,testSet)
RETURN trainSet,validSet,testSet
END_funzione
```

PSEUDOCODICE TRASFORMAZIONI BASE

```
REQUISITI:
X, dataset immagini
Y, etichette corrispondenti a X
r=[rx ry], probabilità specchio
s=[sx sy], probabilità scala
t=[tx ty], probabilità traslazione

FUNCTION BASE(X,Y,r,s,t)
newSet <- {}
// basta che un elemento di r,s o t sia maggiore di zero
IF (r OR s OR t) != ∅
        FOR (x,y) in (X,Y) DO
                // applico trasformazioni in cascata
                newImage <- MIRROR(x,rx,ry)
                newImage <- SCALA(newImage,sx,sy)
                newImage <- TRASLA(newImage,tx,ty)
                newSet <- newSet U {(newImage,y)}
        END
END
// aggiunta immagini originali
newSet <- newSet U {(X,Y)}
// mescola set
newSet <- shuffle(newSet)
RETURN newSet
END_function
```

PSEUDOCODICE SOTTOFUNZIONI TRASFORMAZIONI BASE

```
MIRROR(x,rx,ry)
rnd <- rand(0,1)
IF rnd < rx
        x <- flipUD(x) // flip upside down
ELSEIF rnd < ry
        x <- flipLR(x) // flip left right
END
RETURN x
END_function


SCALA(x,[rows_x cols_x depth_x],sx,sy)
xNew <- x
(a,b) <- (0.7,1.3)
rnd <- rand(0,1)
IF rnd < sx
        scx <- a+(b-a)*rand(0,1)
        x <- imresize(x,[scx*rows_x cols_x])
        new_rows <- rows_x
ELSEIF rnd < sy
        scy <- a+(b-a)*rand(0,1)
        x <- imresize(x,[rows_x scy*cols_x])
END
(dx,dy) <- (|(cols_xNew - cols_x)/2|,|(rows_xNew - rows_x)/2|)
IF rows_x >= rows_xNew
        IF cols_x >= cols_xNew
                xNew <- x(dx+1:dx+cols_xNew,dy+1:dy+rows_xNew,:)
        ELSE
                xNew(:,dy+1:dy+rows_x,:) <- x(dx+1:dx+cols_xNew,:,:)
        END
ELSE
        IF cols_x >= cols_xNew
                xNew(dx+1:dx+cols_x,:,:) <- x(:,dy+1:dy+rows_xNew,:)
        ELSE
                xNew(dx+1:dx+cols_x,dy+1:dy+rows_x,:) <- x(:,:,:)
        END
RETURN xNew
END_function


TRASLA(x,tx,ty)
(c,d) <- (-30,30) // range traslazioni
rnd <- rand(0,1)
IF rnd < rx
        traslx <- c+(d-c)*rand(0,1)
        x <- imtranslate(x,[traslx 0])
ELSEIF rnd < ry
        trasly <- c+(d-c)*rand(0,1)
        x <- imtranslate(x,[0 trasly])
END
RETURN x
END_function
```

PSEUDOCODICE TRASFORMATE DI FOURIER

```
REQUISITI:

X, dataset immagini
Y, etichette corrispondenti a X
mask1, attivazione maschera 1
mask2, attivazione maschera 2
mask3, attivazione maschera 3

FUNCTION FOURIER(X,Y,mask1,mask2,mask3)
step <- mask1 + mask2 + mask3
newSet <- {}
FOR i = 1:step:length(X)
        image <- X(i)
        w <- i
        IF mask1 == 1
                IF w <= length(X)
                        imgM1 <- M1(image)
                        newSet <- newSet U {(imgM1,y(w))}
                        w <- w + 1
                END
        END
        IF mask2 == 1
                IF w <= length(X)
                        imgM2 <- M2(image)
                        newSet <- newSet U {(imgM2,y(w))}
                        w <- w + 1
                END
        END
        IF mask3 == 1
                IF w <= length(X)
                        imgM3 <- M3(image)
                        newSet <- newSet U {(imgM3,y(w))}
                END
        END
END
newSet <- newSet U {(X,Y)}
newSet <- shuffle(newSet)
RETURN newSet
END_function
```

PSEUDOCODICE SOTTOFUNZIONI TRASFORMATE DI FOURIER

