



UNIVERSITÀ DEGLI STUDI DI CASSINO E DEL LAZIO MERIDIONALE

Corso di Laurea Magistrale in Ingegneria Informatica

Blood Cells Detection and Classification using Deep Learning and Machine Learning

Versione 0.1

**Cannavale Achille
Colacicco Nunziamaria
La Torre Noemi**

July 4, 2025

Contents

1		3
1.1	Introduzione	3
1.2	Strumenti utilizzati	4
1.3	Composizione Dataset	4
2	Deep Learning	6
2.1	Trasformazioni	6
2.2	Personalizzazione delle Anchor Boxes	8
2.3	Focal Loss	8
2.4	Architetture del Modello	9
2.5	Valutazione	11
2.5.1	Iperparametri	11
2.5.2	Visualizzazione dei Risultati	12
2.6	Estrazione delle Feature	13
3	Machine Learning	15
3.1	Introduzione	15
3.2	Preprocessing	16
3.2.1	BorderlineSMOTE	16
3.2.2	Variance Threshold	16
3.3	Feature Selection: SelectKBest	17
3.4	Dimension Reduction: LDA	17
3.5	Classifier	17
3.6	Title Formatting	17
3.6.1	Page Geometry	17
3.7	Image Positioning	17
3.8	Defined Environments	19
3.8.1	Writing Equations	20
3.8.2	Designing a Table	20
4	Plotting your data using PGF/TikZ	22
4.1	Introduction	22
4.1.1	A Simple 2D Plot	23
4.1.2	Plotting 3D plots	24

Chapter 1

Contents

1.1	Introduzione	3
1.2	Strumenti utilizzati	4
1.3	Composizione Dataset	4

1.1 Introduzione

La malaria è una delle principali malattie infettive a livello globale, con un impatto significativo sulla salute pubblica, in particolare nei paesi tropicali e subtropicali. Tra i diversi parassiti responsabili di questa patologia, il **Plasmodium vivax** rappresenta una delle specie più diffuse. La diagnosi precoce e accurata è fondamentale per un trattamento tempestivo ed efficace, e l'analisi microscopica degli strisci ematici rimane uno degli strumenti diagnostici più affidabili. In questo progetto, viene affrontato il problema della rilevazione e classificazione automatica delle cellule del sangue infettate da P. vivax, attraverso tecniche di machine e deep learning. Utilizzando il dataset **P. vivax malaria infected human blood smears**, che contiene oltre **1.300** immagini microscopiche annotate con più di **86.000** cellule identificate e classificate, si propone una pipeline [FOTO] che combina **Deep Learning** e **Machine Learning** per:

- individuare automaticamente le regioni di interesse (ROI) contenenti cellule
- estrarre le feature significative dalle ROI
- classificare ogni cellula in una delle categorie predefinite, tra cui cellule sane (red blood cell e leukocyte) e diversi stadi del parassita infetto (trophozoite, ring, gametocyte, schizont)

L'analisi di questo lavoro ha evidenziato una marcata sbilanciatura del dataset, aspetto che ha richiesto un'attenta progettazione delle strategie di addestramento.

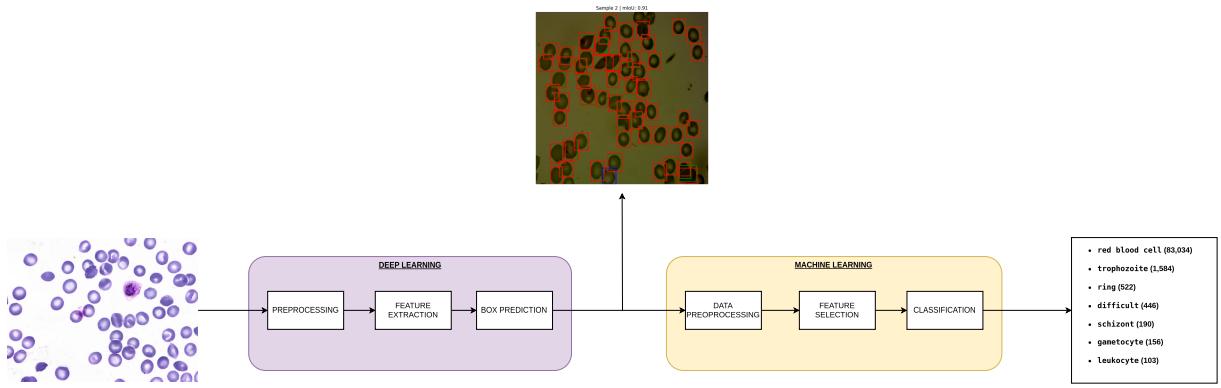


Figure 1.1: Pipeline Generale del nostro progetto, suddivisa in una fase di Deep Learning per la rilevazione delle ROI e l'estrazione delle features e una fase di Machine Learning per la classificazione delle cellule.

1.2 Strumenti utilizzati

Per lo sviluppo del progetto sono stati adottati diversi strumenti software, suddivisi in base alle funzionalità specifiche per il *deep learning*, il *machine learning* e l'analisi dei dati. La scelta di ciascuno di essi è stata guidata da criteri di efficienza, semplicità d'uso e ampia diffusione nella comunità scientifica.

- **Deep Learning:** *PyTorch*

Utilizzato per la progettazione, l'addestramento e la valutazione dei modelli di deep learning. PyTorch si distingue per la sua flessibilità e l'approccio dinamico alla costruzione delle reti neurali, risultando particolarmente adatto per attività di ricerca e prototipazione rapida.

