

RELAZIONE DI PROGETTO DI "DEEP LEARNING" - UNIVERSITÀ  
DI BOLOGNA, CAMPUS DI CESENA

---

# **Generazione di radiografie e classificazione secondo la scala Kellgren-Lawrance**

---

*Componenti del gruppo:*

- Alessandro Magnani (0001026336)
- Andrea Matteucci (0001026901)
- Simone Montanari (0001026332)

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Prefazione . . . . .	4
1.2	Osteoartrite . . . . .	4
1.3	Gradi di Kellgren-Lawrence . . . . .	4
1.4	Obiettivo del progetto . . . . .	5
1.4.1	Classificazione binaria e multi-classe . . . . .	6
<b>2</b>	<b>Stato dell'arte</b>	<b>7</b>
2.1	Stato dell'arte: uso del deep learning per la classificazione di immagini mediche . . . . .	7
2.1.1	Metodologie . . . . .	7
2.1.1.1	Data Augmentation e sbilanciamento dei dataset . . . . .	7
2.1.1.2	Transfer Learning . . . . .	8
2.1.1.3	Approcci multi-view . . . . .	8
2.1.2	Modelli proposti . . . . .	8
2.2	Stato dell'arte: l'evoluzione dei Vision Trasformer per l'analisi di immagini mediche . . . . .	11
2.2.1	Progressi recenti . . . . .	11
2.2.2	Modelli proposti . . . . .	12
2.3	Stato dell'arte: uso del deep learning per la generazione di dati sintetici in medicina . . . . .	14
2.3.1	Privacy e regolamentazione . . . . .	14
2.3.2	Generazione di dati sintetici . . . . .	15
2.3.3	Studio sui DeepFake delle radiografie dell'osteoartrite del ginocchio . . . . .	15
<b>3</b>	<b>Progettazione</b>	<b>18</b>
3.1	Dataset . . . . .	18
3.1.1	Dataset I . . . . .	18
3.1.2	Dataset II . . . . .	19
3.2	Data Pre-Processing . . . . .	19
3.2.0.1	Equalizzazione . . . . .	19
3.2.1	CNN . . . . .	20
3.2.1.1	Data Augmentation . . . . .	21
3.2.1.2	Conversione e centratura sullo zero . . . . .	21
3.2.2	ViT . . . . .	22
3.2.2.1	Cropping . . . . .	22
3.2.3	WGAN-GP . . . . .	22
3.2.3.1	Rimozione di immagini non adatte . . . . .	23
3.2.3.2	Flipping orizzontale delle immagini di ginocchia destre . . . . .	24
3.2.3.3	Focus filtering . . . . .	25
3.2.3.4	Normalizzazione . . . . .	26
3.3	CNN . . . . .	27

3.3.1	ResNet50 . . . . .	27
3.3.2	Classificazione multi-classe . . . . .	28
3.3.2.1	Configurazione dei set per l'addestramento . . . . .	28
3.3.2.2	Aggiunta di pesi alle classi . . . . .	28
3.3.2.3	Configurazione del modello . . . . .	29
3.3.3	Classificazione binaria . . . . .	33
3.3.3.1	Processo di sviluppo . . . . .	33
3.4	ViT . . . . .	34
3.4.1	Architettura . . . . .	34
3.4.1.1	Suddivisione delle immagini in patch . . . . .	34
3.4.1.2	Embedding delle patch . . . . .	35
3.4.1.3	Blocco Transformer . . . . .	35
3.4.1.4	Classificazione . . . . .	36
3.4.2	Scelte progettuali . . . . .	36
3.4.2.1	Data modification . . . . .	36
3.4.2.2	Evoluzione del progetto: dal multi-classe al binario . . . . .	37
3.5	WGAN-GP . . . . .	38
3.5.1	Componenti della WGAN-GP . . . . .	41
3.5.1.1	Generatore . . . . .	41
3.5.1.2	Discriminatore . . . . .	42
3.5.2	Scelte progettuali . . . . .	42
<b>4</b>	<b>Analisi delle performance e Considerazioni</b>	<b>45</b>
4.1	CNN . . . . .	45
4.1.1	Classificazione Multi-classe . . . . .	45
4.1.1.1	Loss e accuracy . . . . .	45
4.1.1.2	Precision, Recall e F1-score . . . . .	46
4.1.1.3	Confusion matrix . . . . .	47
4.1.1.4	Grad-CAM . . . . .	48
4.1.2	Classificazione Binaria Dataset I . . . . .	48
4.1.2.1	Loss e accuracy . . . . .	48
4.1.2.2	Precision, Recall e F1-score . . . . .	49
4.1.2.3	Confusion matrix . . . . .	49
4.1.2.4	Grad-CAM . . . . .	50
4.1.3	Classificazione Binaria Dataset II . . . . .	50
4.1.3.1	Loss e accuracy . . . . .	50
4.1.3.2	Precision, Recall e F1-score . . . . .	51
4.1.3.3	Confusion matrix . . . . .	51
4.1.3.4	Grad-CAM . . . . .	52
4.2	ViT . . . . .	52
4.2.1	Classificazione Multi-classe . . . . .	52

4.2.1.1	Loss e accuracy . . . . .	53
4.2.1.2	Precision, Recall e F1-score . . . . .	54
4.2.1.3	AUC . . . . .	55
4.2.1.4	Confusion matrix . . . . .	56
4.2.1.5	Grad-CAM . . . . .	57
4.2.2	Classificazione Binaria . . . . .	57
4.2.2.1	Loss e accuracy . . . . .	58
4.2.2.2	Precision, Recall e F1-Score . . . . .	58
4.2.2.3	AUC . . . . .	59
4.2.2.4	Confusion matrix . . . . .	59
4.2.2.5	Grad-CAM . . . . .	60
4.3	WGAN-GP . . . . .	61
4.3.1	Potenziali miglioramenti e limiti . . . . .	61
4.3.2	Analisi delle loss . . . . .	63
<b>5 Conclusioni e considerazioni finali</b>		<b>64</b>
<b>Bibliografia</b>		<b>65</b>

# 1 Introduzione

## 1.1 Prefazione

La relazione descrive un programma progettato per determinare, mediante l'analisi di radiografie del ginocchio, la necessità di un intervento protesico e per generare immagini delle stesse. Questo programma fa parte di un progetto più ampio che prevede l'uso di un visore in sala operatoria, al fine di assistere il chirurgo durante le fasi dell'operazione, migliorando così l'efficacia del metodo di lavoro.

## 1.2 Osteoartrite

L'*osteoartrite* è una malattia articolare che provoca degenerazione della cartilagine e deformità ossee, causando dolore, rigidità, e gonfiore. Clinicamente, il ginocchio è la parte del corpo più colpita da questa patologia [1].

Una diagnosi precoce può rallentare la progressione della malattia, migliorando la qualità della vita e la mobilità del paziente. Nei casi avanzati, la sostituzione totale del ginocchio può significativamente migliorare la qualità della vita, alleviando il dolore e ripristinando la funzionalità articolare.

L'intelligenza artificiale, attraverso l'uso di reti neurali, può aiutare a identificare precocemente i segni di degenerazione articolare e valutare la gravità dell'osteoartrite, supportando i medici nella decisione di procedere con interventi chirurgici.

## 1.3 Gradi di Kellgren-Lawrence

Per classificare la gravità dell'osteoartrite e guidare le decisioni terapeutiche, viene comunemente utilizzata la classificazione di Kellgren-Lawrence [2]. Questo sistema assegna un punteggio da 0 a 4 per descrivere la progressione dell'osteoartrite:

- **Grado 0:** nessun segno visibile di osteoartrite;
- **Grado 1:** dubbio restringimento dello spazio articolare e piccola formazione di osteofiti (sporgenze anomale);
- **Grado 2:** formazione limitata di osteofiti e possibile restringimento dello spazio articolare;
- **Grado 3:** multiple formazioni moderate di osteofiti e restringimento visibile e limitato dello spazio articolare;
- **Grado 4:** severo restringimento dello spazio articolare, deformazione ossea visibile, ampia formazione di osteofiti.

Questa scala serve per standardizzare la valutazione dell'osteoartrite, facilitando la comunicazione tra medici e la pianificazione del trattamento, inclusa la decisione su interventi protesici basati sulla gravità della malattia.

Grade 0 No OA	Grade 1 Doubtful OA	Grade 2 Mild OA	Grade 3 Moderate OA	Grade 4 Severe OA
No Osteophites	Possible Osteophites	Definite Osteophites	Moderate Osteophites	Large Osteophites
No JSON	Doubtful JSON	Possible JSON	Definite JSON	Great JSON

Figura 1: Gradi della Classificazione Kellgren-Lawrance

## 1.4 Obiettivo del progetto

L'obiettivo principale del progetto è sviluppare un sistema basato su AI per migliorare la diagnosi e il trattamento dell'osteoartrite del ginocchio attraverso l'analisi delle radiografie. Questo progetto è stato suddiviso in tre parti principali, ognuna delle quali è stata affrontata da un membro del team:

- Vision Transformer (ViT) per la classificazione:** un membro del team si è focalizzato sulla costruzione e l'addestramento di un ViT. Questo modello ha l'obiettivo di classificare le immagini delle radiografie secondo i gradi di Kellgren-Lawrence. L'uso dei Transformer nel contesto delle immagini mediche è relativamente nuovo e promette di catturare le caratteristiche rilevanti delle immagini con una precisione elevata grazie alla sua capacità di considerare le relazioni tra le porzioni dell'immagine.
- ResNet per la classificazione:** un secondo membro ha adottato un approccio diverso, utilizzando una ResNet50 per il medesimo task. Questo approccio complementare al ViT ci permette di confrontare i punti di forza di entrambe le architetture per migliorare l'accuratezza della classificazione.
- WGANGP per la generazione di immagini:** il terzo membro del team ha lavorato sulla creazione di una GAN per generare nuove immagini di radiografie. L'obiettivo è quello di aumentare il dataset disponibile per l'addestramento dei modelli ViT e ResNet. Le GAN permettono di espandere artificialmente il numero di campioni disponibili senza dover raccogliere ulteriori dati clinici, compito quanto più complesso in ambito medico viste le numerose restrizioni legali legate alla privacy dei pazienti.

#### 1.4.1 Classificazione binaria e multi-classe

Oltre alla classificazione a 5 classi basata sui gradi KL, si è deciso di implementare anche una classificazione binaria. In questa modalità, i gradi 0 e 1 sono considerati come *ginocchio sano*, mentre i gradi 2, 3 e 4 sono classificati come *ginocchio con necessità di protesi*. Passare da un problema multi-classe a uno binario semplifica la complessità del task, aiutando i modelli a migliorare le performance.

L'integrazione di questi tre approcci ci permette di sviluppare un sistema che non solo classifica accuratamente le radiografie, ma che migliora continuamente attraverso la generazione di nuovi dati sintetici. Questo sistema ha il potenziale di supportare i medici nel prendere decisioni, migliorando la diagnosi e personalizzando i trattamenti per i pazienti.

## 2 Stato dell'arte

### 2.1 Stato dell'arte: uso del deep learning per la classificazione di immagini mediche

Negli ultimi anni, l'adozione di tecniche di deep learning ha portato a significativi progressi nella classificazione delle immagini mediche [2].

#### 2.1.1 Metodologie

Le principali metodologie utilizzate includono la data augmentation, il transfer learning e gli approcci multi-view.

##### 2.1.1.1 Data Augmentation e sbilanciamento dei dataset

I modelli di deep learning necessitano di grandi dataset ben etichettati per raggiungere buone performance. Uno dei principali problemi nell'analisi delle immagini mediche è lo sbilanciamento dei dataset, aggravato dalle restrizioni sulla privacy, dalla scarsità di dati e dalle etichette spesso non accurate [3].

Le tecniche di data augmentation sono estremamente utilizzate per colmare queste lacune. Queste infatti aumentano la variabilità dei dati, migliorano la generalizzazione del modello e bilanciano le classi sottorappresentate.

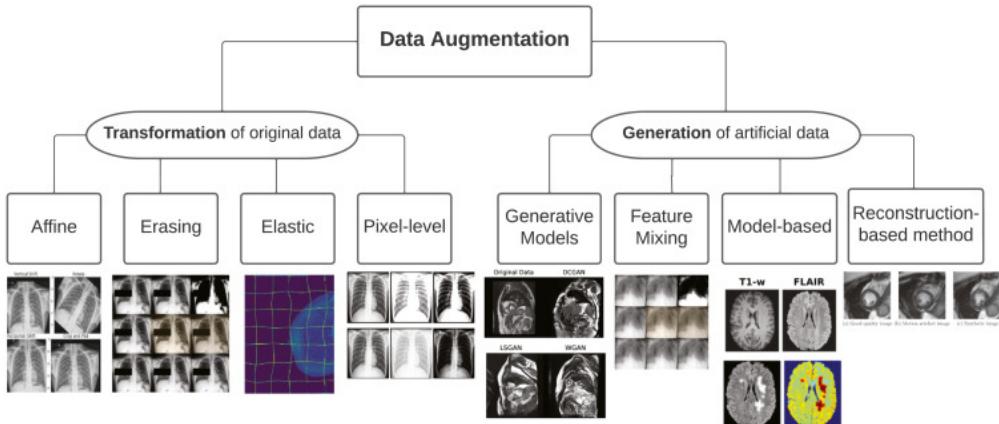


Figura 2: Tassonomia delle tecniche di Data Augmentation [3].

Sono stati pubblicati molti studi che valutano l'efficacia di queste tecniche. In particolare, gli autori di questo paper [3] hanno analizzato più di 300 articoli pubblicati tra il 2018 e il 2022 riguardanti la data augmentation nel dominio medico, riscontrando un miglioramento delle prestazioni sotto tutti i punti di vista.

### 2.1.1.2 Transfer Learning

A causa della limitata disponibilità di grandi dataset etichettati in campo medico, il *Transfer Learning* è ormai diventato una prassi consolidata. Questo approccio sfrutta modelli pre-addestrati come punto di partenza, per poi adattarli e raffinarli su dataset specifici mediante *Fine Tuning*, ottimizzando così le prestazioni.

È stato dimostrato che l'utilizzo del Transfer Learning da modelli pre-addestrati su grandi dataset come *ImageNet* per risolvere task in ambito medico risulta essere estremamente vantaggioso [4].

### 2.1.1.3 Approcci multi-view

Gli approcci multi-view [5] nel deep learning in ambito medico combinano immagini da diverse prospettive o modalità. L'obiettivo di queste tecniche è quello di integrare ed elaborare tali informazioni *eterogenee* al fine di ottenere una rappresentazione più accurata.

Soltanamente queste tecniche utilizzano più reti neurali per elaborare le diverse prospettive.

## 2.1.2 Modelli proposti

In letteratura sono stati presentati numerosi modelli per la classificazione di radiografie del ginocchio secondo la scala di Kellgren-Lawrence [2]. Di seguito sono riportati alcuni dei lavori più rilevanti, distinguendo tra quelli che hanno utilizzato lo stesso dataset del progetto e quelli che hanno impiegato dataset differenti.