```
M1(image)
image <- rgb2gray(image)
image <- FT(image)
// sposto al centro le basse frequenze
image <- FTshift(image)
// inizializzo maschera quadrata
maschera <- ones(rows_image,cols_image)
// creo due matrici di indici
(x,y) <- meshgrid(1:cols_image,1:rows_image)
// coordinate del centro
(cx,cy) <- (cols_image/2,rows_image/2)
// distanza euclidea punti nelle griglie dal centro
R <- ||(x-cx) + (y-cy)||
// scelgro raggio r nel quale gli elementi della maschera si annullano
r <- 2
maschera(R<=r) <- 0
// applico la maschera
image <- image.*maschera
image <- |iFT(iFTshift(image))|
image <- gray2rgb(image)
RETURN image
END_function

// pongo a zero il p% dei dei pixel della trasformata di Fourier
M2(image)
p <- 0.5
image <- rgb2gray(image)
image <- FT(image) // trasformata di fourier
elementi <- numel(image)
indici_random <- randperm(elementi)
elementi_scelti <- p*elementi
indici_random_scelti <- indici_random(1:elementi_scelti)
// azzero
image(indici_random_schelti) <- 0
image <- real(iFT(image))
image <- gray2rgb(image)
RETURN image
END_function

// DCT
M3(image)
image <- rgb2gray(image)
image <- DCT(image)
// elimino componenti alle basse frequenze
image(1:10,1:10) <- 0
image <- iDCT(image)
RETURN image
END_function
```

PSEUDOCODICE SHARPEN

```
REQUISITI:

X, dataset immagini
Y, etichette corrispondenti a X
gauss1, attivazione maschera 1
avg, attivazione maschera 2
gauss2, attivazione maschera 3

FUNCTION FOURIER(X,Y,gauss1,avg,gauss2)
step <- gauss1 + avg + gauss2
newSet <- {}
FOR i = 1:step:length(X)
        image <- X(i)
        w <- i
        IF gauss1 == 1
                IF w <= length(X)
                        imgG1 <- G1(image,[7 7],2)
                        newSet <- newSet U {(imgG1,y(w))}
                        w <- w + 1
                END
        END
        IF avg == 1
                IF w <= length(X)
                        imgAVG <- AVERG(image,[7 7])
                        newSet <- newSet U {(imgAVG,y(w))}
                        w <- w + 1
                END
        END
        IF gauss2 == 1
                IF w <= length(X)
                        imgG2 <- G2(image,2,1.5,0)
                        newSet <- newSet U {(imgG2,y(w))}
                END
        END
// aggiunta immagini originali
newSet <- newSet U {(X,Y)}
// mescola set
newSet <- shuffle(newSet)
RETURN newSet
END_function
```

PSEUDOCODICE SOTTOFUNZIONI SHARPEN

```
G1(image,[rows_filtro cols_filtro],devStd)
// creo filtro gaussiano
PSF <- fspecial('gaussian',[rows_filtro cols_filtro],devStd)
// applico deconvoluzione
image <- deconvblind(image,PSF)
RETURN image
END_function

AVERG(image,[rows_filtro cols_filtro])
// creo filtro average
PSF <- fspecial('average',[rows_filtro cols_filtro])
// applico deconvoluzione
image <- deconvblind(image,PSF)
RETURN image
END_function