- **Machine Learning:** *Scikit-learn*

Impiegato per l'implementazione di algoritmi di machine learning tradizionali, grazie alla sua vasta collezione di modelli predefiniti e strumenti per la valutazione delle prestazioni.

- **Strumenti Ausiliari:**

- *NumPy* e *Pandas*: utilizzati per la manipolazione, la pulizia e l'analisi dei dati, fondamentali nella fase di preprocessing e gestione dei dataset.
- *Matplotlib*: utilizzata per la visualizzazione grafica dei risultati, utile per l'analisi esplorativa e la presentazione delle performance dei modelli.

1.3 Composizione Dataset

Il dataset utilizzato per questo progetto è composto da **1.328 immagini** contenenti in totale **86.035 oggetti annotati**. Ogni oggetto rappresenta una singola cellula o elemento biologico rilevato nelle immagini microscopiche, ed è associato a una delle seguenti **sette classi**:

- red blood cell (83.034)

- trophozoite (1.584)
- ring (522)
- difficult (446)
- schizont (190)
- gametocyte (156)
- leukocyte (103)

Come si può osservare dalla distribuzione, il dataset presenta un **forte sbilanciamento tra le classi**, con la stragrande maggioranza degli oggetti appartenenti alla categoria `red blood cell`, che rappresenta da sola circa il **96,5%** del totale.

Questo sbilanciamento costituisce una **sfida rilevante per i modelli di classificazione**, in quanto tende a favorire la previsione della classe dominante a discapito delle classi minoritarie, che possono essere trascurate o predette con bassa accuratezza.

Per affrontare questo problema, durante lo sviluppo sono state considerate diverse strategie, tra cui:

- il bilanciamento del dataset,
- l'uso di tecniche di *data augmentation*,
- l'assegnazione di pesi differenti alle classi durante l'addestramento,
- e l'adozione di metriche di valutazione più robuste rispetto alla sola accuratezza, come *precision*, *recall* e *F1-score*.

Chapter 2

Deep Learning

Contents

2.1	Trasformazioni	6
2.2	Personalizzazione delle Anchor Boxes	8
2.3	Focal Loss	8
2.4	Architetture del Modello	9
2.5	Valutazione	11
2.5.1	Iperparametri	11
2.5.2	Visualizzazione dei Risultati	12
2.6	Estrazione delle Feature	13

2.1 Trasformazioni

Per garantire una buona generalizzazione del modello, e ridurre l'overfitting, è stata utilizzata la tecnica di *data augmentation* tramite la libreria Albumentations. Dopo una fase di sperimentazione, sono state selezionate le seguenti trasformazioni come le più efficaci per il nostro caso di studio:

Trasformazione	Parametri
A.HorizontalFlip	p=0.5
A.VerticalFlip	p=0.5
A.ColorJitter	brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1, p=0.3
A.RandomBrightnessContrast	p=0.3
A.MotionBlur	p=0.2
A.GaussNoise	p=0.2
A.CLAHE	p=0.2
A.CoarseDropout	num holes ange=(3, 6), hole height range=(10, 20), hole width range=(10, 20), fill="random uniform", p=0.2
A.Resize	800x800
A.Normalize	mean=[0.7205, 0.7203, 0.7649], std=[0.2195, 0.2277, 0.1588]

Table 2.1: Lista delle trasformazioni utilizzate per il dataset.

In particolare, i parametri della normalizzazione sono stati calcolati dal dataset stesso:

Parametro	Valore
Media RGB	(0.7205, 0.7203, 0.7649)
Deviazione standard RGB	(0.2195, 0.2277, 0.1588)

Table 2.2: Valori di media e deviazione standard per la normalizzazione delle immagini.

Una delle trasformazioni che ha avuto il maggiore impatto sulle prestazioni è stata la CoarseDropout (fig. 2.1), che oscura regioni rettangolari casuali dell'immagine, simulando occlusioni ottiche e migliorando la robustezza del modello in scenari reali dove gli oggetti potrebbero essere parzialmente visibili.

```

A.CoarseDropout(
    num_holes_range=(3, 6),           # Numero casuale di buchi tra 3 e 6
    hole_height_range=(10, 20),        # Altezza dei buchi tra 10 e 20 px
    hole_width_range=(10, 20),         # Larghezza dei buchi tra 10 e 20 px
    fill="random_uniform",           # Riempie i buchi con valori casuali
    p=0.2                            # Probabilità di applicazione 20%
)

```

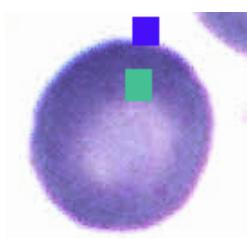


Figure 2.1: Esempio di CoarseDropout applicata su una cellula.

2.2 Personalizzazione delle Anchor Boxes

Per migliorare le prestazioni del modello di object detection, abbiamo effettuato un'analisi approfondita della distribuzione delle dimensioni delle bounding box presenti nel dataset. Le anchor boxes predefinite, come quelle utilizzate nei dataset generici come COCO, non risultavano ottimali per il nostro caso specifico, in quanto le cellule da rilevare presentano dimensioni e proporzioni piuttosto omogenee e differenti rispetto agli oggetti comuni.

A tal fine, abbiamo estratto tutte le bounding box del dataset di addestramento e calcolato le dimensioni (larghezza e altezza) di ciascuna. I dati così ottenuti sono stati utilizzati come input per un algoritmo di clustering K-Means, impostato per individuare i centroidi rappresentativi delle dimensioni più ricorrenti. In particolare, è stato scelto un numero di cluster pari a 6, corrispondente al numero di anchor boxes da definire. Le nuove ancore suggerite sono state quindi utilizzate per sostituire quelle standard all'interno del modello.