Articoli che hanno sfruttato lo stesso dataset (3.1.1):

- **A Convolution Neural Network Design for Knee Osteoarthritis Diagnosis Using X-ray Images [6]**

Gli autori di questo paper hanno proposto una CNN personalizzata per la classificazione multi-classe.

Il modello ha raggiunto un'accuratezza complessiva del 98% con un tempo di elaborazione di 1 ora 47m. Come illustrato nell'immagine 3, il modello ha dimostrato prestazioni eccellenti in tutti i gradi della scala KL.

Model Name	Class Name	Precision	Recall	F1 Score	Processing Time	Accuracy
InceptionResNetV2	Minimal	0.30	0.32	0.30	2h 37 m	0.68
	Healthy	0.23	0.29	0.25		
	Moderate	0.26	0.30	0.28		
	Doubtful	0.76	0.55	0.64		
	Severe	1.00	0.60	0.75		
Xception	Minimal	0.93	0.73	0.82	2 h 18m	0.91
	Healthy	0.96	0.85	0.90		
	Moderate	0.64	0.90	0.75		
	Doubtful	0.81	0.90	0.85		
	Severe	0.95	0.96	0.95		
Proposed CNN Model	Minimal	0.97	0.98	0.99	1h 47m	0.98
	Healthy	0.99	0.92	0.95		
	Moderate	0.94	0.98	0.96		
	Doubtful	1.00	0.99	0.99		
	Severe	1.00	1.00	1.00		

Figura 3: Comparazione dei modelli addestrati nell'articolo [6].

Inoltre, gli autori dello studio hanno riportato una tabella comparativa che riassume vari approcci alla classificazione delle immagini di radiografie e risonanze magnetiche del ginocchio.

Reference No./Year of Publishing	Dataset Used	Technique	Results
[34]/ 2010	X-ray images	SDM-NN and SDM-SVM	78.8% accuracy with SDM-NN and 85.4% accuracy with SDM-SVM
[35]/ 2016	X-ray images	Random Forest Classifier	An accuracy of about 87.92 % is achieved.
[36]/ 2019	X-ray images	Proposed Deep neural networks	79.39% accuracy
[21]/ 2019	MRI images	Deep learning-based convolutional neural network (DenseNet)	75% accuracy
[37]/ 2020	X-ray images	R-CNN	74.3% accuracy 93.6% sensitivity 74.2% specificity
[38]/ 2021	MRI images	CNN	83% accuracy
<b>Proposed Model</b>	<b>X-ray images</b>	<b>CNN</b>	<b>98% accuracy</b>

Figura 4: Comparazione con lo stato dell'arte [6].

- **Knee Osteoarthritis Detection and Severity Classification Using Residual Neural Networks on Preprocessed X-ray Images [7]**

In questo articolo sono state utilizzate reti neurali **residue** per la classificazione binaria e multi-classe.

Questo articolo è stato di grande ispirazione per quanto riguarda le operazioni di data pre-processing che sono state seguite nello sviluppo del progetto. In particolare, le tecniche di *equalizzazione* e *cropping* descritte nello studio sono state utilizzate nell'elaborato.

Di seguito sono riportati i risultati dell'articolo per entrambe le classificazioni.

Results of different classifiers on Dataset I containing 5 classes.

Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Testing Accuracy	Precision	Recall	F1 Score
MobileNetV2	0.9951	0.0409	0.5363	2.0755	0.67	0.69	0.69	0.67
ResNet101	0.9504	0.1475	0.5484	1.8031	0.69	0.67	0.67	0.65
VGG16	0.8228	0.4589	0.5605	1.3003	0.66	0.66	0.64	0.66
VGG19	0.7045	0.6873	0.5424	1.2559	0.64	0.59	0.64	0.58
InceptionResNetV2	0.6992	0.7659	0.5291	1.3470	0.63	0.64	0.63	0.60
DenseNet121	0.7186	0.6854	0.5751	1.1337	0.64	0.65	0.64	0.59

Figura 5: Risultati della classificazione multi-classe [7].

Results of different classifiers on Dataset II containing 2 classes.

Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Testing Accuracy	Precision	Recall	F1 Score
MobileNetV2	0.8594	0.3225	0.7869	0.5358	0.81	0.83	0.82	0.81
ResNet101	0.9825	0.2060	0.7966	0.7018	0.83	0.83	0.81	0.81
VGG16	0.9396	0.0334	0.7851	1.5386	0.82	0.83	0.83	0.83
VGG19	0.8515	0.3317	0.7809	0.5231	0.81	0.83	0.82	0.81
InceptionResNetV2	0.9043	0.2312	0.7833	0.6096	0.81	0.82	0.81	0.81
DenseNet121	0.8949	0.2419	0.8087	0.5264	0.82	0.83	0.82	0.82

Figura 6: Risultati della classificazione binaria [6].

Articoli che hanno sfruttato dataset differenti:

- **Automatic detection and classification of knee osteoarthritis using deep learning approach [8]:** è stato effettuato *Transfer Learning* su una rete *AlexNet* pre-addestrata raggiungendo una precisione del 98.90%. È doveroso sottolineare che il set di dati utilizzato

in questo studio conteneva solamente poco più di 3.000 immagini, dimostrando l'efficacia del modello anche con una quantità limitata di dati.

- **Grading of Knee Osteoarthritis Using Convolutional Neural Networks** [9]: questo studio introduce un modello chiamato **MCBCNN** (Multiscale Convolutional Blocks in Convolutional Neural Network), composto da filtri convoluzionali multiscala e tre diverse reti pre-addestrate. La precisione riscontrata è pari al 95%;

## 2.2 Stato dell'arte: l'evoluzione dei Vision Trasformer per l'analisi di immagini mediche

Le CNN sono state a lungo la scelta obbligata per l'elaborazione di immagini mediche. Nonostante le prime applicazioni risalgano agli anni '90, è solo nell'ultimo decennio che le CNN hanno veramente dominato il campo, con architetture come U-Net che hanno stabilito nuovi benchmark in molte aree dell'imaging medico.

Tuttavia, come citato nel paper [10] il panorama sta cambiando rapidamente con l'introduzione dei Vision Transformers. Originariamente sviluppati per l'elaborazione del linguaggio naturale, i Transformers stanno ora dimostrando un grande potenziale nell'analisi delle immagini mediche. Offrono vantaggi significativi, come una maggiore capacità di comprendere il contesto globale dell'immagine che è cruciale nella diagnosi medica dove è importante considerare non solo l'area di interesse ma anche i tessuti e gli organi circostanti.

Nonostante questi vantaggi, i ViT presentano anche delle sfide. Queste architetture infatti richiedono generalmente più dati per l'addestramento, il che può essere problematico nel settore medico dove le risorse e i dati possono essere limitati per questioni di privacy dei pazienti.

### 2.2.1 Progressi recenti

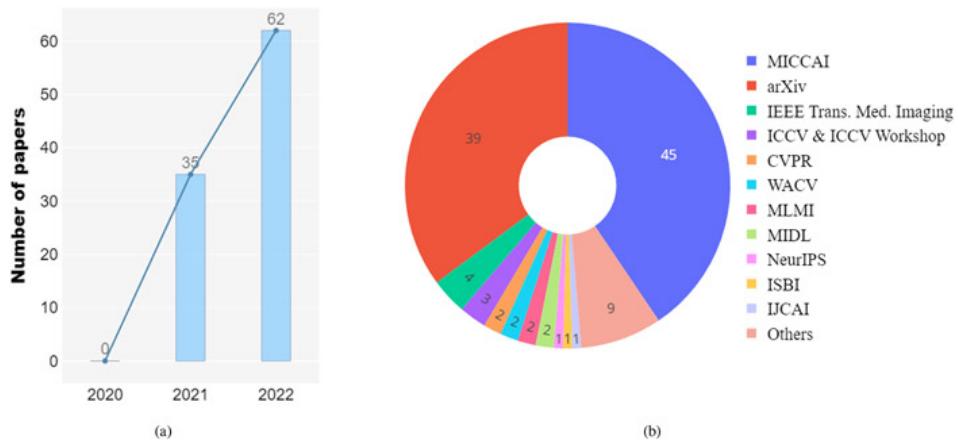


Figura 7: Progressi recenti riguardanti i Vision Transformers

L'interesse per i Vision Transformers nel campo dell'analisi di immagini mediche ha visto una crescita esponenziale negli ultimi anni. Come evidenziato in figura 7, c'è stato un aumento significativo nel numero di pubblicazioni, passando da zero a oltre 60 in due anni.

La ricerca copre un'ampia gamma di applicazioni, tra cui segmentazione di immagini mediche, riconoscimento e classificazione, rilevamento, registrazione, ricostruzione e miglioramento. Questa diversità di applicazioni sottolinea la versatilità e il potenziale dei Vision Transformers nel campo medico.

È importante notare che, nonostante la crescita rapida, questo campo è ancora relativamente nuovo. Ciò suggerisce che si è all'inizio di una tendenza emergente, con ampio spazio per ulteriori innovazioni nel prossimo futuro.

### 2.2.2 Modelli proposti

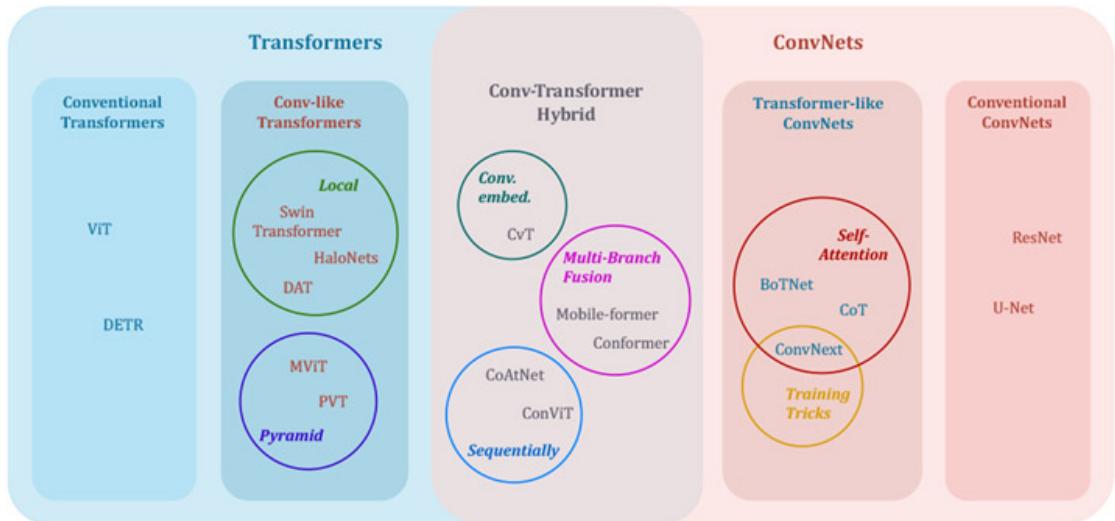


Figura 8: Diverse tipologie e combinazioni di approcci Trasformer-CNN

Nel campo dell'elaborazione delle immagini, più recentemente, si sta assistendo a una convergenza interessante tra le CNN e i ViT. Questa fusione ha dato vita a tre principali categorie di modelli ibridi. In particolare, siccome non sono stati trovati paper significativi per questo task specifico riguardante ViT, si è scelto di approfondire i risultati delle architetture Transformer Convolutional-like.

Questi modelli rappresentano un tentativo di infondere nei Transformer alcune delle caratteristiche vantaggiose delle CNN. L'obiettivo principale è mantenere la capacità dei Transformer di catturare relazioni a lungo raggio, introducendo al contempo concetti tipici delle CNN come la località e la gerarchia.

Un esempio concreto di applicazione di questa categoria di modelli ibridi è rappresentato da un recente studio sulla diagnosi dei gradi dell’osteoartrite di ginocchio usando uno **Swin Transformer**, riportato nel paper [11]. Il metodo proposto sfrutta le capacità del Swin Transformer come estrattore di caratteristiche, combinandolo con una rete *multi-prediction head* per migliorare la classificazione.

Il modello proposto nel paper viene considerato come riferimento per lo stato dell’arte in termini di accuratezza (70.17%) e F1-score (67%) per la classificazione a 5 classi. In particolare, questa architettura ha dimostrato un miglioramento significativo nella rilevazione del grado ”1”, che è notoriamente difficile da distinguere dai gradi ”0” e ”2” a causa della loro alta somiglianza.

<b>Method</b>	<b>Acc (%) ↑</b>	<b>F1 ↑</b>
Antony et al. 2016 [2]	53.40	0.43
Antony et al. 2017 [1]	63.60	0.59
Ordinal Loss (Vgg19) [5]	69.60	-
Ordinal Loss (ResNet50) [5]	66.20	-
Ordinal Loss (ResNet101) [5]	65.50	-
Siamese net [16]	66.71	-
Wang et al. [19]	69.18	-
<b>Ours experiment 5</b>	<b>70.17</b>	<b>0.67</b>

**Table 4: Results for OAI dataset.**

Figura 9: Valori di accuratezza e F1-Score calcolati nel paper [11]

Le confusion matrix fornite nel paper illustrano chiaramente questo miglioramento. Nell’esperimento con la *multi-prediction head*, il modello è riuscito a classificare correttamente 54 immagini di grado ”1”, mentre nell’esperimento con un singolo classificatore, nessuna immagine di grado ”1” era stata classificata correttamente.

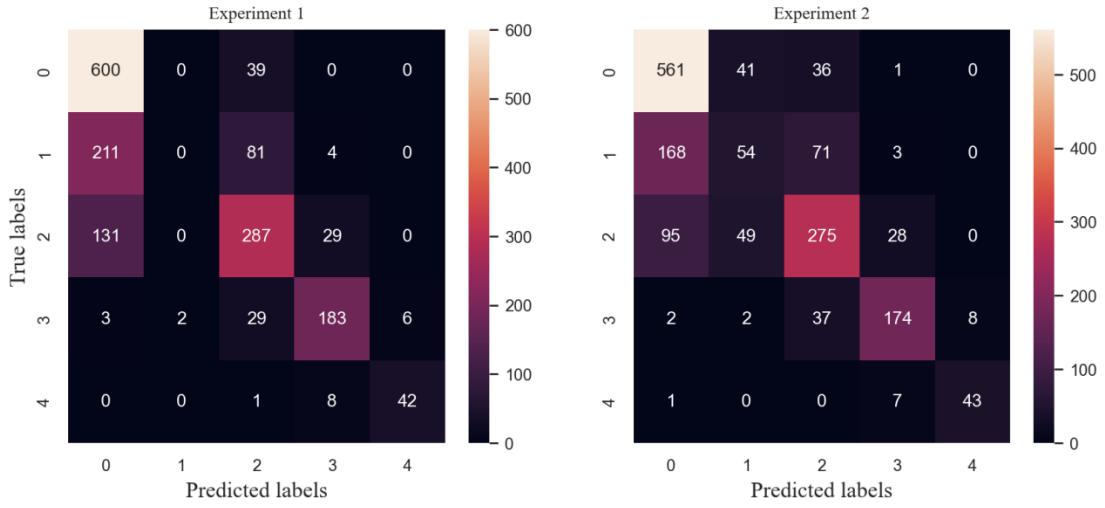


Figura 10: Predizioni ottenute nel paper [11]

Nel paper è stata proposta anche una valutazione qualitativa usando la visualizzazione Grad-CAM che dimostra come il modello si concentri su regioni clinicamente rilevanti, confermando la sua capacità di apprendere le caratteristiche significative.