// amount: durezza effetto smussamento
// threshold: contrasto minimo richiesto per un pixel per considerarlo di bordo
G2(image,devStd,durezza,min_contrasto)
// uso funzione built-in di matlab
image <- imsharpen(image,'Radius'=devStd,'Amount'=durezza,'Threshold'=min_contrasto)
RETURN image
END_function
```

PSEUDOCODICE IMAGE ENHANCEMENT

```
REQUISITI:
X, dataset immagini
Y, etichette corrispondenti a X
p1, probabilità trasformata 1
p2, probabilità trasformata 2
p3, probabilità trasformata 3
FUNCTION IMGE(X,Y,p1,p2,p3)
FOR (x,y) in (X,Y)
        image <- rgb2gray(x)
        (s1,s2) <- (rows_image,cols_image)
        // valore intensità più alta
        M <- max(image)
        // riduco i valori
        image <- image.*((M-30)/M)
        K <- // kernel gaussiano
        // creo pose aggiuntive in cascata
        image1 <- conv(K/16,image)
        image2 <- image.^1.15
        image3 <- conv(image2,K/8)
        // eseguo resize e prendo i valori reali
        image1 <- real(imresize(image1,[s1 s2]))
        image2 <- real(imresize(image2,[s1 s2]))
        image3 <- real(imresize(image3,[s1 s2]))
        IF rand(0,1) < p1
                newSet <- {(image1,y)}
        END
        IF rand(0,1) < p2
                newSet <- {(image2,y)}
        END
        IF rand(0,1) < p3
                newSet <- {(image3,y)}
        END

// aggiunta immagini originali
newSet <- newSet U {(X,Y)}
// mescola set
newSet <- shuffle(newSet)
RETURN newSet
END_function
```

KERNEL USATO IN IMGE