Questa ottimizzazione ha permesso un miglior allineamento tra le proposte del modello e le reali dimensioni degli oggetti nel dataset, contribuendo a un incremento consistente delle metriche di accuratezza. L'impatto dell'adozione delle nuove anchor boxes è illustrato nella Tabella 2.3, dove si evidenzia un miglioramento del valore di mAP rispetto alla configurazione predefinita.

Configurazione Anchor	mAP@0.5
Default (COCO-style)	valore
Personalizzate (K-Means)	0.78

Table 2.3: Confronto tra ancore standard e personalizzate.

2.3 Focal Loss

Un aspetto cruciale di questo problema è il forte sbilanciamento del dataset: le cellule sane (globuli rossi e leucociti) sono molto più numerose rispetto alle cellule infette, che rappresentano una minoranza spesso difficile da distinguere.

Questo squilibrio comporta che durante l'addestramento i modelli tendano a privilegiare la classificazione corretta delle classi maggioritarie, a discapito di quelle minoritarie, che sono però di massima importanza per una diagnosi accurata e precoce della malaria.

Per mitigare questo problema, abbiamo adottato la *Focal Loss* (Lin et al., 2017), una funzione di perdita che estende la Cross Entropy introducendo un termine di modulazione. Tale termine riduce il peso degli esempi facili (correttamente classificati) e concentra l'apprendimento sugli esempi difficili o mal classificati, tipicamente appartenenti alle classi minoritarie. La formulazione della Focal Loss è:

$$\text{FL}(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t)$$

dove p_t è la probabilità stimata per la classe corretta, α è un fattore di bilanciamento tra classi e γ , detto "focusing parameter", controlla la riduzione del peso degli esempi facili.

Nel nostro caso, i parametri $\alpha = 0.25$ e $\gamma = 3.0$ sono stati scelti per ottenere un buon equilibrio tra stabilità e miglioramento delle performance. L'uso della Focal Loss ha permesso di:

- ridurre l'impatto degli esempi di sfondo abbondanti,
- aumentare la sensibilità verso le cellule infette meno rappresentate,
- migliorare la convergenza e la capacità discriminativa del modello.

La funzione di perdita per la regressione delle bounding box è stata mantenuta invariata, affidandosi alla *Smooth L1 Loss*, che risulta efficace per la localizzazione precisa delle cellule. Questa scelta ha avuto un impatto positivo sulle metriche di accuratezza e recall, fondamentali in ambito medico per evitare falsi negativi.

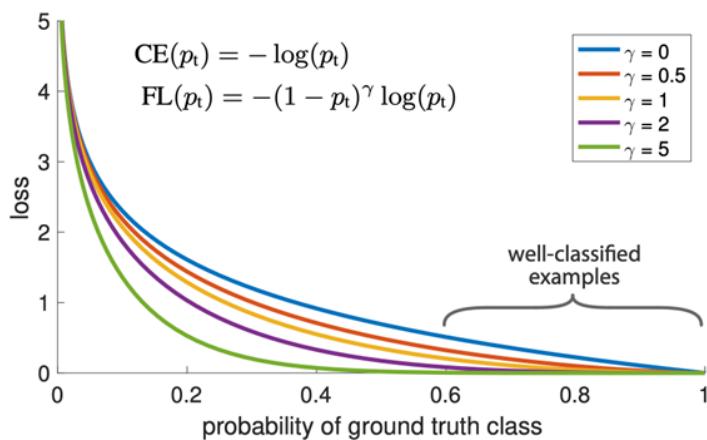


Figure 2.2: Grafico Focal Loss.

2.4 Architetture del Modello

Durante lo sviluppo del progetto sono state testate diverse architetture di rete neurale, combinando vari **backbone** con due principali tipologie di **head**: **Faster R-CNN** e **RetinaNet**. L'obiettivo era identificare la combinazione più efficace in termini di accuratezza, capacità di generalizzazione e prestazioni computazionali.

In un modello di object detection, il **backbone** funge da estrattore di feature: riceve l'immagine in input e produce mappe di caratteristiche utili per individuare e classificare gli oggetti. I backbone testati nel progetto includono:

- **ResNet50**: una rete convoluzionale profonda basata su **residual blocks**, che permette un'ottima propagazione del gradiente anche in reti molto profonde.
- **ResNet101**: una versione più profonda di ResNet50, che migliora la rappresentazione delle feature a scapito di un maggiore costo computazionale.
- **MobileNet**: una rete più leggera ed efficiente, pensata per dispositivi mobili, ma con performance inferiori in contesti complessi.

- **ResNeXt101**: un'estensione di ResNet che introduce la dimensione della cardinalità (cioè il numero di percorsi paralleli in ogni blocco), migliorando la capacità di rappresentazione senza aumentare drasticamente i parametri.

Il **detection head**, invece, riceve le feature estratte e produce le predizioni finali (classi e bounding box). Le due architetture di head utilizzate sono:

- **Faster R-CNN**: un approccio a due stadi, in cui un **Region Proposal Network (RPN)** genera proposte di oggetti che vengono successivamente classificate e regolarizzate. È noto per l'elevata accuratezza, ma è più lento rispetto ad approcci monostadio.
- **RetinaNet**: un metodo **one-stage** che introduce la **Focal Loss** per gestire lo squilibrio tra oggetti e background, migliorando le performance su oggetti piccoli o rari. Offre un buon compromesso tra accuratezza e velocità.