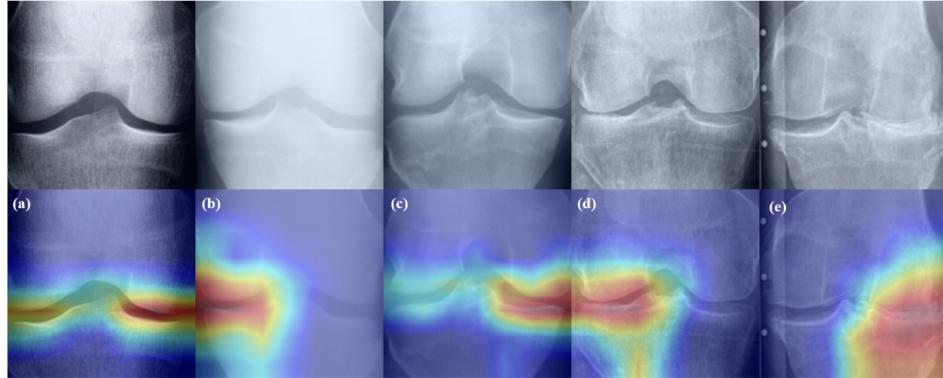


Figura 11: Risultati ottenuti nel paper [11] mediante visualizzazione Grad-CAM

## 2.3 Stato dell'arte: uso del deep learning per la generazione di dati sintetici in medicina

### 2.3.1 Privacy e regolamentazione

Le normative sulla privacy e sull'uso dei dati medici vengono costantemente aggiornate a livello globale. Negli Stati Uniti, l'HIPAA e il GDPR dell'Unione Europea sono progettati per limitare il flusso di dati e garantire il consenso dei pazienti per la loro diffusione.

Inoltre lo scambio di dati sanitari tra continenti aggiunge ulteriori complicazioni, rendendo praticamente impossibile condividere dati di ricerca senza una preparazione preliminare e accordi formali.

Questi regolamenti complicano il lavoro dei gruppi di ricerca, proprio perché le applicazioni di deep learning necessitano di grandi dataset open-source e di contributi disponibili pubblicamente per continuare a migliorare.

### 2.3.2 Generazione di dati sintetici

La generazione di dati può risolvere i problemi di privacy. I dati generati infatti possono essere condivisi liberamente, rispettando le normative e migliorando le applicazioni di deep learning senza compromettere la privacy dei pazienti. Questi dati possono essere combinati con dati reali in un approccio di aumento additivo, che ha dimostrato di migliorare le prestazioni delle reti.

Per poter generare dati sintetici vengono usate le reti GAN. In alcuni casi le GAN sostituiscono completamente i dati reali, mentre in altre situazioni li integrano aumentando la dimensione del dataset. Tuttavia, gli effetti dell'aumento variano tra i contesti medici e solo pochi sistemi sono stati validati da esperti.

### 2.3.3 Studio sui DeepFake delle radiografie dell'osteoartrite del ginocchio

Per quanto riguarda la generazione di immagini, esistono studi di GAN in grado di produrre un numero illimitato di radiografie di ginocchio a diversi gradi KL come quello del paper [12], preso come riferimento dello stato dell'arte in quanto è risultato essere il progetto con i risultati migliori. La GAN progettata dal paper è una WGAN che consiste di un generatore e discriminatore da 6 milioni di parametri ciascuno. L'addestramento è stato svolto su 1000 epoche. I risultati sono stati validati con esperti medici e le immagini generate sono state testate per l'aumento additivo nel deep learning.

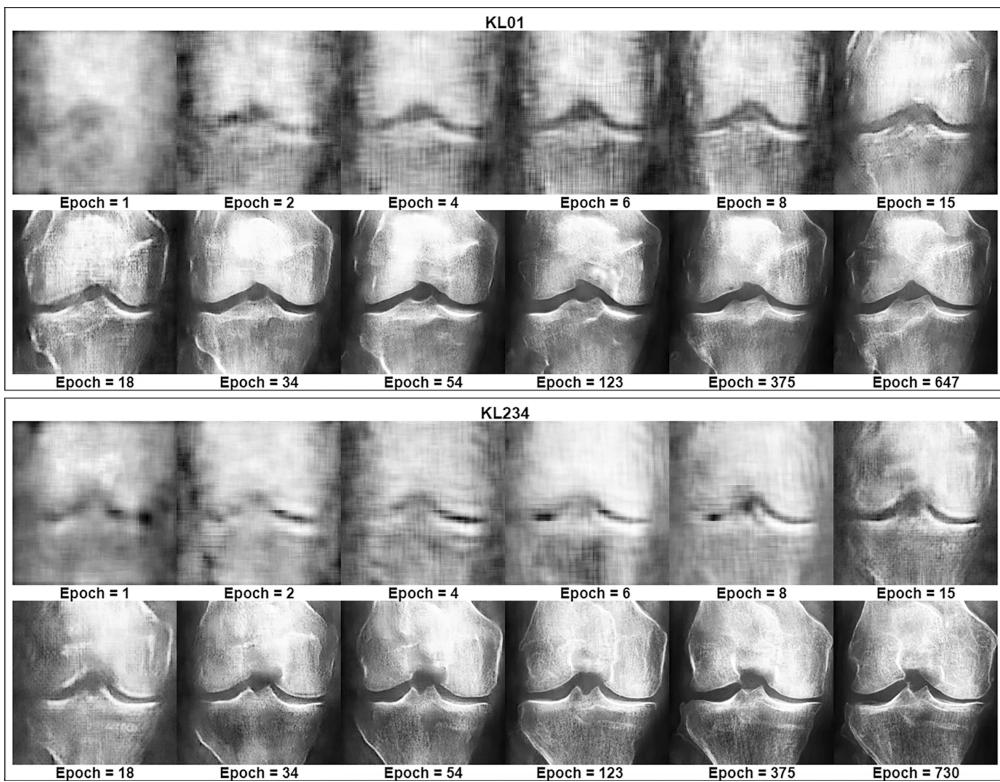


Figura 12: Immagini generate nel corso delle epoche.

I risultati sono stati distribuiti a 10 radiologi e 5 chirurghi ortopedici. Come si può notare dalla tabella sotto riportata, le immagini generate venivano confuse con quelle reali in molti casi persino dagli esperti [12]. Inoltre è stata scelta la FID come metrica di valutazione anche se non ne sono stati riportati i risultati.

Medical experts	Accuracy	Precision (%)	F1 score (%)
Orthopedic surgeons ( $n = 5$ )	65.25% ( $\pm 6.95\%$ )	65.81	65.09
Radiologists ( $n = 10$ )	59.40% ( $\pm 12.01\%$ )	59.95	58.96
All	61.35% ( $\pm 10.71\%$ )	61.91	61

Figura 13: Accuratezza, precisione e F1 score degli esperti medici nel classificare le immagini come reali o DeepFake [12]. Si può facilmente notare come gli esperti si siano sbagliati diverse volte nel riconoscere quali fossero reali e quali fake.

Nel paper è stato studiato il potenziale dei dati generati per l'aumento additivo. È stato ideato un esperimento di transfer learning per classificare tra le classi KL01 e KL234 usando una va-

riante dell'architettura VGG16 pre-addestrata con ImageNet.

I dataset di dati reali sono stati incrementati del 50%, 100%, 150% e 200% con immagini DeepFake. Infine, i dati reali sono stati completamente sostituiti con dati DeepFake.

<b>Dataset</b>	<b>Testing accuracy (@Best validation loss) (%)</b>	<b>Testing loss (@Best validation loss)</b>	<b>Validation accuracy (Best) (%)</b>	<b>Validation loss (Best)</b>
Real	71.21	4.142	80.3	4.07
Real +50% Fakes	73.48	3.819	81.06	3.809
Real +100% Fakes	72.73	3.404	81.06	3.428
Real +150% Fakes	73.48	3.205	78.03	3.295
Real +200% Fakes	75.76	2.833	78.79	2.925
Replace Real 100%	67.42	4.280	78.45	4.301

Figura 14: Punteggi di accuracy e loss per ciascun dataset utilizzato per l'aumento dei dati. Si può osservare che tutti i punteggi di accuracy e loss sono risultati migliori nei set di dati aumentati. L'effetto di aumento più significativo è stato ottenuto con un +200% di dati fake, raggiungendo un punteggio di test del 75,76%

## 3 Progettazione

### 3.1 Dataset

Per lo svolgimento dell'elaborato sono stati utilizzati due diversi *dataset*.

#### 3.1.1 Dataset I

Il primo dataset utilizzato (Knee Osteoarthritis Dataset with Severity Grading), disponibile su *Kaggle*, contiene 9786 immagini di radiografie al ginocchio divise secondo la scala KL.



Figura 15: Esempio di immagini per ogni grado KL.

Il dataset era inizialmente diviso in quattro directory: `train`, `val`, `test` e `auto_test`, quest'ultima contenente una versione trasformata delle immagini di test. All'interno di ogni directory, le immagini sono suddivise in sottodirectory numerate da 0 a 4, corrispondenti ai cinque gradi della scala.

Cardinalità totale della cartella <code>train</code> : 5778	Cardinalità totale della cartella <code>val</code> : 826
Immagini di grado 0: 2286	Immagini di grado 0: 328
Immagini di grado 1: 1046	Immagini di grado 1: 153
Immagini di grado 2: 1516	Immagini di grado 2: 212
Immagini di grado 3: 757	Immagini di grado 3: 106
Immagini di grado 4: 173	Immagini di grado 4: 27
Cardinalità totale della cartella <code>test</code> : 1656	Cardinalità totale della cartella <code>auto-test</code> : 1526
Immagini di grado 0: 639	Immagini di grado 0: 604
Immagini di grado 1: 296	Immagini di grado 1: 275
Immagini di grado 2: 447	Immagini di grado 2: 403
Immagini di grado 3: 223	Immagini di grado 3: 200
Immagini di grado 4: 51	Immagini di grado 4: 44
Cardinalità totale del dataset: 9786	
Immagini di grado 0: 3857	
Immagini di grado 1: 1770	
Immagini di grado 2: 2578	
Immagini di grado 3: 1286	
Immagini di grado 4: 295	

Figura 16: Cardinalità del dataset.

È stata presa la decisione di **eliminare** la directory `auto_test` (contenente 1531 immagini come si nota in figura 16) per evitare duplicati delle immagini di test.

Il problema dello sbilanciamento del dataset emerge chiaramente in figura 16, dove si evidenzia una netta inferiorità delle immagini di grado 3 e 4 rispetto alle immagini appartenenti agli altri gradi.

### 3.1.2 Dataset II

Il secondo dataset utilizzato (Synthetic (DeepFake) Knee Osteoarthritis X-ray Images from Generative Adversarial Neural Networks), disponibile sul sito *Mendeley Data*, contiene 320000 immagini generate dalla GAN riportata nel paper [13].

Il dataset risulta diviso in due classi:

- **Negative:** contenente le immagini generate di grado KL 0 e 1.
- **Positive:** contenente le immagini generate di grado KL 2, 3 e 4;

## 3.2 Data Pre-Processing

Durante la fase di *data pre-processing* sono state eseguite alcune operazioni comuni a tutti e tre i membri del team, mentre altre operazioni sono state utilizzate solo per uno dei tre task specifici. Quelle in comune sono:

- **Divisione in training, validation e test set;**
- **Equalizzazione.**

### 3.2.0.1 Equalizzazione

L'equalizzazione dell'istogramma è una tecnica di *data pre-processing* utilizzata per **migliorare il contrasto**. Le immagini delle radiografie nei dataset presentavano alcune limitazioni intrinseche come basso contrasto, dettagli poco definiti e luminosità mal bilanciata. L'applicazione dell'equalizzazione ha evidenziato con maggior chiarezza i dettagli anatomici e l'eventuale *osteoartrite*.

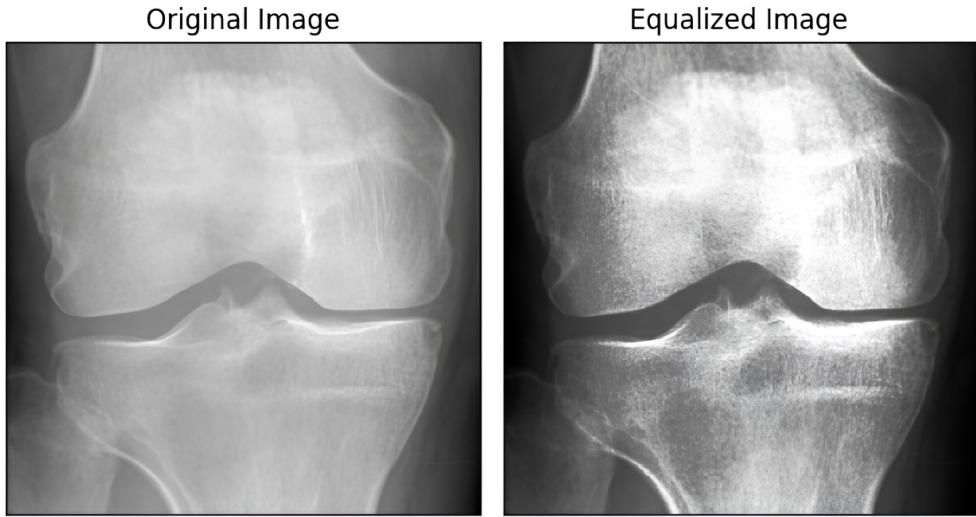


Figura 17: Effetto visivo dell'equalizzazione.

L'equalizzazione è stata applicata sfruttando la funzione `equalizeHist()` della libreria *OpenCV*, dopo aver convertito le immagini in scala di grigi.

```

1 # Lettura dell'immagine in grayscale.
2 image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
3
4 # Equalizzazione.
5 equalized_image = cv2.equalizeHist(image)

```

Listing 1: Applicazione dell'equalizzazione

La scelta di utilizzare l'equalizzazione è stata ispirata dal paper *Knee Osteoarthritis Detection and Severity Classification Using Residual Neural Networks on Preprocessed X-ray Images* [7], che ha dimostrato la sua efficacia nel migliorare la qualità delle immagini.