```
K=[-1 -1 -1 -1 -1;
   -1  2  2  2 -1;
   -1  2  8  2 -1;
   -1  2  2  2 -1;
   -1 -1 -1 -1 -1];
```

Capitolo 4

Fase sperimentale

4.1 Aspetti generali

Per rendere più veloce l'allenamento delle reti e avere una precisione migliore, oltre al data augmentation, visto il numero dei dati a disposizione, sono state usate altre due tecniche: il transfer-learning, cioè l'utilizzo di una rete pre-addestrata su un set molto ampio, nel nostro caso AlexNet e ResNet50 pre-addestrate col dataset di ImageNet, e il fine-tuning, ovvero il rimpiazzo degli ultimi livelli con altri livelli di classificazione adeguando il numero delle classi.
Prima di iniziare l'addestramento è importante configurare gli iperparametri: la dimensione dei mini-batch, il learning rate, il numero di fold e il numero di epoche.
Per l'addestramento di AlexNet, la quale prende in ingresso immagini della dimensione 227x227, sono stati usati i seguenti iperparametri:

- Numero epoche: 20
- Numero mini-batch: 30
- Learning rate: 0.0001
- Numero di fold: 5

Per l'addestramento di ResNet50, con immagini in ingresso di dimensione 224x224, per motivi computazionali, è stato usato un numero di mini-batch e di fold inferiore rispetto ad AlexNet:

- Numero epoche: 3-4
- Numero mini-batch: 5
- Learning rate: 0.0001
- Numero di fold: 1

Le tecniche di data augmentation usate sono le seguenti: rotazione casuale (CDA) tra 0° e 360°, straightening, scala dell'immagine lungo gli assi X e Y di un valore casuale compreso tra 0.7 e 1.3, riflessione casuale lungo entrambi gli assi, traslazione lungo entrambe le direzioni compresa tra -30 e 30, FT, DCT, sharpening e image enhancement.