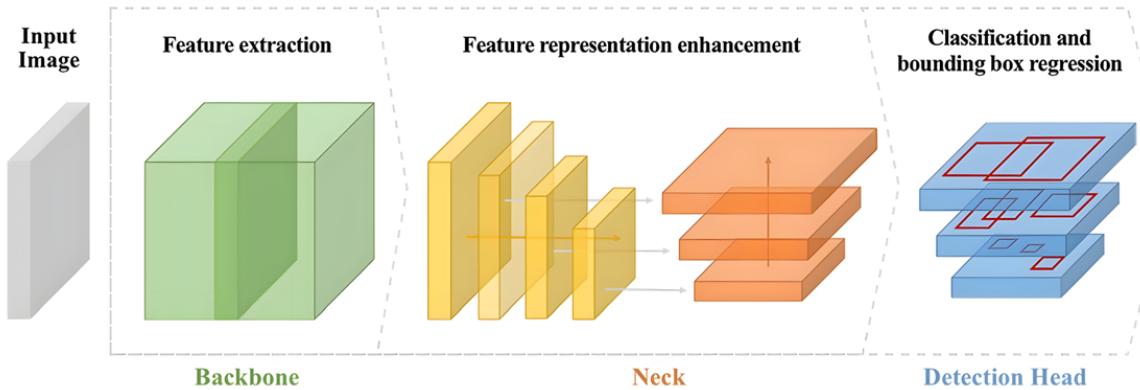


Figure 2.3: Pipeline Generale di una rete.

Le configurazioni testate sono le seguenti:

- **Backbone:** ResNet50 **Head:** Faster R-CNN
- **Backbone:** ResNet101 **Head:** Faster R-CNN
- **Backbone:** MobileNet **Head:** Faster R-CNN
- **Backbone:** ResNet50 **Head:** RetinaNet
- **Backbone:** ResNeXt101 **Head:** Faster R-CNN
- **Backbone:** ResNet101 **Head:** RetinaNet

L'architettura che ha fornito i migliori risultati, sia in termini di accuratezza sia di generalizzazione, è stata quella composta da **ResNeXt101** come backbone e **Faster R-CNN** come head. In Figura 2.4 è riportata una rappresentazione schematica della pipeline del modello.

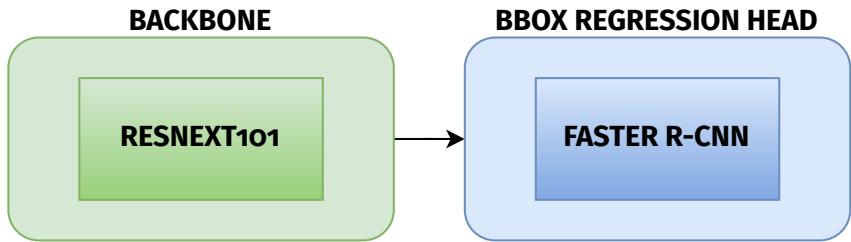


Figure 2.4: Pipeline Generale del miglior modello, composto da ResNeXt101 come backbone e Faster R-CNN come head.

Metric	Value
mIoU	0.8871
Precision	0.9733
Recall	0.9683
F1 Score	0.9708
mAP	0.7397
mAP@50	0.9479
mAP@75	0.8874

Table 2.4: Evaluation metrics of ResNeXt101 + Faster R-CNN model.

2.5 Valutazione

2.5.1 Iperparametri

Parameter	Value
BATCH_SIZE	4
NUM_EPOCHS	5
LEARNING_RATE	0.0003
WEIGHT_DECAY	0.001
PATIENCE	5
OPTIM	AdamW
SCHEDULER	ReduceLROnPlateau
CustomAnchor	True
loss	FocalLoss
VAL_SIZE	0.2

Table 2.5: Configurazione degli Iperparametri del Miglior Modello.

Gli iperparametri riportati nella Tabella 2.5 sono il risultato di un'attenta fase di sperimentazione in cui sono stati confrontati diversi valori e configurazioni. In particolare, si è osservato che un numero di epoche pari a 5 è sufficiente per ottenere una buona convergenza: aumentare il numero di epoche a 10 non ha portato a miglioramenti significativi in termini di accuratezza o generalizzazione.

Per quanto riguarda l'ottimizzatore, si è confrontato l'uso di Adam e **AdamW**, con quest'ultimo

che ha dimostrato maggiore stabilità e performance migliori, probabilmente grazie alla gestione più efficace della regolarizzazione tramite **weight decay**.

La politica di gestione del learning rate è stata valutata tra **ReduceLROnPlateau** e **CosineAnnealingLR**. **ReduceLROnPlateau** si è rivelata più adatta al nostro task, poiché riduceva il learning rate solo quando la metrica di validazione non migliorava, consentendo un adattamento più dinamico e mirato rispetto al coseno annealing, che segue un programma fisso.

Infine, la scelta della funzione di perdita si è concentrata tra la **standard loss** e la **Focal Loss**. Quest'ultima ha mostrato un vantaggio concreto nella gestione dello sbilanciamento delle classi presenti nel dataset, migliorando la capacità del modello di rilevare correttamente anche le classi meno rappresentate.

2.5.2 Visualizzazione dei Risultati

In questa sezione vengono mostrati i risultati ottenuti dall'addestramento del modello. La Figura 2.5 presenta il grafico dell'andamento della **training loss** e della **validation loss** durante le epoch, evidenziando una buona convergenza e un'assenza di overfitting significativo.