### 3.2.1 CNN

Oltre alle tecniche di *data pre-processing* sopracitate, per quanto riguarda nello specifico lo sviluppo della CNN (*ResNet50*) sono state applicate ulteriori operazioni. Questo l'elenco completo:

- Divisione in training, validation e test set
- Equalizzazione;
- **Data Augmentation;**
- **Conversione e centratura sullo zero.**

### 3.2.1.1 Data Augmentation

La *Data Augmentation* è stata usata per aumentare la cardinalità dei dataset, generando nuove immagini *trasformate* a partire da quelle originali.

In particolare, le operazioni di *Data Augmentation* eseguite includono il **flipping orizzontale**, la **regolazione della luminosità**, la **traslazione orizzontale**, l'**ingradimento** e il **riempimento degli spazi vuoti**.

Queste tecniche si sono dimostrate utili per mitigare il problema dell'*overfitting*, ampiamente riscontrato nella classificazione multi-classe. Nei notebook, è stata impiegata la classe `ImageDataGenerator` di Keras per applicare le operazioni di *data augmentation*.

```
1 # Creazione dell'oggetto ImageDataGenerator.  
2 aug_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
3     preprocessing_function=tf.keras.applications.resnet50.preprocess_input,  
4     horizontal_flip=True,  
5     brightness_range=[0.3, 0.8],  
6     width_shift_range=[-50, 0, 50, 30, -30],  
7     zoom_range=0.1,  
8     fill_mode="nearest",  
9 )  
10  
11 # Creazione del generatore di dati per il training.  
12 train_generator = aug_datagen.flow_from_directory(  
13     train_path, class_mode="categorical", target_size=target_size, shuffle=True  
14 )
```

Listing 2: Operazioni di Data Augmentation.

### 3.2.1.2 Conversione e centratura sullo zero

Le immagini dei dataset vengono elaborate sfruttando la funzione `preprocess_input()` di Keras per ottimizzare i dati di input per il modello *ResNet50*.

Questa funzione svolge due operazioni principali:

- **Conversione da RGB a BGR:** le immagini vengono convertite nel formato BGR poichè *ResNet50*, come molti altri modelli, è stata originariamente addestrata su immagini BGR;
- **Centratura sullo zero:** ogni canale colore viene centrato sullo zero sottraendo i valori medi del dataset *ImageNet*. Questo processo di normalizzazione è mirato a migliorare la convergenza e la stabilità.

Come si può osservare nel codice 2, la funzione `preprocess_input()` è stata integrata all'interno dell'oggetto `ImageDataGenerator` come prima operazione.

### 3.2.2 ViT

Per la creazione del modello ViT è stata fatta anche un operazione di cropping.

Riassumendo, le operazioni di pre-processing svolte per l'addestramento di ViT sono state:

- Divisione in training, validation e test set
- **Conversione da grayscale a RGB**
- **Cropping**
- Equalizzazione (solo nella classificazione a cinque classi)

#### 3.2.2.1 Cropping

Dopo essere state caricate, separate e convertite in RGB, le immagini vengono ritagliate per rimuovere eventuali bordi o regioni non necessarie. Questo approccio è stato fatto per eliminare quelle zone dell'immagine che non sarebbero servite alla rete per fare una predizione accurata, come visto nel paper [7]. Così facendo, vengono ridotti i tempi di addestramento senza che venga intaccata la qualità della classificazione. Nel codice di classificazione binaria, le immagini avevano una dimensione originale di 299x299 pixel e vengono ritagliate rimuovendo 45 pixel dalla parte superiore e inferiore, in modo da evidenziare la zona significativa dell'immagine. Nel codice di classificazione multi-classe, invece, le immagini avevano una dimensione originale di 224x224 pixel e vengono ritagliate rimuovendo 57 pixel dalla parte superiore e inferiore.

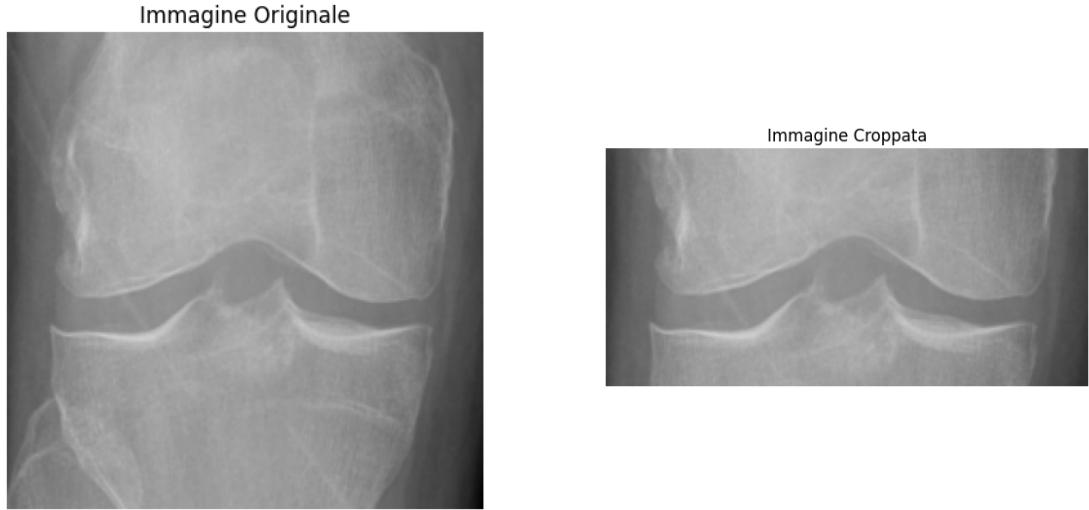


Figura 18: Differenza visiva tra un'immagine originale e la stessa immagine dopo essere stata ritagliata

### 3.2.3 WGAN-GP

Le operazioni di pre-processing svolte per l'addestramento della WGAN-GP sono state:

- Divisione in training, validation e test set;
- **Rimozione di immagini non adatte**
- **Flipping orizzontale delle immagini di ginocchia destre;**
- Equalizzazione;
- **Focus filtering;**
- **Normalizzazione;**

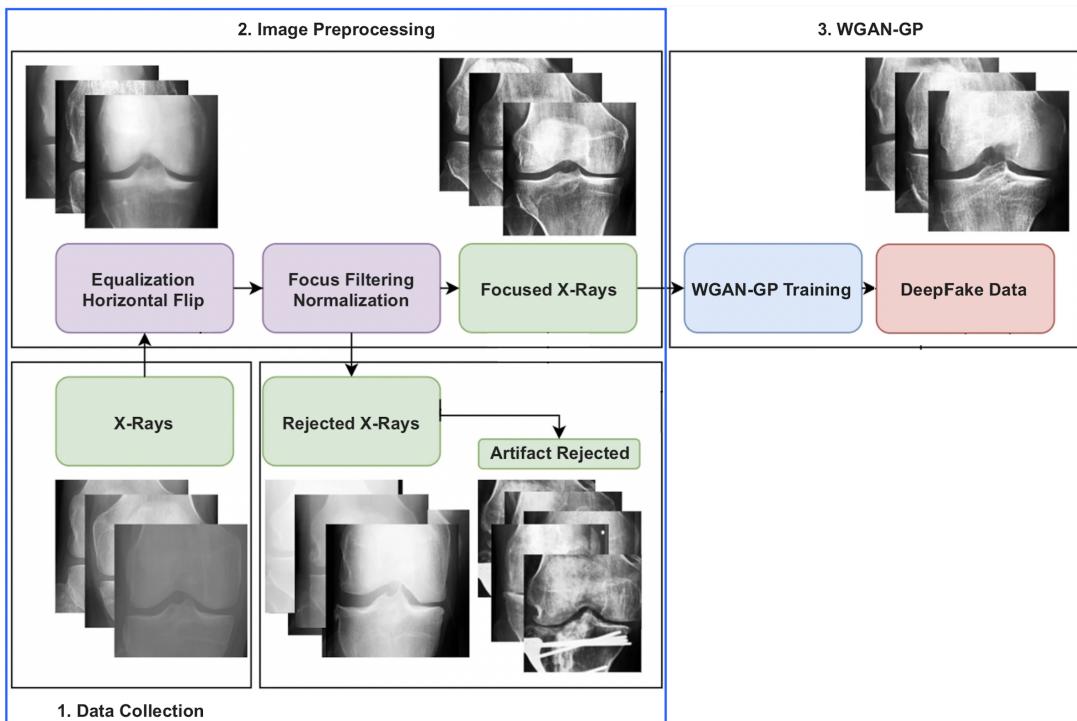


Figura 19: Flow diagram delle attività e dei dati coinvolti in questo studio. I dati reali sono evidenziati in verde; i dati DeepFake sono in rosso. Il colore viola indica le operazioni di elaborazione dei dati, mentre il blu rappresenta le procedure relative alla generazione. Le intestazioni dei blocchi contengono numeri in ordine crescente per indicare l'ordine delle operazioni (da 1 a 3).

### 3.2.3.1 Rimozione di immagini non adatte

Sono state rimosse le immagini che avevano delle protesi e altre distorsioni visibili (strappi, graffi o imperfezioni nella radiografia) per minimizzare potenziali interferenze durante l'addestramento. Inoltre sono state rimosse anche le immagini che per qualche motivo avevano i canali invertiti (ultimo esempio di figura 20).



Figura 20: Esempi di immagini rimosse.

### 3.2.3.2 Flipping orizzontale delle immagini di ginocchia destre

Le radiografie delle ginocchia destre vengono specchiamente orizzontalmente. Le ginocchia destre e sinistre non sono perfettamente identiche: presentano alcune lievi differenze che permettono di distinguere una dall'altra. Tuttavia, sono quasi speculari tra loro. Ciò significa che quando una radiografia di un ginocchio destro viene specchiata, essa appare molto simile a una radiografia di un ginocchio sinistro. Di conseguenza, se da una radiografia si può riconoscere se il ginocchio è destro o sinistro, un ginocchio destro specchiato sembrerà un ginocchio sinistro.

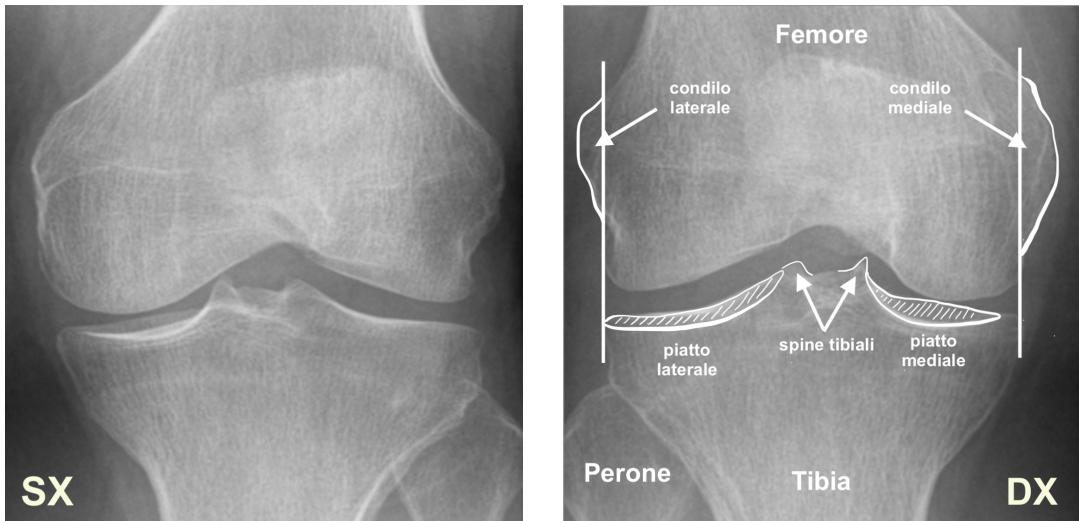


Figura 21: Radiografie di un ginocchio sinistro e destro. Si possono notare alcune differenze che rendono possibile distinguere quale è quello destro e quale quello sinistro. Si può capire dalla posizione di tibia e perone (se nella radiografia il perone è a sinistra della tibia, è un ginocchio destro), o dalla dimensione dei condili (quello laterale è più piccolo rispetto al condilo mediale, e se quello laterale è a sinistra, si tratta di un ginocchio destro), o ancora dalle spine tibiali (se nella radiografia tendono a sporgere verso destra, si tratta di un ginocchio destro).

Allenare la rete a generare immagini di ginocchia di un solo tipo riduce notevolmente la complessità del task, permettendo di ottenere immagini di qualità superiore. Questo perché tutte le immagini sono orientate nello stesso verso, anziché essere una combinazione di immagini di ginocchia destre e sinistre.



Figura 22: Radiografie di un ginocchio sinistro e di un ginocchio destro specchiato orizzontalmente. Ovviamente, seppure si tratta di ginocchia dello stesso paziente, le due immagini non sono perfettamente uguali. Nonostante questo, è chiaro che siano nettamente più simili rispetto alle due radiografie mostrate in precedenza.

### 3.2.3.3 Focus filtering

Successivamente è stato fatto il focus filtering poiché si è osservato che ampie differenze nella messa a fuoco e nella chiarezza della texture delle radiografie avrebbero confuso il generatore.

Per farlo, sono stati eseguiti i seguenti passaggi:

1. **Convoluzione con kernel laplaciano:** è stato usato per effettuare il rilevamento dei bordi sulle immagini. Evidenzia le aree di rapido cambiamento di intensità, catturando efficacemente i bordi e i dettagli. Il kernel applicato è:

$$\text{laplace\_kernel} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figura 23: Kernel laplaciano utilizzato.

2. **Calcolo della varianza:** per ciascuna immagine, è stata calcolata la varianza dei valori assoluti. Misura la quantità di dettaglio in ogni immagine. Le immagini con varianza maggiore sono considerate più a fuoco e quindi più utili per l'addestramento.

```
def calculate_variances(images):
    return [np.var(np.abs(convolve(img.squeeze(), laplace_kernel))) for img in images]
```

Figura 24: Funzione per il calcolo delle varianze.

3. **Determinazione della soglia:** per distinguere tra immagini ad alta e bassa nitidezza, sono state calcolate le soglie di varianza al 50° percentile. In altre parole, le immagini con una varianza superiore a questo valore sono state considerate ad alta nitidezza, mentre quelle con varianza inferiore a bassa nitidezza.
4. **Filtraggio delle immagini e delle etichette:** usando queste soglie, sono state filtrate le immagini e le loro etichette. Sono state mantenute solamente le immagini con varianza superiore alla soglia mentre quelle con varianza inferiore sono state scartate.