4.2 Dataset

Il dataset iniziale è stato diviso in 5 fold, creati in modo casuale, uguali composti da 2986 immagini. All'inizio del programma, dopo aver impostato gli iperparametri, i vari fold, che cambiano iterativamente, vengono divisi in train pattern contenente 2387 immagini e test pattern formato dalle restanti 599.
Come detto in precedenza, con AlexNet sono stati usati tutti i fold, con ResNet50, dato che prestazioni maggiori rispetto ad AlexNet e sia per motivi computazionali, è stato usato un solo fold. Nei test, invece, per entrambe sono stati usati tutti i fold.

Capitolo 5

Risultati sperimentali

Questi sono i risultati ottenuti dai test dopo gli allenamenti descritti nel Capitolo 4.

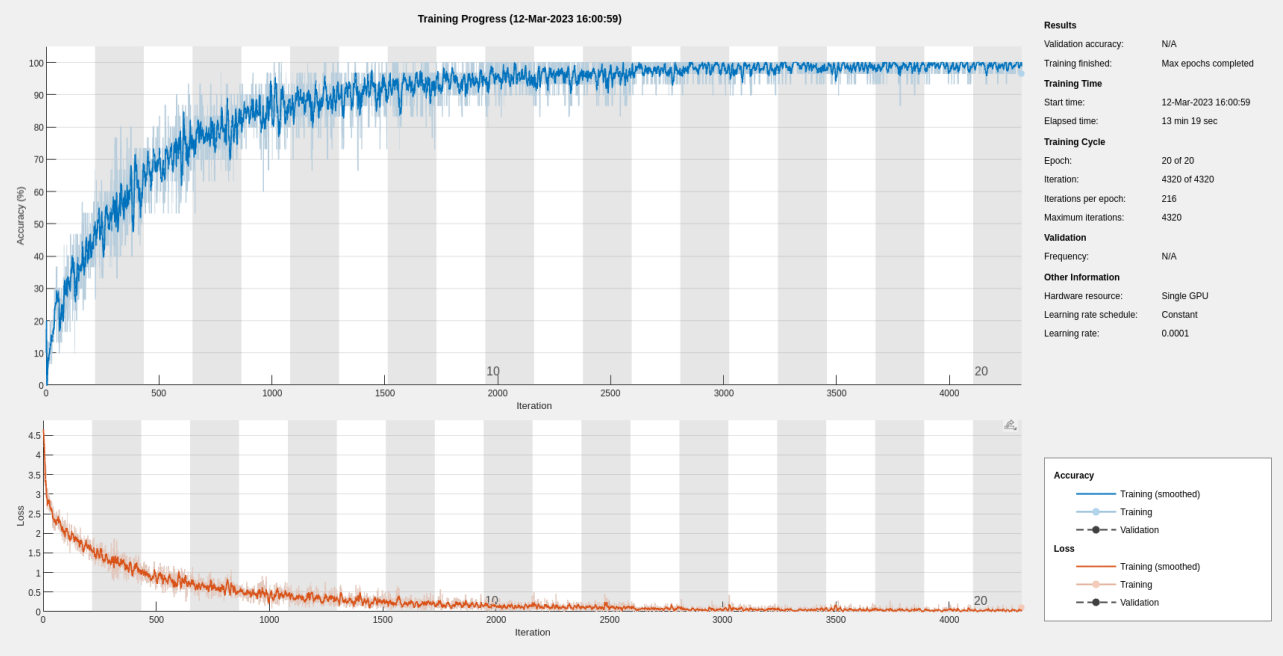
ANet	FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5
CDA	0.8647	0.9782	0.9883	0.9849	0.9766
STR	0.8481	0.9866	0.9917	0.9800	0.9950
BASE	0.8130	0.9850	0.9833	0.9833	0.9833
FOURIER	0.8815	0.9967	1	0.9983	0.9967
SHARP	0.8864	1	0.9983	1	0.9949
IMGE	0.8932	0.9967	0.9983	1	0.9983

Tabella 1.1: Accuratezze ottenute sui test set con AlexNet

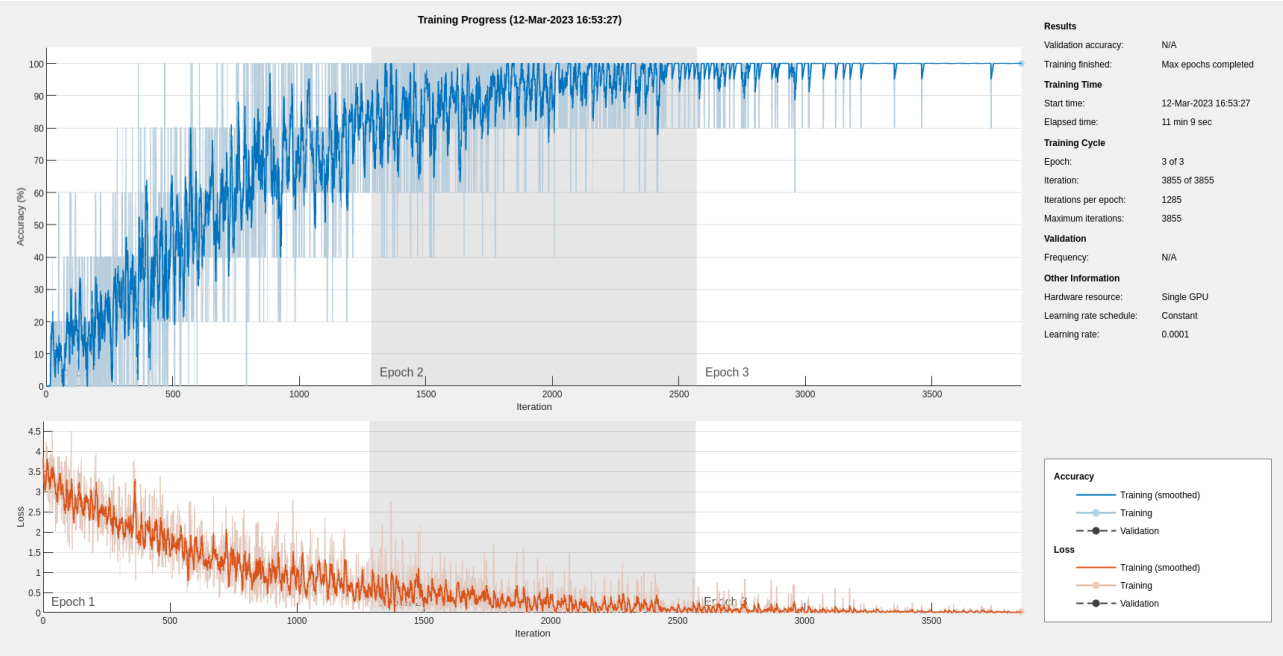
RNet50	FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5
CDA	0.8631	0.9733	0.9783	0.9783	0.9783
STR	0,8574	0,9752	0,9712	0,9811	0,9836
BASE	0.8865	0.9800	0.9933	0.9866	0.9833
FOURIER	0.8898	0.9917	0.9900	0.9967	0.9933
SHARP	0.8982	0.9983	0.9950	0.9917	0.9917
IMGE	0.9098	0.9950	1	0.9983	0.9983

Tabella 1.2: Accuratezze ottenute sui test set con ResNet50

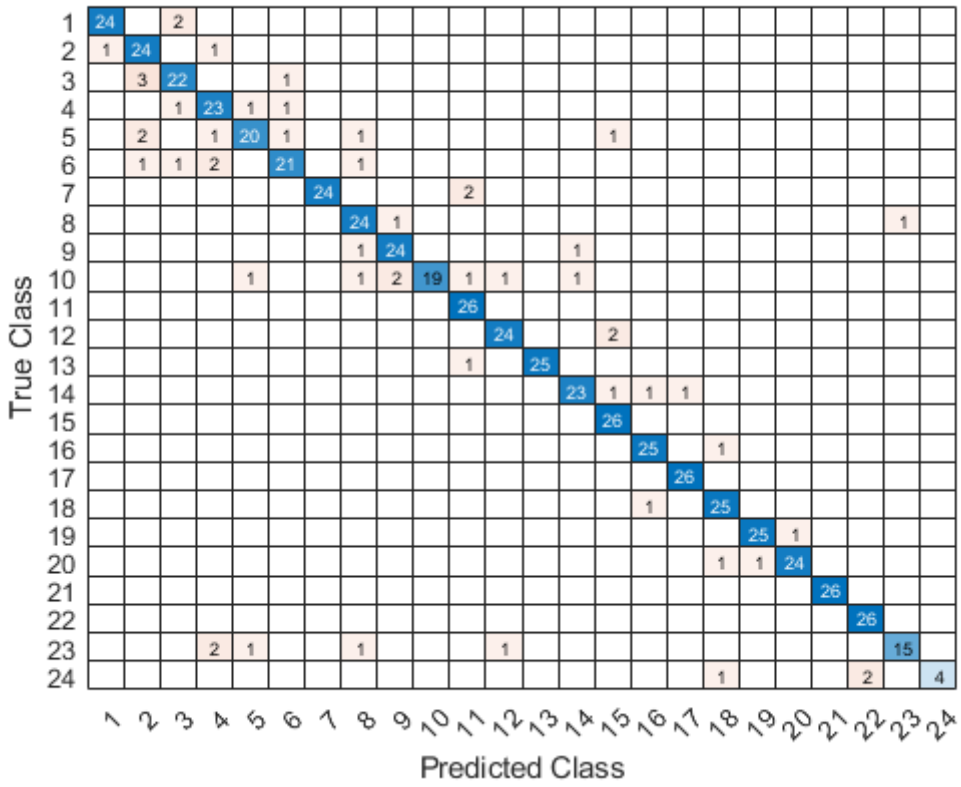