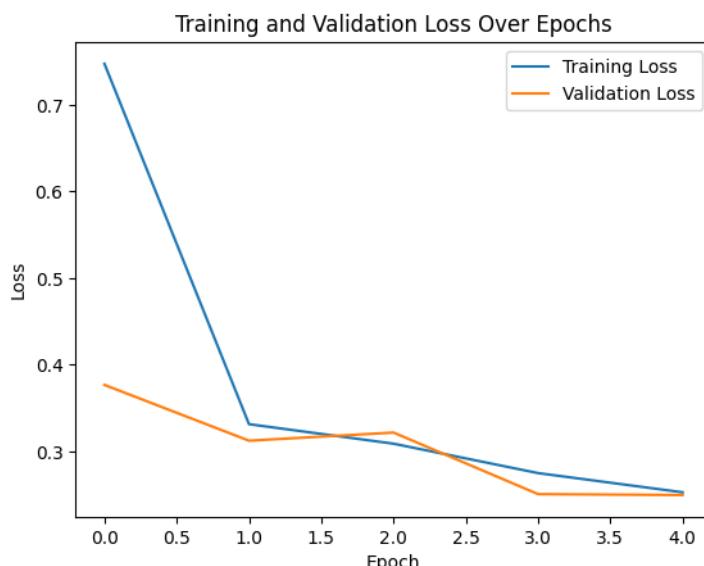


Figure 2.5: Grafico della Training Loss e della Validation Loss.

La Figura 2.6 mostra un esempio di output del modello su un'immagine del set di test. Le regioni di interesse (ROI) delle cellule sono rappresentate tramite rettangoli di diversi colori: le ROI esatte del dataset sono in **verde**, quelle predette dal modello in **rosso**, mentre in **blu** sono evidenziate le cellule non rilevate. Il modello raggiunge un valore medio di **mIoU** pari a **0.88**, confermando un'alta precisione nella localizzazione degli oggetti, con un'ottima corrispondenza tra predizioni e annotazioni di riferimento.

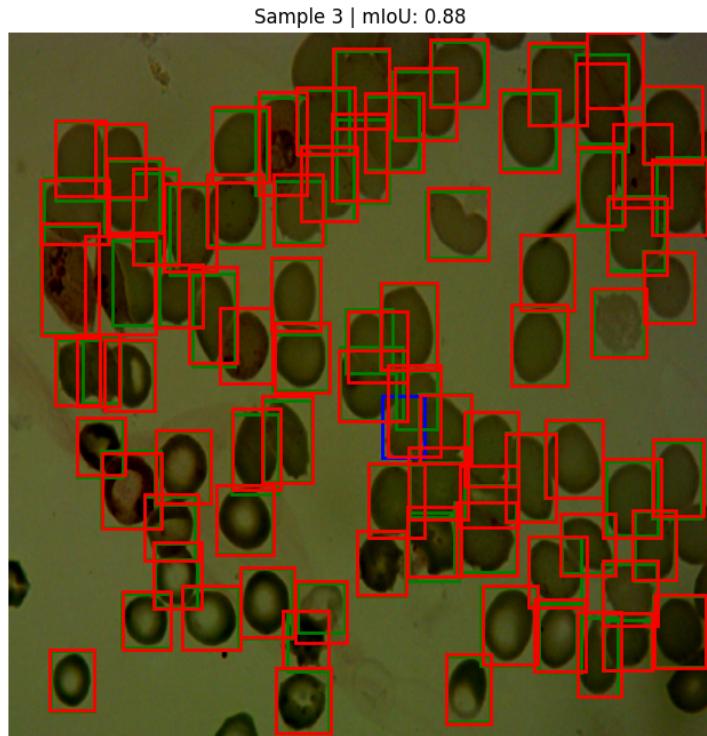


Figure 2.6: Risultato di esempio. In rosso sono evidenziate le bounding box correttamente previste dal modello, in blu quelle mancate.

2.6 Estrazione delle Feature

Per l’analisi delle rappresentazioni apprese dal modello, abbiamo implementato una procedura di estrazione delle caratteristiche visive associate alle ROIs predette. Il metodo si basa su un modello di object detection preaddestrato, nello specifico una variante di **Faster R-CNN con Feature Pyramid Network (FPN)**, che consente di sfruttare rappresentazioni multi-scala per migliorare la localizzazione e classificazione di oggetti di dimensioni variabili.

Durante l’inferenza, ogni immagine del dataset viene elaborata per ottenere le predizioni del modello, comprensive di bounding boxes e punteggi di confidenza. Le predizioni con uno score inferiore a `score_threshold` (default: 0.5) vengono scartate. Le restanti predizioni vengono confrontate con i ground truth tramite l’Intersection over Union (IoU), mantenendo solo quelle che superano una soglia di similarità. Questo matching consente di associare ciascuna ROI predetta alla corrispondente annotazione reale.

Le feature vengono estratte dai livelli intermedi del modello tramite la **backbone con FPN**, che restituisce una gerarchia di mappe di attivazione a risoluzioni diverse. A partire da queste mappe, il modulo `box_roi_pool` seleziona e aggredisce le porzioni rilevanti in base alle bounding boxes predette. Le feature pooled vengono poi elaborate dalla `box_head`, e ulteriormente arricchite concatenando i **logit del classificatore** (`cls_score`), ottenendo così una rappresentazione congiunta che riflette sia le informazioni visive locali sia le risposte semantiche del modello.