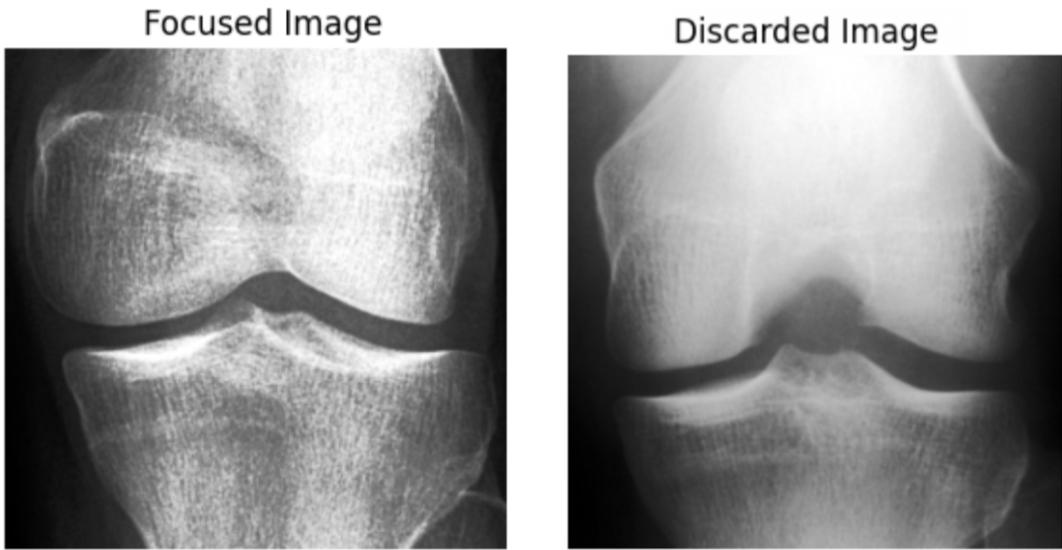


Figura 25: Esempio di immagine mantenuta e di immagine scartata.

### 3.2.3.4 Normalizzazione

La normalizzazione delle immagini aiuta a stabilizzare l'addestramento e garantisce che i gradienti durante la backpropagation non diventino né troppo grandi né troppo piccoli, migliorando così la convergenza del modello. Normalizzare in batch consente di gestire dataset di grandi dimensioni in modo efficiente, limitando l'uso della RAM. Poiché la funzione di attivazione utilizzata è tanh, che trasforma i dati in un range [-1, 1], la normalizzazione è stata fatta per portare i valori dei pixel nello stesso range.

```
def normalize_data_in_batches(data, batch_size=32):
    num_samples = data.shape[0]
    normalized_data = np.zeros_like(data, dtype=np.float32)
    for i in range(0, num_samples, batch_size):
        batch = data[i:i+batch_size]
        normalized_batch = (batch / 127.5) - 1 # Normalize to range [-1, 1]
        normalized_data[i:i+batch_size] = normalized_batch
    return normalized_data
```

Figura 26: Funzione di normalizzazione.

### 3.3 CNN

Le CNN sono una classe di reti neurali particolarmente efficaci nell'analisi e nell'interpretazione di dati visivi, come immagini e video.

#### 3.3.1 ResNet50

La scelta di questa rete è stata guidata dalla sua architettura profonda, che facilita la generalizzazione del modello, rendendola particolarmente idonea per la classificazione di questo tipo di immagini.

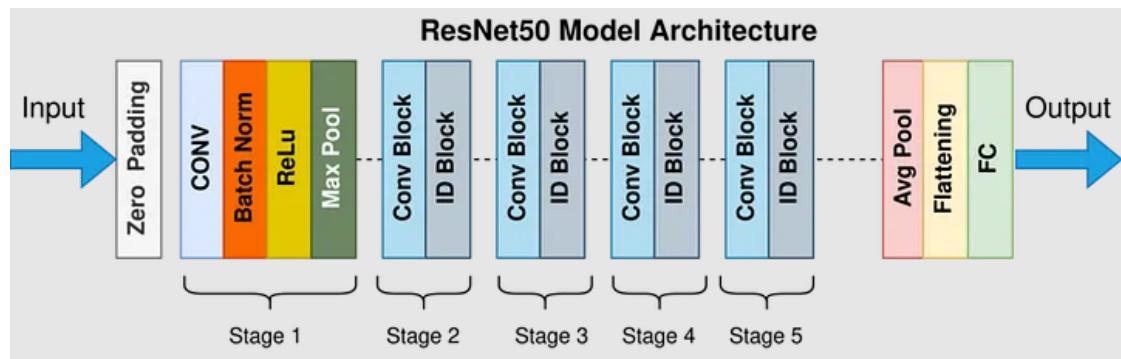


Figura 27: Architettura del modello *ResNet50* [14].

Vengono di seguito riportate le principali peculiarità del modello:

- **Skip connections**

Le skip connection risolvono il problema del *vanishing gradient*, spesso riscontrato durante la fase di *backpropagation* nelle reti deep. Queste connessioni permettono al gradiente di essere retropropagato bypassando alcuni layer, in modo che arrivi fino ai primi strati della rete senza riscontrare il problema sopracitato;

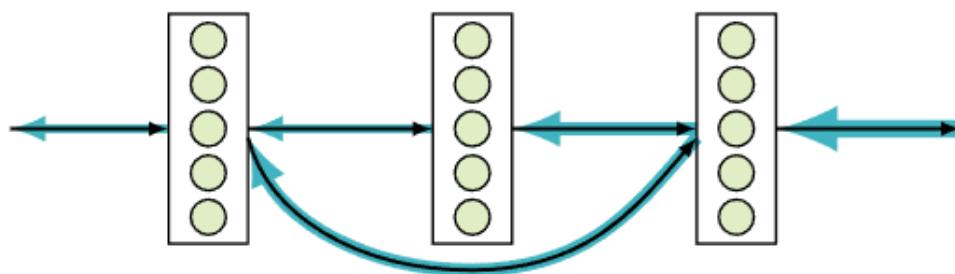


Figura 28: Rappresentazione semplificata di skip connection.

- **Residual block**

Il modello *ResNet50* è composto da quattro blocchi fondamentali chiamati **residual block**

[15]. Ognuno di essi contiene un numero variabile di sottoblocchi composti da layer convoluzionali con un numero costante di kernel. In ogni residual block sono state implementate le skip connections;

- **Batch normalization**

La batch normalization viene usata durante la fase di *training* per accelerare e stabilizzare la convergenza delle reti profonde al fine di migliorarne la generalizzazione. Questo metodo prevede il calcolo di media e varianza per ogni mini-batch per ridimensionarli e approssimarli a una distribuzione con media pari a 0 e varianza a 1. La batch normalization è stata introdotta per sopperire al problema dell'**internal covariate shift**.

Solitamente si scaricano i pesi di una ResNet50 pre-addestrata modificando i layer fully-connected nella parte finale per adattare il modello al proprio task mediante *Transfer Learning*.

### 3.3.2 Classificazione multi-classe

Per affrontare il problema di classificazione multi-classe, è stato utilizzato il Dataset I, che include immagini suddivise secondo i cinque gradi della scala KL.

Dopo aver completato tutte le fasi di pre-processing delle immagini, i dati sono pronti per l'addestramento. Il processo di sviluppo è proseguito testando diverse configurazioni di iperparametri per massimizzare le prestazioni del modello.

#### 3.3.2.1 Configurazione dei set per l'addestramento

Inizialmente sono state esplorate varie configurazioni per le dimensioni dei set di addestramento, validazione e test per garantire una distribuzione equilibrata dei dati e migliorare la robustezza del modello. Nello specifico, sono state valutate le seguenti configurazioni:

Configurazione	Train (%)	Val (%)	Test (%)
1°	70	15	15
2°	70	20	10
3°	67.5	22.5	10

Tabella 1: Configurazioni sperimentate.

La **terza configurazione** si è rivelata essere la **migliore** tra quelle testate. Questa configurazione ha permesso di mantenere un adeguato equilibrio tra i vari set. In particolare, la scelta di destinare il 22.5% dei dati al validation set ha garantito una valutazione più robusta e rappresentativa.

#### 3.3.2.2 Aggiunta di pesi alle classi

Come citato nella sezione 3.1.1, il dataset presenta un significativo sbilanciamento, con una prevalenza delle classi iniziali rispetto all'ultima. Per affrontare questa disparità è stato deciso di

assegnare pesi specifici alle classi. Questo approccio ha lo scopo di bilanciare l'influenza di ciascuna classe durante l'addestramento, compensando così il divario.

Per calcolare e applicare i pesi delle classi, è stato utilizzata la funzione `compute_class_weight()` di `scikit-learn`. Questo metodo consente di stimare pesi proporzionali alla frequenza di ciascuna classe nei dataset *sbilanciati*.

Per calcolare i pesi delle classi sono stati seguiti questi passaggi:

- **Estrazione delle etichette dalle classi;**
- **Calcolo della frequenza delle classi:** sfruttando la funzione `unique` di NumPy è stato calcolato il numero di occorrenze di ciascuna classe;
- **Stima dei pesi delle classi:** infine i pesi delle cinque classi sono stati calcolati utilizzando la funzione `compute_class_weight()`, con il parametro `class_weight` impostato su `balanced`, al fine di ottenere un bilanciamento equo tra le classi.

```
1   from sklearn.utils.class_weight import compute_class_weight  
2  
3   class_weights = compute_class_weight(  
4       class_weight="balanced", classes=np.unique(y_train), y=y_train  
5   )
```

Listing 3: Stima dei pesi.

I pesi ottenuti sono i seguenti:

Classe	Peso
0	0.51
1	1.11
2	0.76
3	1.52
4	6.63

Tabella 2: Pesi delle classi calcolati con `compute_class_weight()`.

### 3.3.2.3 Configurazione del modello

Sono stati poi selezionati e ottimizzati gli iperparametri, un aspetto fondamentale per migliorare le prestazioni. La configurazione del modello ha richiesto una valutazione approfondita di diversi parametri chiave come l'ottimizzatore, il learning rate, le tecniche di regolarizzazione e le funzioni di attivazione.

## Caricamento del modello

Il modello di *ResNet50* è stato importato utilizzando la funzione `tf.keras.applications.ResNet50` di TensorFlow.

```
1 model = tf.keras.applications.ResNet50(  
2     input_shape=(img_shape),  
3     include_top=False,  
4     weights="imagenet",  
5 )
```

Listing 4: Caricamento del modello di ResNet50 pre-addestrato su *ImageNet*.

## Configurazione dell'architettura

Il modello preaddestrato di *ResNet50* è stato configurato per adattarsi alle esigenze specifiche del progetto. In particolare, tutti i livelli del modello sono stati resi addestrabili impostando a `True` la componente `trainable`, consentendo così l'aggiornamento dei pesi durante l'addestramento.

```
1 for layer in model.layers:  
2     layer.trainable = True
```

Listing 5: Pesi del modello resi addestrabili.

## Sostituzione dei layer fully-connected

Data l'esclusione dei livelli fully-connected finali dal modello preaddestrato (parametro `include_top` impostato a `False`), sono stati manualmente aggiunti nuovi layer nella parte finale della rete per completarne la configurazione:

- **Global Average Pooling 2D layer**, che calcola la media globale di ogni feature map, trasformando l'output del modello preaddestrato in una rappresentazione adatta per la classificazione finale;
- **Dropout layer**, che applica una regolarizzazione disattivando casualmente il 20% dei neuroni durante l'addestramento per contrastare l'overfitting;
- **Dense layer**, che funge da classificatore finale. Questo layer contiene cinque neuroni corrispondenti alle classi da predire. È stata scelta la `softmax` come funzione di attivazione per calcolare le probabilità di appartenenza a ciascuna classe.

```
1 final_model = tf.keras.models.Sequential(  
2     [  
3         model,  
4         tf.keras.layers.GlobalAveragePooling2D(),  
5         tf.keras.layers.Dropout(0.2),  
6         tf.keras.layers.Dense(5, activation="softmax"),  
7     ]  
8 )
```

Listing 6: Sostituzione dei layer fully-connected

Model: "sequential"		
Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
global_average_pooling2d ( GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 5)	10245

Total params: 23597957 (90.02 MB)
Trainable params: 23544837 (89.82 MB)
Non-trainable params: 53120 (207.50 KB)

Figura 29: Struttura aggiornata del modello *ResNet50*.

### Scelta della dimensione dei batch

La dimensione dei *batch* influisce sulla stabilità e sulla velocità del processo di ottimizzazione. Durante il processo di sviluppo, sono state sperimentate diverse dimensioni per identificare la configurazione ottimale.

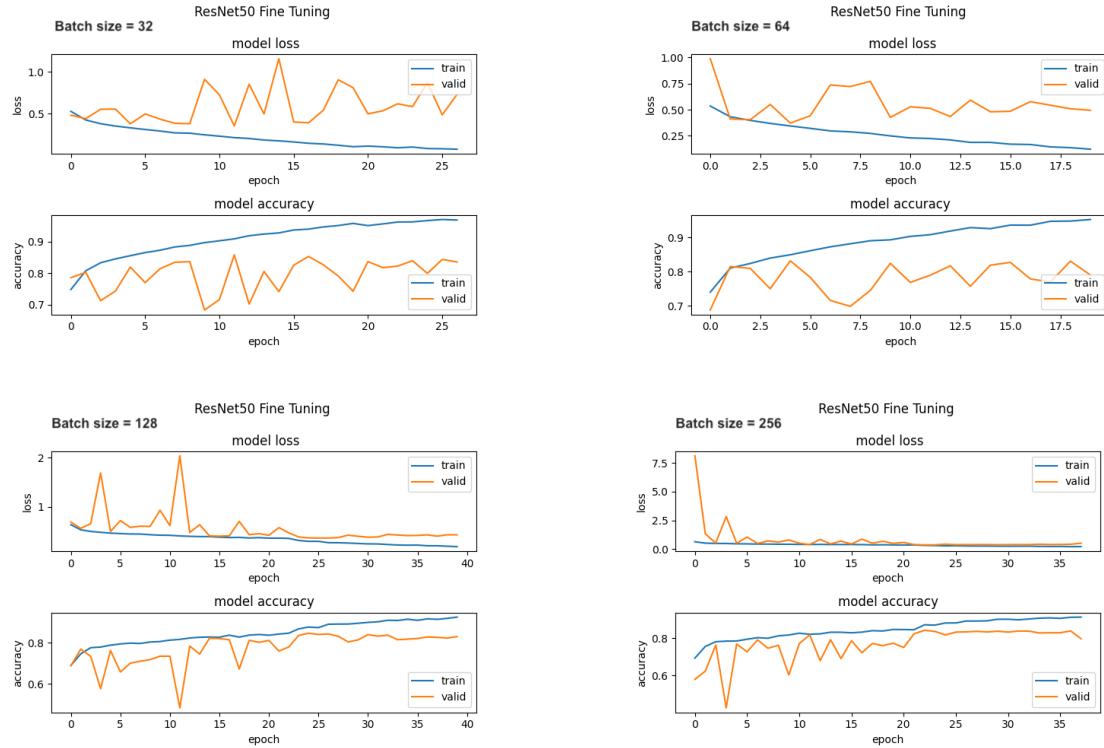


Figura 30: Risultati ottenuti con valori di batch size rispettivamente di 32, 64, 128 e 256.