Da questi risultati si potrebbe dire che le trasformazioni hanno avuto un buon successo, però la bassa quantità di dati, seppur aumentata, restituisce accuratezze significativamente diverse tra il primo fold e gli altri.
I risultati migliori, sia con AlexNet che con ResNet50, sono per la maggior parte ottenuti da Image Enhancement (IMGE), metodo analizzato, mostrando l'efficacia di questo tipo di trasformazioni.



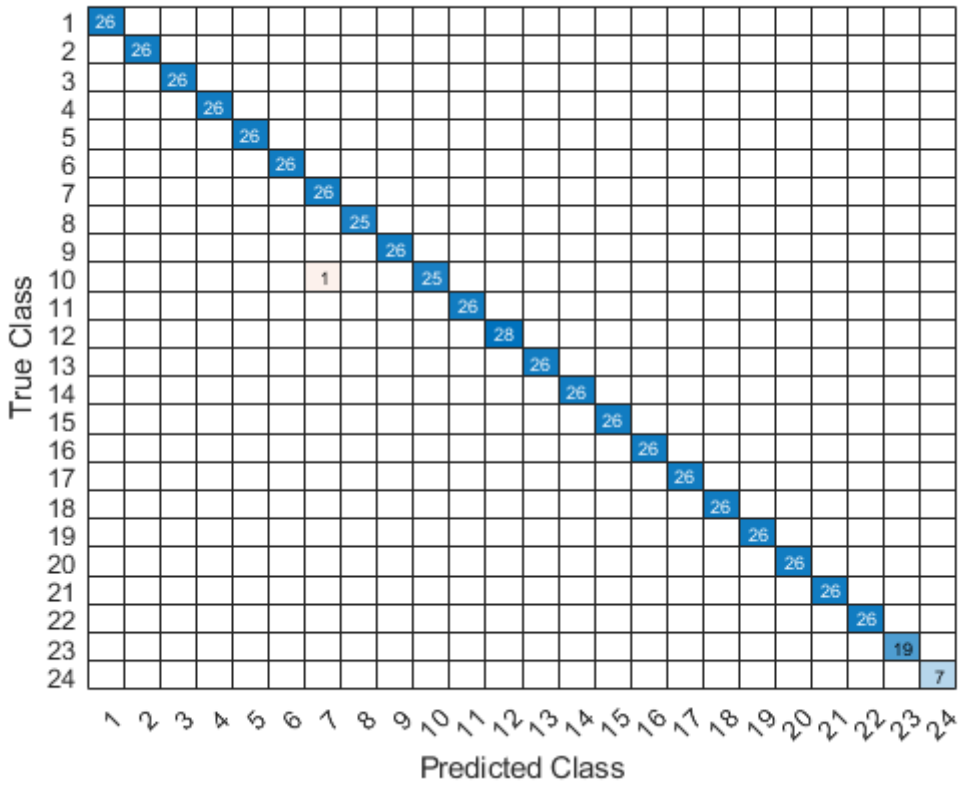
Andamento del training di AlexNet con Image Enhancement



Andamento del training di ResNet50 con Image Enhancement



Matrice di confusione di Image Enhancement con ResNet50 – fold 1



Matrice di confusione di Image Enhancement con ResNet50 – fold 1

Capitolo 6

Conclusioni

Il tipo di CNN usata, come si può vedere dai grafici del training, migliora significativamente i risultati a discapito però di un tempo di allenamento e di classificazione maggiori. Infatti, ResNet50, rete molto più profonda rispetto AlexNet, raggiunge la massima accuratezza in sole 3 epoche, contro le 20 usate per AlexNet; per entrambe le reti, anche se con grande differenza di epoche, sono state allenate con gli stessi tempi.
Come si può vedere dalla tabella nel capitolo precedente, il metodo Image Enhancement (IMGE) come Data Augmentation risulta quello con prestazioni migliori.
Le immagini con questa trasformazione hanno effettivamente più informazioni in quanto viene migliorata la risoluzione con lo sharpening e aumentato il contrasto, portando così ad avere immagini più definite.
L'unico problema è la grande differenza nelle accuratezze tra il primo fold e gli altri, facendo venire qualche dubbio sull'affidabilità dei risultati.