Ogni vettore di feature viene associato a metadati esplicativi, tra cui:

- l’identificativo dell’immagine

- le coordinate della bounding box predetta
- il punteggio di confidenza
- l'etichetta del ground truth associato

Tutti i dati raccolti vengono infine aggregati in un dataframe strutturato, che unisce metadati e rappresentazioni numeriche. Il risultato viene salvato in formato CSV per facilitarne l'analisi quantitativa e la visualizzazione.

Chapter 3

Machine Learning

Contents

3.1	Introduzione	15
3.2	Preprocessing	16
3.2.1	BorderlineSMOTE	16
3.2.2	Variance Threshold	16
3.3	Feature Selection: SelectKBest	17
3.4	Dimension Reduction: LDA	17
3.5	Classifier	17
3.6	Title Formatting	17
3.6.1	Page Geometry	17
3.7	Image Positioning	17
3.8	Defined Environments	19
3.8.1	Writing Equations	20
3.8.2	Designing a Table	20

3.1 Introduzione

La parte di Machine Learning ha come input il file csv generato nella parte di Deep Learning, che contiene le feature estratte dalle ROIs.

In un primo momento, la struttura generale di questa parte era costituita da un solo Classificatore, che aveva il compito di distinguere le cellule in tutte e 7 le classi. Tuttavia, a causa della grande **class imbalance** del dataset (3.1), si è deciso di adottare un approccio diverso, creando 2 classificatori distinti:

- **Classificatore 1: RBC vs Infected Cells**

Questo classificatore ha il compito di distinguere le cellule sane (RBC e Leukocyte) dalle cellule infette (tutte le altre classi).

- **Classificatore 2: Infected Cells Classification**

Questo classificatore si occupa di distinguere le cellule infette tra loro, classificandole in trophozoite, ring, difficult, schizont e gametocyte

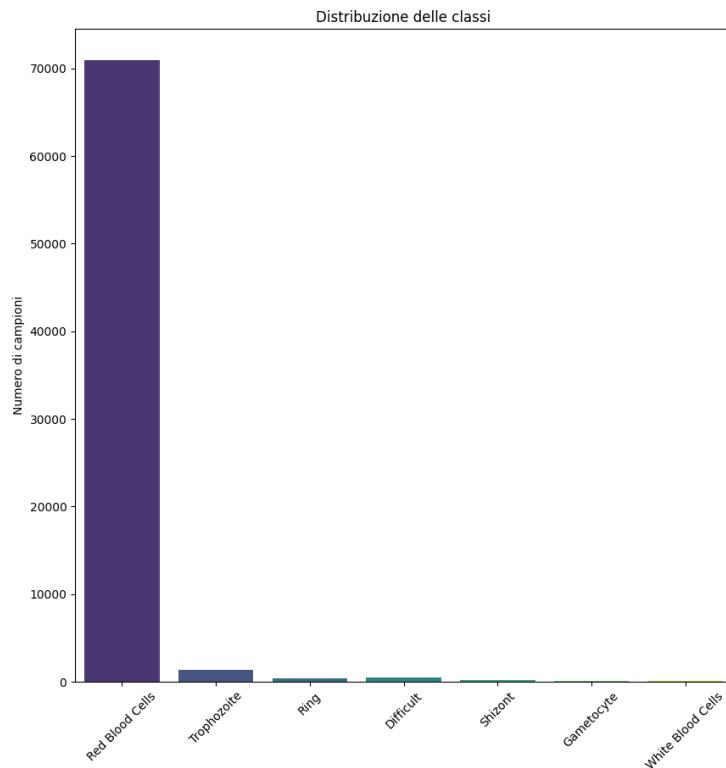


Figure 3.1: Distribuzione delle classi nel dataset prima del preprocessing.

3.2 Preprocessing

3.2.1 BorderlineSMOTE

Per affrontare il problema dello sbilanciamento delle classi, è stato utilizzato l'algoritmo **BorderlineSMOTE**, una variante del Synthetic Minority Over-sampling Technique (SMOTE). Questo metodo genera nuovi campioni sintetici per le classi minoritarie, migliorando la capacità del modello di apprendere le caratteristiche distintive delle classi meno rappresentate.

3.2.2 Variance Threshold

Per ridurre la dimensionalità del dataset e migliorare le prestazioni del modello, è stata applicata la tecnica di **Variance Threshold**. Questa tecnica rimuove le feature con una varianza inferiore a una soglia specificata, eliminando le caratteristiche che non apportano informazioni significative. Nel nostro caso, è stata impostata una soglia di $1e-5$.

3.3 Feature Selection: SelectKBest

3.4 Dimension Reduction: LDA

3.5 Classifier

3.6 Title Formatting

To begin a new content, always start the content with a chapter heading. To insert the chapter with an additional `minitoc` use `\Chapter`, which produces an heading as you see at the beginning of this page, to have a heading without a `minitoc` use the standard `\chapter`. The document relies on the user to use the **correct** title commands to keep the formatting consistent. The commands that starting with a capital letter are the overloaded commands of the standard ones. The following are the current ones:

```
\Chapter{...}  
\Section{...}  
\Subsection{...}  
\Subsubsection{...}
```

Unless there is a specific reason, it is suggested to use the aforementioned commands. However, original LaTeX command should work as well if you want to use.