In conclusione, la configurazione con una dimensione di *batch* pari a 32 si è rivelata la più efficace, offrendo un ottimo equilibrio tra stabilità dell'addestramento e tempi di esecuzione.

### Scelta dell'ottimizzatore

Sono stati sperimentati diversi ottimizzatori, ognuno dei quali è stato valutato in base alle metriche di valutazione più comuni, quali i grafici di *loss* e *accuracy* sui dati di addestramento e validazione, le metriche *F1 score*, *precision* e *recall*, e la *confusion matrix*.

In particolare, sono stati sperimentati i seguenti ottimizzatori: **Adam**, **AdamW**, **Adamax**, **Adadelta**, **Adagrad**, **Adafactor**, **Nadam**, **SGD**, **RMSProp**, **Ftrl** e **Lion**.

Tra questi, l'ottimizzatore **Adam** si è rivelato il più efficace per il compito di classificazione multi-classe, mostrando le migliori prestazioni complessive e un equilibrio ottimale tra le diverse metriche.

```
1 model_ft.compile(
2     optimizer=Adam(learning_rate=0.0001),
3     loss="categorical_crossentropy",
4     metrics=["accuracy"])
```

Listing 7: Implementazione dell'ottimizzatore Adam.

Inoltre, è stata utilizzata la funzione di loss **categorical\_crossentropy**, particolarmente adatta per la classificazione multi-classe.

### Scelta del Learning Rate

Per ottimizzare le prestazioni del modello, sono stati testati diversi valori di *learning rate* per identificare il valore più appropriato. I valori sperimentati includono **0.001**, **0.0001** e **0.0005**. Tra questi, il valore che ha dimostrato le migliori prestazioni è stato **0.0001**.

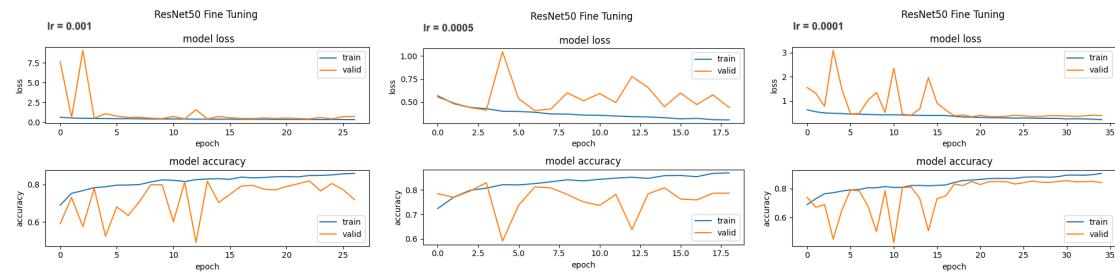


Figura 31: Risultati ottenuti rispettivamente con valori di learning rate a 0.001, 0.0005 e 0.0001.

Sono state sperimentate anche tecniche di adattamento del *learning rate* come **ReduceLROnPlateau()** e **ExponentialDecay()** ma nessuna delle due tecniche ha prodotto un significativo miglioramento delle performance.

### **Esplorazione delle tecniche di regolarizzazione**

Per ottimizzare la generalizzazione del modello, sono state testate diverse tecniche di regolarizzazione, tra cui L1, L2 e la combinazione L1-L2. Tuttavia, nonostante l'applicazione di queste metodologie finalizzate a migliorare la robustezza e prevenire l'overfitting, non è stato osservato alcun miglioramento significativo nelle performance.

#### **3.3.3 Classificazione binaria**

Il task di classificazione binaria è stato affrontato allenando la *ResNet50* su entrambi i dataset sopracitati, trattati separatamente.

Nel Dataset I le immagini sono state riorganizzate, le radiografie di classe 0 e 1 sono state aggregate in un'unica classe, mentre quelle di classe 2, 3 e 4 in un'altra. Questa riorganizzazione ha permesso di applicare un modello di classificazione binaria adeguato alla scala KL.

Non è stato necessario riorganizzare il Dataset II in quanto le immagini erano già divise in due classi, come riportato nella sezione 3.1.2.

##### **3.3.3.1 Processo di sviluppo**

Per entrambi i modelli, il processo di sviluppo ha seguito le stesse fasi fondamentali della classificazione multi-classe, con alcune differenze specifiche:

- **Dimensioni dei batch**

- **Dataset I:** è stato mantenuto un `batch_size` di 32, che ha mostrato un buon equilibrio tra stabilità e tempi di addestramento;
- **Dataset II:** è emersa fin da subito una differenza significativa tra le dimensioni di batch 64 e 128. È stata quindi testata una dimensione intermedia di 92, che ha mostrato prestazioni superiori in termini di stabilità e risultati complessivi.

- **Configurazione dei pesi iniziali**

- **Dataset I:** utilizzo di ResNet50 pre-addestrato su `ImageNet`;
- **Dataset II:** utilizzo di ResNet50 **non** pre-addestrato (parametro `weights` impostato a `None`), vista la dimensione del set di dati;

- **Ottimizzatore**

- **Dataset I:** Adamax;
- **Dataset II:** Adam;

- **Architettura finale del modello:** per entrambi i dataset, l'architettura finale del modello è stata adattata per la classificazione binaria. È stato aggiunto un layer *Dense* composto da 2 neuroni.

### 3.4 ViT

Il ViT è una rete neurale che applica i concetti del Transformer, originariamente sviluppato per il Natural Language Processing (NLP), alla visione artificiale. Il ViT suddivide le immagini in piccole patch non sovrapposte e tratta queste patch come token, analogamente alle parole di un testo, per catturare relazioni a lungo termine tra le caratteristiche dell'immagine.

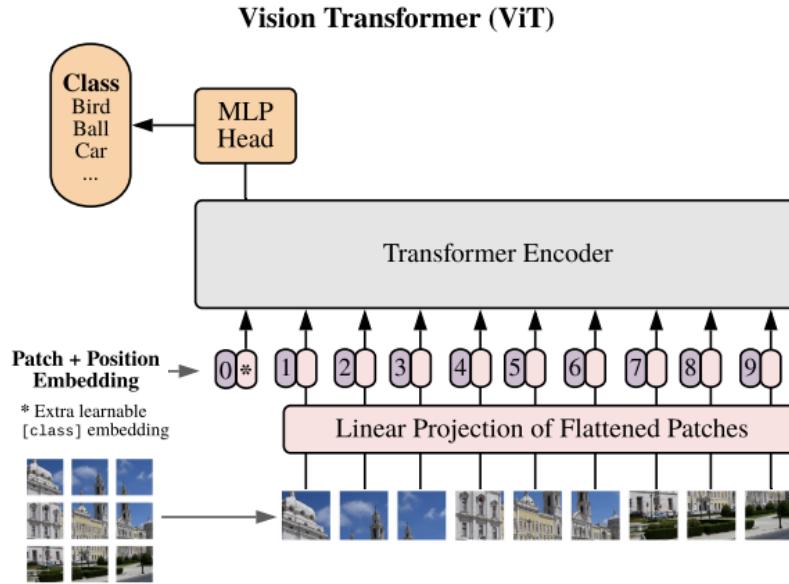


Figura 32: Struttura del Vision Transformer

#### 3.4.1 Architettura

Il funzionamento del ViT può essere suddiviso in queste fasi chiave:

- Suddivisione delle immagini in patch
- Embedding delle patch
- Blocco Transformer
- Classificazione

##### 3.4.1.1 Suddivisione delle immagini in patch

Le immagini di input vengono suddivise in patch di dimensione prefissata. Questo processo trasforma un'immagine in una sequenza di porzioni più piccole, facilitando l'applicazione dei meccanismi di *attention* dei Transformer.

Ad esempio, se abbiamo un’immagine di 32x32 pixel e la suddividiamo in patch di 8x8 pixel, ne otterremo 16 (4 righe e 4 colonne).

### 3.4.1.2 Embedding delle patch

Dopo aver suddiviso l’immagine, ciascuna patch deve essere convertita in un vettore che può essere utilizzato dal modello. Questo processo è chiamato *embedding delle patch*. In pratica, ogni patch viene *tradotta* in un vettore di numeri (*embedding*) che rappresenta le sue caratteristiche. Per fare ciò, utilizziamo un dense layer che trasforma ogni patch in un vettore di dimensione predefinita rendendole comparabili tra loro.

Inoltre, per mantenere l’ordine delle patch (poiché la posizione all’interno dell’immagine è importante), aggiungiamo una *codifica posizionale* univoca.

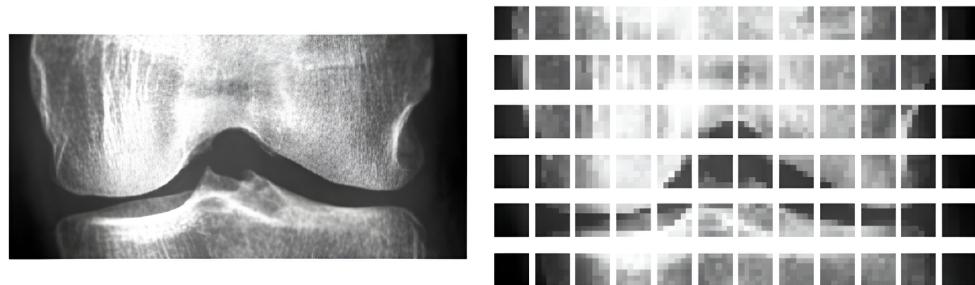


Figura 33: Suddivisione dell’immagine in patch, dopo aver eseguito le operazioni di pre-processing

### 3.4.1.3 Blocco Transformer

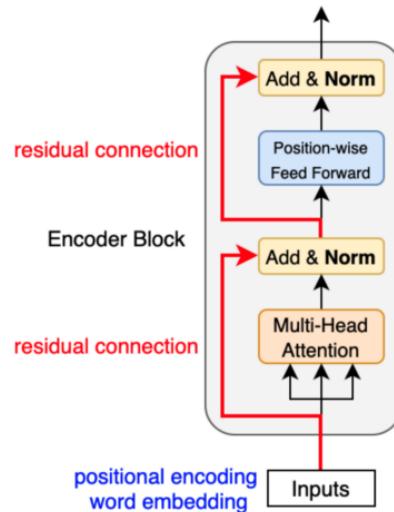


Figura 34: Architettura del blocco transformer

Gli embedding delle patch vengono processati da una serie di blocchi Transformer, più precisamente sfruttando la parte encoder. Ognuno di questi, come indicato dalla figura 34, contiene diverse sottoparti:

- **Multi-head self-attention:** permette al modello di determinare l'importanza di ogni patch rispetto a tutte le altre. Nello specifico, grazie ad un uso ripetuto del modulo di self-attention, il modello è in grado di catturare non solo la relazione più preponderante tra i token, bensì più relazioni con diversi gradi di complessità. In particolare, il numero di relazioni catturate è proporzionale al numero di *teste* usate.
- **Layer normalization**
- **MLP (Multy-Layer Perceptron):** dopo la multi-head self-attention, le rappresentazioni delle patch vengono ulteriormente elaborate da una MLP, che consiste in uno o più layer densi con attivazioni non lineari (come GELU, utilizzata nel progetto) e dropout. Questo viene fatto per processare ulteriormente la patch, applicando trasformazioni non lineari ai dati.

Questi componenti vengono applicati in cascata consentendo al modello di raffinare progressivamente le rappresentazioni delle patch attraverso più layer.

#### 3.4.1.4 Classificazione

L'output prodotto dal blocco trasformer viene infine passato ad un'altra MLP Head per produrre la classificazione finale. Nel caso della classificazione multi-classe, viene utilizzato un layer denso con una funzione di attivazione softmax.

Nel problema di classificazione binaria, invece, viene usata la sigmoide come funzione di attivazione nell'ultimo layer. La sigmoide è più adatta per questo tipo di problemi poiché produce un output compreso tra 0 e 1.

### 3.4.2 Scelte progettuali

#### 3.4.2.1 Data modification

In questa sezione sono riportate le operazioni che vengono utilizzate per migliorare la robustezza e la capacità di generalizzazione del modello. Queste istruzioni non vengono eseguite nella sezione di pre-processing, dove verrebbero richiamate una sola volta, bensì vengono eseguite in tempo reale durante l'addestramento. Questo approccio è noto come *data augmentation on-the-fly*, rinominato qui `data_modification`, e va ad applicare diverse operazioni ad ogni mini-batch.

Questo blocco include diversi layer, ognuno dei quali non va ad aumentare i dati a disposizione del modello, ma va a modificarli volta per volta:

- **Normalizzazione**

- **Ridimensionamento**, che facilita l’elaborazione da parte della rete, riducendo il carico computazionale necessario per processare le immagini
- **Rotazione e zoom casuali**, che aumentano la varietà delle immagini di addestramento. In particolare, la rotazione casuale al 2% e lo zoom al 20% simulano scenari reali in cui le immagini possono essere acquisite da angolazioni e distanze leggermente diverse, come spiegato nel capitolo 2.1.1.3.

Nel caso del Dataset II, queste tecniche hanno contribuito a gestire efficacemente il grande volume di dati, prevenendo problemi di sovraccarico della memoria.

```
data_modification = keras.Sequential(
    [
        layers.Normalization(),
        layers.Resizing(36, 72),
        layers.RandomRotation(factor=0.02),
        layers.RandomZoom(height_factor=0.2, width_factor=0.2),
    ]
)
```

La funzione `data_modification` è integrata direttamente nel modello ViT all’interno della definizione del modello. Durante la creazione del modello, la `data_modification` viene applicata ai dati di input. Questo significa che ogni volta che il modello processa un batch di dati durante l’addestramento, le immagini di input vengono normalizzate, ridimensionate e sottoposte a trasformazioni casuali. Vieni qui riportata la parte di codice relativa a tale integrazione:

```
def create_vit_classifier():
    inputs = keras.Input(shape=input_shape)
    augmented = data_modification(inputs)
    patches = Patches(patch_size)(augmented)
    ...
    return model
```

### 3.4.2.2 Evoluzione del progetto: dal multi-classe al binario

L’evoluzione del progetto ha seguito un percorso significativo, partendo dalla classificazione multi-classe per poi focalizzarsi sulla classificazione binaria. Inizialmente, il modello ViT è stato progettato per la classificazione a 5 classi secondo i gradi Kellgren-Lawrence, utilizzando un dataset composto da 9600 immagini. Tuttavia, nonostante gli sforzi, l’accuratezza ottenuta sul validation set si attestava solo intorno al 52%. Di fronte a questi risultati insoddisfacenti, è stata introdotta la data augmentation con l’obiettivo di migliorare le prestazioni del modello.

L’aggiunta della data augmentation, che includeva trasformazioni come rotazioni e zoom casuali delle immagini direttamente nella fase di pre-processing, non ha però portato ai miglioramenti

sperati. L'accuratezza del modello è rimasta invariata, suggerendo che il problema non risiedeva nella variabilità del dataset, ma piuttosto nella complessità del compito di classificazione multi-classe o nelle dimensioni limitate del dataset stesso.