Visto questo problema si è pensato di aumentare il dataset in modo significativo, dato che attualmente raggiunge al massimo 4 volte la dimensione originale.
Ulteriori test verranno eseguiti con un dataset più grande e unendo diversi metodi di data augmentation controllando che non ci siano grandi differenze tra le accuratezze.

Bibliografia

[1]: *R. S. Remya, H. Prasad, S. Hariharan and C. Gopakumar, "Chromosome Image Enhancement for Efficient Karyotyping," 2022 International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 2022, pp. 1-6, doi: 10.1109/ICITIIT54346.2022.9744195.*

[2]: *Team, D. S. (2020, December 15). Una passeggiata di Alexnet - Programmazione. DATA SCIENCE. Retrieved March 13, 2023, from https://datascience.eu/it/programmazione/una-passeggiata-di-alexnet/*

[3]: *Deep Network designer. Rete neurale convoluzionale ResNet-50 - MATLAB resnet50 - MathWorks Italia. (n.d.). Retrieved March 13, 2023, from https://it.mathworks.com/help/deeplearning/ref/resnet50.html*

[4]: *CIR-NET: Automatic classification of human chromosome ... - IEEE xplore. (n.d.). Retrieved March 13, 2023, from https://ieeexplore.ieee.org/document/9120197/*

[5]: *Straighten. MathWorks. (n.d.). Retrieved March 13, 2023, from https://it.mathworks.com/matlabcentral/fileexchange/72266-straighten?tab=discussions*

[6]: *Nanni, L., Paci, M., Brahnam, S., & Lumini, A. (2022, August 22). Feature transforms for Image Data Augmentation - neural computing and applications. SpringerLink. Retrieved March 13, 2023, from https://link.springer.com/article/10.1007/s00521-022-07645-z*