3.6.1 Page Geometry

The page geometry is set to the following settings. This is done using the standard package `\usepackage{geometry}` which is defined in the `Hebdomon.cls`.

```
top 2.5cm,  
right 2.0cm,  
bottom 2.5cm,  
left 3.0cm.
```

3.7 Image Positioning

The image positioning could be done with the following code snippet:

```
\begin{figure}[ht]  
    \centering  
    \includegraphics[options]{figures/path.pdf}  
    \caption{\label{fig:label}}  
\end{figure}
```

Here there are a few options worth mentioning:

`figures/path` The place where the image is kept. If the image is in the same folder where the main.tex file resides, it is as simple as writing the files name. If the file is in a folder called image, just write `image/innsbruck.jpg`. Finally if the image is in a higher directory (i.e., image is in folderA/innsbruck.jpg and the main tex is in folderA/document/main.tex then the path becomes `../innsbruck.jpg`

`\caption{..}` Where you write the caption of the image. For consistency make sure every image has a caption as if the image does not need a caption, maybe it not be present to begin with.

`\label{}` This is an identifier for you to use when you need to cite this Figure in a place somewhere. For example if you were to have and image with the following:

```
\begin{figure}[ht]
    \centering
    \includegraphics[width=\textwidth]{figures/path.pdf}
    \caption{A photo I found on the web.}\label{fig:innsbruck}
\end{figure}
```

Now this image is referenced as `fig:innsbruck`, which means if we write the following:

To see the image, have a look at Figure `\ref{fig:innsbruck}`

This line of command will be presented as

To see the image, have a look at Figure 1.1.

Finally you can see the image here as well.



Figure 3.2: The famous Innsbruck houses near the river Inn. This image is placed with a width value of `width=\linewidth`. This is also a good opportunity to showcase the hanging behaviour of the figure caption.

3.8 Defined Environments

excerpt The template relies on the excellent `tcolorbox` package for formatting the boxes within the document and for that end different styles were created.

Sometimes one needs to quote either a proverb or to create drama, for this use the `excerpt` environment with the following notation and effect.

```
\begin{excerpt}
    To be, or not to be...
\end{excerpt}
```

Compiling this code snippet would show as in the document

To be, or not to be...

code During the preparation of your document, it is useful to showcase some code either in the shape of all the document or a snippet of it. There are two (2) ways of doing this where the first one will be discussed here.

For example to print out a hello world in python, please use the following environment

```
\begin{code}{python}
print("Hello, World!")
\end{code}
```

Producing the following:

```
print("Hello, World!")
```

The class also come with some predefined environments to modify the behaviour/aesthetics of the document. Highlighting text is **very easy**, here is an example on how to write one.

Highlighting text is `\highlight{very easy}`, here is an example:

example Sometimes you need to showcase an example or need to highlight a certain idea. For these things the environment Example could be useful.

For example to show as simple example or give a slight attention to a topic you can do the following.

This is an example. This could be anything which you would like to have a certain amount of attention but not too much as to distract from the flow of the document.

highlight Or sometimes you need to give a clear break to the flow of the document and ask the reader to look at your banner. For that use highlight.

Hey! Pay attention as this is a highlight box.

3.8.1 Writing Equations

One of the strong suits of LaTeX compared to other editors and programs is its simplicity and ease of use methods of writing equations. Consider the following equation:

$$f(x) = x^2 + 2x + 1$$

In code form this would be written as:

```
\begin{equation*}
    f(x) = x^2 + 2x + 1
\end{equation*}
```

All equations that have their newline and centre staged are mostly written in an environment where it has a `begin` and an `end`. You may have noticed the asterisks sign just after the equation. This implies the environment is **not numbered**, meaning you won't be able to reference it. This is used to limit the numbering of equations to just the essential parts in the document and not reach 3 digits by the time you are in page 8. For a numbered equation like the following

$$f(x) = x^2 + 2x + 1 \quad (3.1)$$

You only need to do:

```
\begin{equation}\label{eq:quad}
    f(x) = x^2 + 2x + 1
\end{equation}
```

where `\label{eq:quad}` is the equation reference label. You could also make matrices as well as `amsmath` is preloaded into this template.

3.8.2 Designing a Table

Finally, no template is done without someone telling you how a table should be designed. Below is a standard table And the code used to generate it:

Section	Scientific Method Step
Introduction	states your hypothesis
Methods	details how you tested your hypothesis
Results	provides raw (i.e., uninterpreted) data collected
Discussion	considers whether the data you obtained support the hypothesis

Table 3.1: A Detailed look into the scientific method.

```
\begin{table}[!ht]
\begin{NiceTabular}{rX}[rules/color=[gray]{0.9},rules/width=1pt]
```

```

\CodeBefore
\rowcolors{1}{black!5}{}{}
\rowcolors{3}{blue!5}{}{}
\Body
\toprule
\textbf{Section} & \textbf{Scientific Method Step} \\
\midrule
\textbf{Introduction} & states hypothesis \\
\textbf{Methods} & how you tested hypothesis \\
\textbf{Results} & provides raw data collected \\
\textbf{Discussion} & whether it support the hypothesis \\
\bottomrule
\end{NiceTabular}
\caption{A Detailed look into the scientific method.}
\end{table}

```

Chapter 4

Plotting your data using PGF/TikZ

Contents

4.1	Introduction	22
4.1.1	A Simple 2D Plot	23
4.1.2	Plotting 3D plots	24

4.1 Introduction

PGFplots and Tikz are powerful scripting languages allowing you to draw high-quality diagrams using only a programming language. PGFplots are generally used for plotting data from a wide variety of representations from simple 2D plots to complex 3D geometries. But wikipedia description put it best:

PGF/TikZ is a pair of languages for producing vector graphics (e.g., technical illustrations and drawings) from a geometric/algebraic description, with standard features including the drawing of points, lines, arrows, paths, circles, ellipses and polygons. PGF is a lower-level language, while TikZ is a set of higher-level macros that use PGF. The top-level PGF and TikZ commands are invoked as TeX macros, but in contrast with PSTricks, the PGF/TikZ graphics themselves are described in a language that resembles MetaPost.