A questo punto, la strategia è stata rivista, concentrandosi sul problema della classificazione binaria. I gradi KL sono stati raggruppati in due classi: i gradi 2, 3 e 4 sono stati classificati come *ginocchio con necessità di protesi*, mentre i gradi 0 e 1 come *ginocchio sano*. Mantenendo l'approccio senza data augmentation, basato sulle osservazioni precedenti, è stato utilizzato lo stesso dataset di 9600 immagini per il nuovo problema binario.

Tuttavia, anche in questo caso, sono emerse delle problematiche: la loss di validazione superava la loss di training, indicando un possibile problema di overfitting. Invece di reintrodurre la data augmentation, come fatto nel tentativo precedente con il problema multi-classe, si è deciso di esplorare nuove opzioni di dataset. È stato quindi individuato un nuovo dataset composto da 320000 immagini sintetiche, caratterizzate da una maggiore uniformità e qualità.

L'adozione di questo nuovo dataset ha rappresentato una svolta decisiva. La maggiore quantità di dati e la loro natura sintetica hanno permesso di addestrare il modello in modo più efficace, riducendo il problema di overfitting. Grazie a questo approccio, è stata raggiunta un'accuratezza del 99% nella classificazione binaria.

### 3.5 WGAN-GP

Le GAN sono una classe di reti neurali efficaci nella generazione di dati sintetici, utili per aggirare i problemi di privacy menzionati in precedenza 2.3.1.

La GAN presentata in questo progetto è una WGAN-GP (Wasserstein GAN con Gradient Penalty) progettata fine di **aumentare il dataset di immagini**.

A differenza delle GAN tradizionali, che utilizzano la divergenza di Jensen-Shannon, la WGAN si basa sulla *distanza di Wasserstein* come funzione di costo per migliorare l'allenamento. Inoltre, la WGAN-GP introduce un *gradient penalty* del discriminatore per campioni interpolati (campioni generati dall'interpolazione lineare di campioni reali e campioni sintetici). La gradient penalty assicura che il discriminatore, che cerca di distinguere tra immagini reali e generate, **mantenga gradienti stabili e non diventi troppo "sicuro" nelle sue valutazioni**.

$$L = \underbrace{\mathbb{E}_{\tilde{x} \sim \mathbb{P}_g} [D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Our gradient penalty}}.$$

Figura 35: Grazie alla gradient penalty, WGAN-GP penalizza il modello se la norma del gradiente si allontana dal valore della norma target 1.

Si è scelto di adottare proprio una WGAN-GP invece di una GAN tradizionale o una DCGAN (Deep Convolutional GAN) per diversi motivi:

- **Qualità delle immagini:** le WGAN-GP, come detto, usa la distanza di Wasserstein, che fornisce una misura più robusta della distanza tra distribuzioni, il che permette di ottenere una qualità superiore delle immagini generate [16].

<b>Min. score</b>	<b>Only GAN</b>	<b>Only WGAN-GP</b>	<b>Both succeeded</b>	<b>Both failed</b>
1.0	0	8	192	0
3.0	1	88	110	1
5.0	0	147	42	11
7.0	1	104	5	90
9.0	0	0	0	200

Figura 36: Risultati dell'addestramento di 200 configurazioni casuali per diverse soglie di successo [17].



Figura 37: Diversi modelli di GAN addestrati con metodi differenti e con un insieme comune di iperparametri mostrano come la WGAN-GP in tutte le situazioni generi immagini di qualità superiore [17].

- **Gradient Penalty:** l'aggiunta di una penalità sul gradiente aiuta a mantenere i gradienti sotto controllo evitando il problema del vanishing gradient, oltre a ridurre l'overfitting [18].

```

def gradient_penalty(discriminator, real_images, fake_images, conditions, batch_size):
    alpha = tf.random.uniform([batch_size, 1, 1, 1], 0.0, 1.0)
    diff = fake_images - real_images
    interpolated = real_images + alpha * diff

    with tf.GradientTape() as gp_tape:
        gp_tape.watch(interpolated)
        pred = discriminator([interpolated, conditions], training=True)

    grads = gp_tape.gradient(pred, [interpolated])[0]
    norm = tf.sqrt(tf.reduce_sum(tf.square(grads), axis=[1, 2, 3]))
    gp = tf.reduce_mean((norm - 1.0) ** 2)
    return gp

```

Figura 38: Funzione per il calcolo della gradient penalty usata all'interno del progetto.

- **Stabilità nelle iterazioni prolungate:** la WGAN-GP è più stabile durante le iterazioni prolungate di addestramento. In scenari dove è richiesto un addestramento estensivo per generare immagini di alta qualità, questa stabilità è fondamentale per evitare il collasso del modello [19].

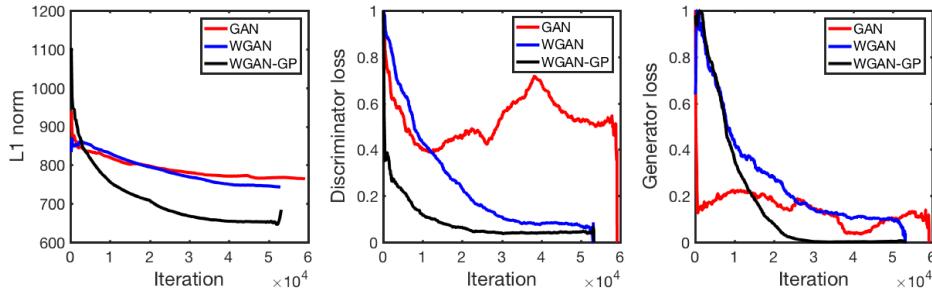


Figura 39: Differenze sulle loss negli addestramenti di GAN, WGAN e WGAN-GP [17].

- **Applicazione efficace a dati complessi:** la struttura e la metodologia delle WGAN sono particolarmente efficaci nella gestione di dati complessi come le immagini mediche, che possono presentare caratteristiche non banali da modellare [20].
- **Robustezza ai rumori nei dati di addestramento:** la WGAN-GP è più robusta al rumore presente nei dati di training, che può essere frequente nelle immagini di radiografie a causa di variazioni nelle condizioni di acquisizione.

Va però tenuto in considerazione che le WGAN-GP tendono a convergere più lentamente rispetto alle DCGAN [21]. Inoltre è stato riscontrato nella maggior parte dei paper come le DCGAN ottengano un valore superiore di inception score, metrica utilizzata per valutare la qualità e la diversità delle immagini generate.

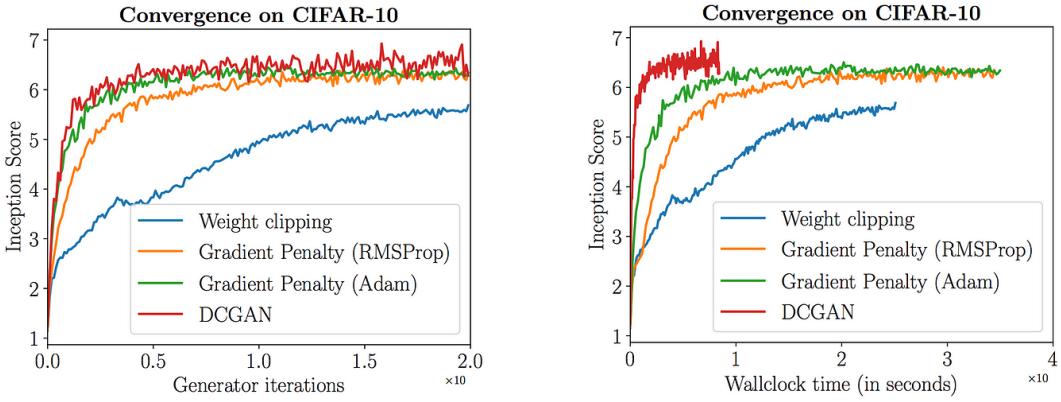


Figura 40: L'inception score su CIFAR-10 in funzione delle iterazioni del generatore (a sinistra) o del tempo (a destra) per quattro modelli: WGAN con clipping dei pesi, WGAN-GP con RMSProp e Adam, e DCGAN. WGAN-GP ha prestazioni di poco inferiori a DCGAN [17].

Method	Inception Score
DCGAN [16]	$6.16 \pm .07$
EBGAN [21]	$7.07 \pm .10$
WGAN-GP [7]	$7.86 \pm .07$
CT GAN [20]	$8.12 \pm .12$
SNGAN [14]	$8.22 \pm .05$
$W^{-\frac{3}{2}, 2}$ -BWGAN	$8.26 \pm .07$
$L^{10}$ -BWGAN	$8.31 \pm .07$
Progressive GAN [9]	$8.80 \pm .05$

Figura 41: Inception scores su CIFAR-10 [22].

### 3.5.1 Componenti della WGAN-GP

#### 3.5.1.1 Generatore

Il generatore implementato nel progetto prende in input un rumore casuale e una condizione, li concatena e li trasforma in un'immagine sintetica attraverso una serie di convolutional layer. L'architettura del generatore è caratterizzata da:

- **Batch normalization e Leaky ReLU:** per stabilizzare l'addestramento e velocizzare la convergenza.
- **Residual block:** ogni blocco contiene layer convoluzionali e di batch normalization, con skip connections per migliorare la qualità delle immagini e rendere il modello più robusto.
- **Self-attention:** utilizzata in un layer intermedio per migliorare la capacità del modello di catturare dettagli globali, essenziale per immagini complesse come le radiografie.

- **Up-Sampling layer:** per aumentare la dimensione dell'immagine fino a raggiungere la risoluzione desiderata.

### 3.5.1.2 Discriminatore

Il discriminatore riceve in input un'immagine e una label che indica se l'immagine è reale o generata. Questi input vengono concatenati e passati attraverso una serie di layer convoluzionali, al fine di determinare se l'immagine è autentica o sintetica. L'architettura del discriminatore è caratterizzata da:

- **Convolutional blocks:** layer convoluzionali con Leaky ReLU usati per estrarre le caratteristiche delle immagini.
- **Flatten e dense layers:** le feature estratte sono appiattite e passate attraverso layer densi per ottenere una valutazione finale.

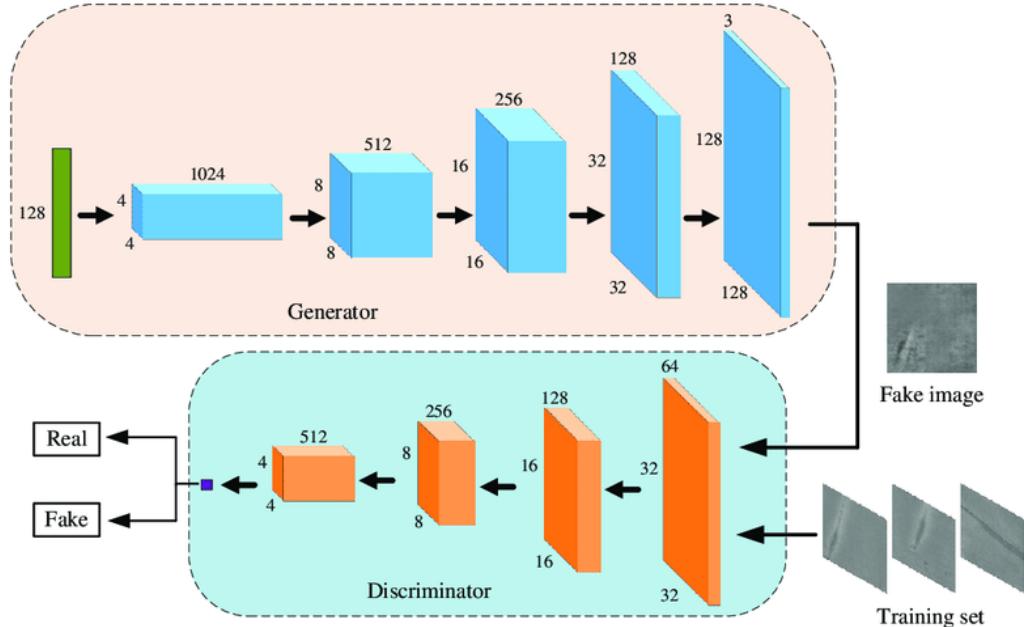


Figura 42: Modello di WGAN-GP rappresentativa del modello implementato.

Nei file esterni `generator_model.png` e `discriminator_model.png` sono presenti la struttura completa di generatore e discriminatore.

### 3.5.2 Scelte progettuali

L'architettura del generatore e del discriminatore è stata attentamente bilanciata per garantire un numero simile di parametri in entrambi i modelli. Questo equilibrio è cruciale per evitare

scenari in cui il generatore possa costantemente "ingannare" il discriminatore o, al contrario, in cui il discriminatore sia sempre in grado di identificare le immagini generate come false.

```
Total params: 56643393 (216.08 MB)
Trainable params: 55898241 (213.23 MB)
Non-trainable params: 745152 (2.84 MB)
```

```
Total params: 55731692 (212.60 MB)
Trainable params: 55731692 (212.60 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figura 43: Numero di pesi rispettivamente di generatore e discriminatore.

La complessità di entrambi i modelli è il risultato di un trade-off tra le risorse computazionali disponibili in termini di memoria RAM e capacità della GPU, i tempi di addestramento e la qualità delle immagini generate, in modo che fossero di alta qualità e fedeli alle caratteristiche del dataset originale.

Il processo di ottimizzazione ha richiesto numerosi esperimenti e iterazioni. Sono state testate architetture molto complesse con oltre 150 milioni di parametri, che si sono rivelate impraticabili a causa delle limitazioni hardware. D'altra parte, modelli più semplici con meno di 3 milioni di parametri producevano immagini rumorose e non riuscivano a catturare le caratteristiche di alto livello necessarie per una riproduzione fedele. Dopo molteplici tentativi, si è giunti a un'architettura ottimale che permette di generare immagini di qualità soddisfacente, catturando sia i dettagli che le strutture complesse, pur mantenendo tempi di addestramento ragionevoli e rispettando i vincoli hardware.

Per l'addestramento dei modelli è stato scelto l'ottimizzatore Adam. Si è optato per questa soluzione banalmente perchè ha mostrato i risultati migliori, infatti sono state provati anche altri ottimizzatori che hanno dato risultati simili ma leggermente peggiori.