For more info please look at the documentation here. It is of course up to the user to select which graphical software to produce the necessary visual components but unless it requires complex functions/processing, it would be easier to have it in PGF/TikZ format for easy editing/maintenance.

For this manual we will be looking at the three (3) plot types you may encounter in your studies.

4.1.1 A Simple 2D Plot

2D plots are simple yet powerful to show the relation of a single parameters and its related function. Below is an example of a simple comparison of two (2) functions. The image above

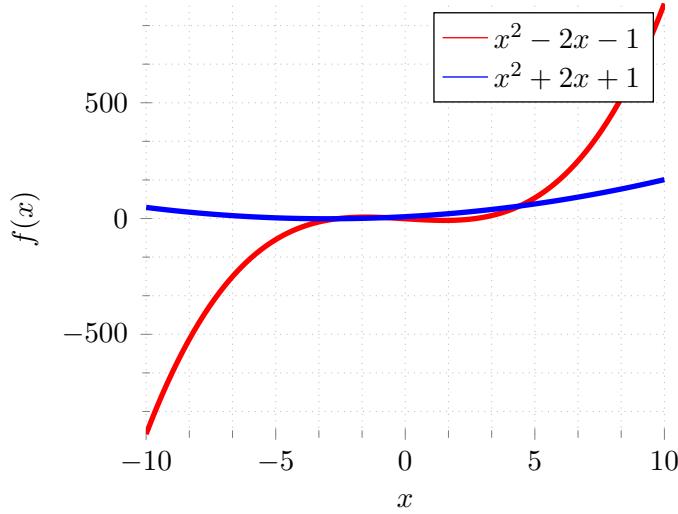


Figure 4.1: This is an example of a 2D PGF plot comparing two functions where these functions are calculated using PGF itself rather than entering/reading from data.

is generated using the following code:

```
\begin{figure}[!ht]
\centering
\begin{tikzpicture}
\begin{axis}[hebdomon, xlabel = \texttt{\$(x\$)}, ylabel = \texttt{\$\$(f(x)\$)}]
%
\addplot [domain=-10:10, samples=100,red]\texttt{\$x^3 - 7*x - 1\$};
\addlegendentry{\texttt{\$(x^2 - 2x - 1\$)}}
%
\addplot [domain=-10:10, samples=100, blue]\texttt{\$x^2 + 6*x + 8\$};
\addlegendentry{\texttt{\$(x^2 + 2x + 1\$)}}
%
\end{axis}
\end{tikzpicture}
\caption{This is an example of a 2D PGF plot comparing two functions where these functions are calculated using PGF itself rather than entering/reading from data.}
\end{figure}
```

As can be seen it is relatively standard to create plots. Some aspect which need mentioning.

\addplot You invoke this command when you want to create a plot. In the square brackets (i.e., `[]`) you insert your **configuration** of your plot. The most important ones are

`domain` the range in which the function will be calculated

`sample` the number of calculations will be done within the defined domain.

4.1.2 Plotting 3D plots

Plotting data with PGFplots is also quite possible and will generate great plot (as long as it is not massively complicated). For more information on the precautions on designing 3D plots, please have a look at here.

Below is the prototypical plot to showcase the 3D capabilities of PGF:

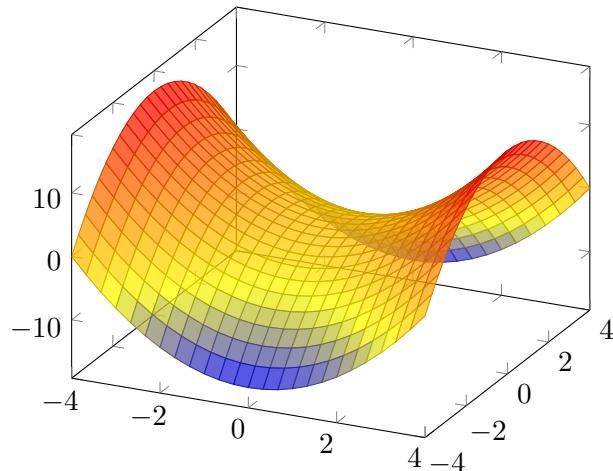


Figure 4.2: An example 3D plot done wit PGFplots.

And, of course the code for generating the plot is given as follows:

```
\begin{figure}[!ht]
\centering
\begin{tikzpicture}
\begin{axis}[view={25}{30},mark layer=like plot]
\addplot3 [
surf,
shader=faceted,
fill opacity=0.75,
samples=25,
domain=-4:4,
y domain=-4:4,
on layer=main,
] {x^2-y^2};
\end{axis}
\end{tikzpicture}
\caption{An example 3D plot done wit PGFplots.}
\end{figure}
```

Some options worth mentioning are as follows:

`surf` Generates a **surface** based on the 2D data it was given (in this case these are x and y).

`shader` Describes, basically how each segment should be filled.

`samples` Similar to 2D plots, tells how many data points will be measured. However, make a note that 3D is significantly more taxing on the TeX memory than 2D and making this sampling high may result in exceeding the memory limit.