```
generator_optimizer = tf.keras.optimizers.Adam(learning_rate=current_learning_rate, beta_1=beta_1, beta_2=beta_2)
discriminator_optimizer = tf.keras.optimizers.Adam(learning_rate=current_learning_rate, beta_1=beta_1, beta_2=beta_2)
```

Figura 44: Ottimizzatore usato all'interno del progetto.

La scelta dei parametri è stata valutata su diversi addestramenti e su determinate nozioni che hanno fatto propendere per aumentarli o diminuirli man mano che gli addestramenti venivano completati. I parametri valutati principalmente sono stati:

- **batch\_size**: aumentandolo è stata riscontrata un maggiore velocità nell'addestramento a discapito di una maggiore richiesta di memoria. Ha condotto a convergenze più stabili ma meno precise. Essendo limitati a livello di risorse RAM e ricercando una convergenza stabile, si è optato per tenere un valore basso. (**batch\_size = 32**).
- **n\_critic**: aumentare questo valore ha rallentato l'addestramento del generatore ma ha migliorato la qualità del discriminatore, pertanto è stato scelto un valore tendenzialmente più alto rispetto ai valori riscontrati in altri studi (**n\_critic = 10**).

- **lambda\_gp**: un aumento di questo valore, ha sempre portato ad un maggiore rafforzamento della gradient penalty (potenzialmente questo dovrebbe stabilizzare l’addestramento e infatti le differenze sono state significative) con un peggioramento della flessibilità del discriminatore. Dal momento che il nostro caso di studio non richiedeva una flessibilità del discriminatore tanto elevata, questo ha portato anche qui a scegliere un valore più alto (`lambda_gp = 15`).
- **initial\_learning\_rate**: si è optato per un learning rate iniziale molto basso, dal momento che l’obiettivo era quello di avere una convergenza più stabile (vedi anche la scelta della `batch_size`), non essendo interessati alla velocità dell’apprendimento (`initial_learning_rate = 0.000001`).
- **beta\_1 e beta\_2**: aumentare questi parametri legati all’ottimizzatore, ha fatto sì che questo ultimo considerasse una storia più lunga dei gradienti, portando a un addestramento più stabile ma più lento nell’adattarsi ai cambiamenti rapidi. Anche in questo caso è stata ricercata stabilità nell’addestramento, pertanto i valori di entrambi sono risultati essere molto alti (`beta_1 = 0.9` e `beta_2 = 0.99`).

## 4 Analisi delle performance e Considerazioni

### 4.1 CNN

Per valutare le performance della *ResNet50*, sono state considerate diverse metriche. In particolare, sono stati esaminati i grafici di loss e accuracy per le fasi di training e validation, l'accuracy sul test set, precision, recall, F1-score, confusion matrix e la Grad-CAM.

#### 4.1.1 Classificazione Multi-classe

Per risolvere il problema di classificazione multi-classe, è stato utilizzato il Dataset I, che include immagini suddivise secondo i cinque gradi della scala KL.

##### 4.1.1.1 Loss e accuracy

Dal grafico in figura 45 si osserva come l'accuratezza sul training set sia **costantemente in aumento**, segnalando un efficace apprendimento del modello. Tuttavia, l'andamento del grafico per il validation set presenta notevoli oscillazioni e **non segue un trend crescente**, indicando un evidente **segnale di overfitting**.

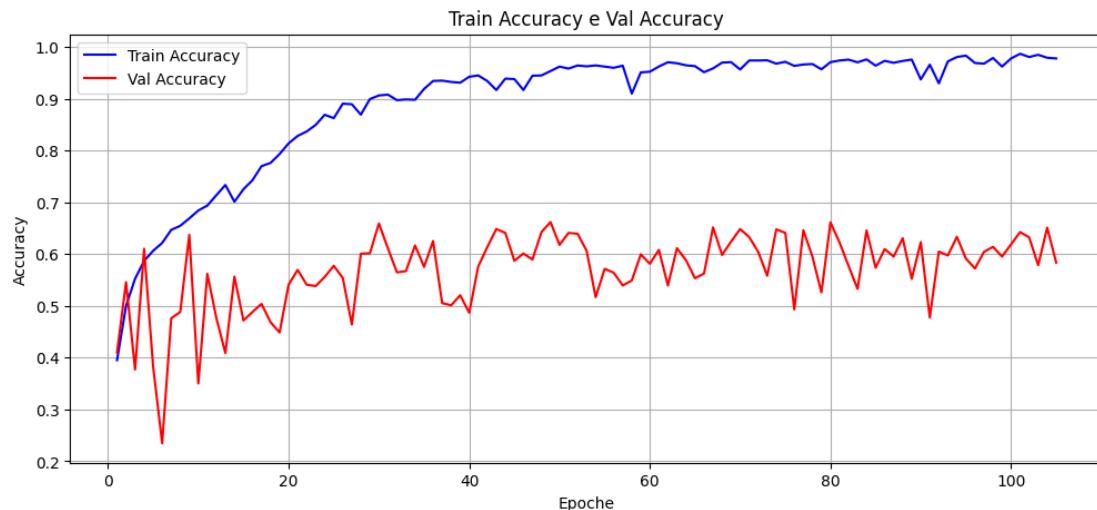


Figura 45: Grafico dell'accuracy.

Dal grafico della loss in figura 46 si nota chiaramente una **conferma del problema di overfitting** evidenziato dall'analisi dell'accuracy. **La loss sul training set diminuisce costantemente**, raggiungendo valori molto bassi, mentre la **loss sul validation set mostra un andamento irregolare**.

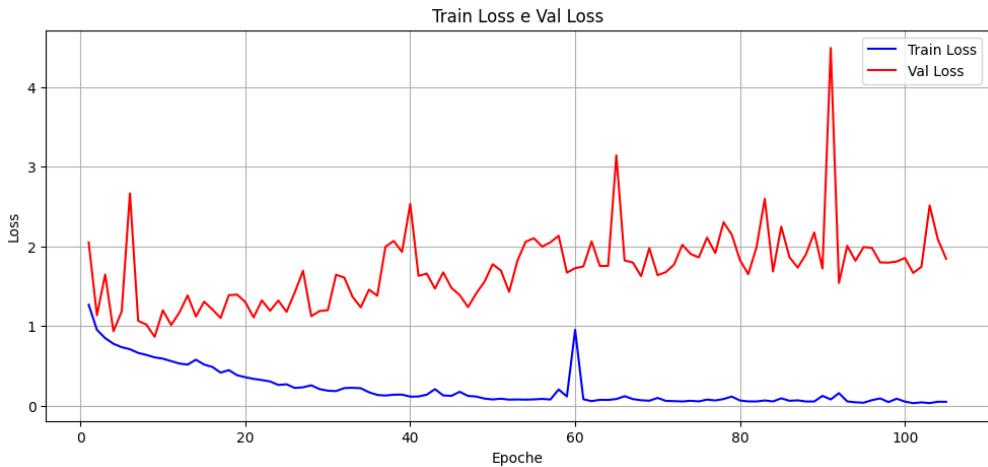


Figura 46: Grafico della loss.

Questo problema di *overfitting* è probabilmente dovuto alla natura delle radiografie dato che sono particolarmente simili tra loro. Per mitigare questo problema, come descritto nella sezione 3.3.2, sono state applicate diverse tecniche, quali data augmentation, regolarizzazione e dropout.

Nonostante le difficoltà sopracitate, il modello ha raggiunto una precisione sul test set pari all'**88.3%**, dimostrando una buona capacità di classificazione. L'accuratezza riscontrata nel progetto supera significativamente quella riportata nel paper [7], che era del 67% (Figura 5).

#### 4.1.1.2 Precision, Recall e F1-score

L'analisi dei valori di *precision*, *recall* e *F1 score* rivela performance elevate per tutti i cinque gradi.

	Precision	Recall	F1
Grado 0	0.886	0.933	0.909
Grado 1	0.772	0.747	0.759
Grado 2	0.907	0.853	0.879
Grado 3	0.945	0.954	0.950
Grado 4	1.000	1.000	1.000
<b>Media</b>	<b>0.883</b>	<b>0.883</b>	<b>0.882</b>

Tabella 3: Risultati di Precision, Recall e F1 per ciascun grado.

La classe 4, il grado KL più alto, evidenzia le prestazioni più elevate, con una precisione e una recall pari al 100%. Mentre la classe 1, presenta le performance peggiori, con precisione e recall pari a 0.77 e 0.75.

Questa discrepanza può essere attribuita alle sottili differenze tra le immagini di grado 0 e 1, che rendono più difficile la distinzione tra queste due classi.

Gli F1 score oscillano tra 0.76 e 1.00, con una media di 0.88. Questi risultati sottolineano l'efficace capacità di classificazione del modello, sebbene rivelino anche le sfide nel distinguere con precisione le immagini di grado 1.

#### 4.1.1.3 Confusion matrix

Nella figura 47 sono riportate due confusion matrix. La prima mostra le percentuali di accuratezza per ogni classe, mentre la seconda riporta il numero assoluto di esempi correttamente ed erroneamente classificati.

Entrambe le matrici evidenziano un'accuratezza complessivamente elevata, indicando l'ottima capacità del modello nel distinguere correttamente i diversi gradi. Si osservano ottime performance per tutte le cinque classi, con valori che variano tra il 75% e il 100%.

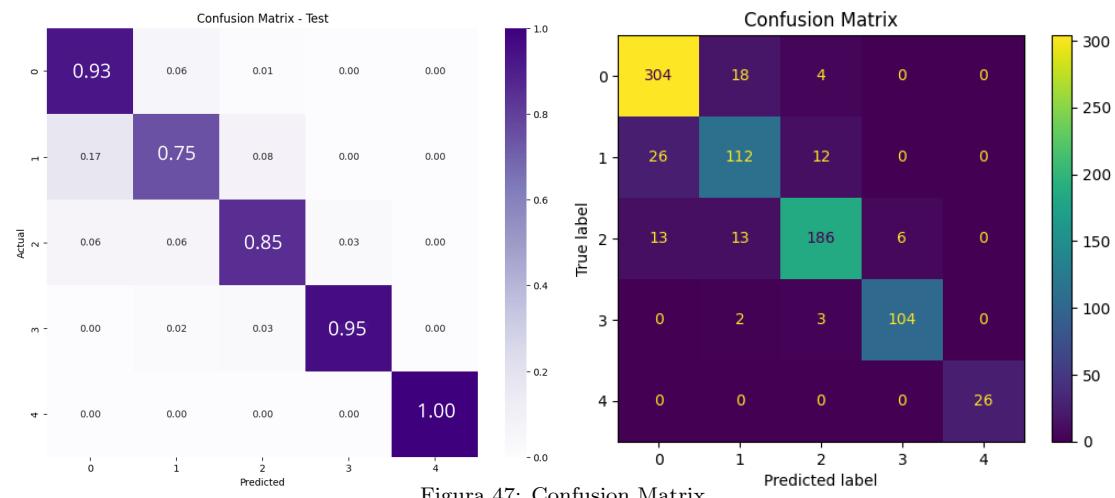


Figura 47: Confusion Matrix.

La **classe 1** presenta l'accuratezza più bassa, con solo il 75% delle istanze classificate correttamente, un 17% confuso con la classe 0 e un 8% con la classe 2. Questo trend negativo è confermato dalla matrice assoluta, dove si nota che su 150 istanze della classe 1, 26 sono state erroneamente classificate come classe 0 e 12 come classe 2.

In conclusione, si osservano principalmente **errori tra classi adiacenti**, il che è comprensibile data la natura progressiva dell'osteoartite, che comporta una notevole similarità tra immagini di gradi consecutivi.

#### 4.1.1.4 Grad-CAM

L'implementazione della Grad-CAM ha contribuito significativamente alla comprensione del processo decisionale della *ResNet50*. Questa tecnica permette di **evidenziare le regioni dell'immagine che influenzano maggiormente le decisioni del modello**.

Nella figura 48, i colori più caldi, indicano le aree di maggiore rilevanza, mentre i toni più freddi rappresentano regioni di minore importanza.



Figura 48: Applicazione della Grad-CAM.

Nella figura 48 si osserva chiaramente la focalizzazione del modello sullo spazio articolare del ginocchio e sulle strutture ossee adiacenti, **arie cruciali per classificare correttamente i diversi gradi di osteoartrite**.

#### 4.1.2 Classificazione Binaria Dataset I

Per risolvere il problema di classificazione binaria, il Dataset I è stato suddiviso in due classi distinte: una con le immagini di grado 0 e 1, e l'altra con quelle di grado 2, 3 e 4.

##### 4.1.2.1 Loss e accuracy

Il grafico in figura 49 evidenzia come l'accuracy sul training set sia in **costante miglioramento**, raggiungendo e mantenendo valori molto alti. Mentre, l'accuracy sul validation set mostra **un'andamento piuttosto oscillante**, indicando anche in questo caso la **presenza di overfitting**.

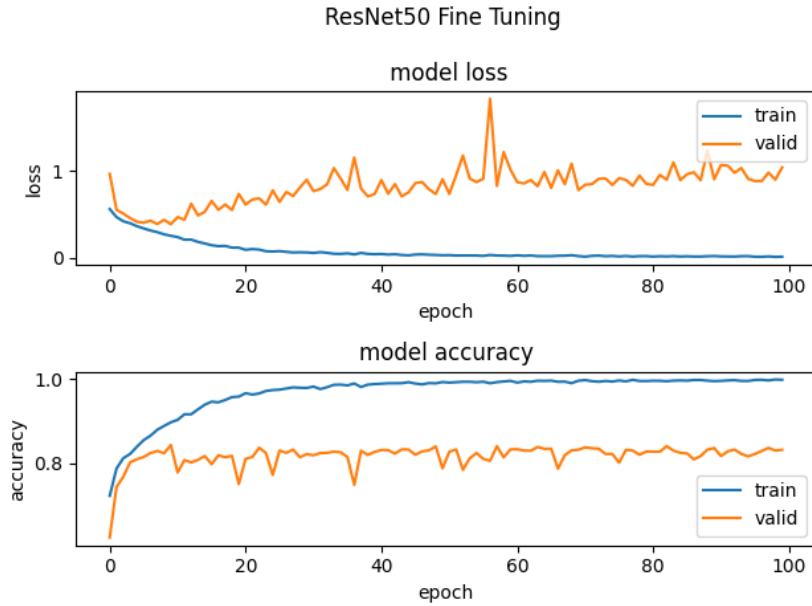


Figura 49: Grafico di loss e accuracy.

Analogamente, la loss sul training set è significativamente bassa mentre quella sul validation set risulta elevata, confermando il **problema di *overfitting***. Anche in questo caso sono state applicate le tecniche sopracitate.

Il modello ha raggiunto un'ottima percentuale di accuracy sul test set pari al **93.7%**. Anche in questo caso l'accuratezza riscontrata nell'articolo [7], pari all'83%, è stata superata.

#### 4.1.2.2 Precision, Recall e F1-score

Anche l'analisi dei valori di *precision*, *recall* e *F1 score* rivela performance eccellenti per entrambe le classi.

	Precision	Recall	F1
Negative	0.925	0.912	0.918
Positive	0.935	0.945	0.940
<b>Media</b>	<b>0.931</b>	<b>0.931</b>	<b>0.931</b>

Tabella 4: Risultati di Precision, Recall e F1 per entrambe le classi.

#### 4.1.2.3 Confusion matrix

Anche le matrici di confusione confermano le ottime performance del modello, mostrando percentuali di accuratezza del **91%** per la classe *Negative* e del **95%** per la classe *Positive*.

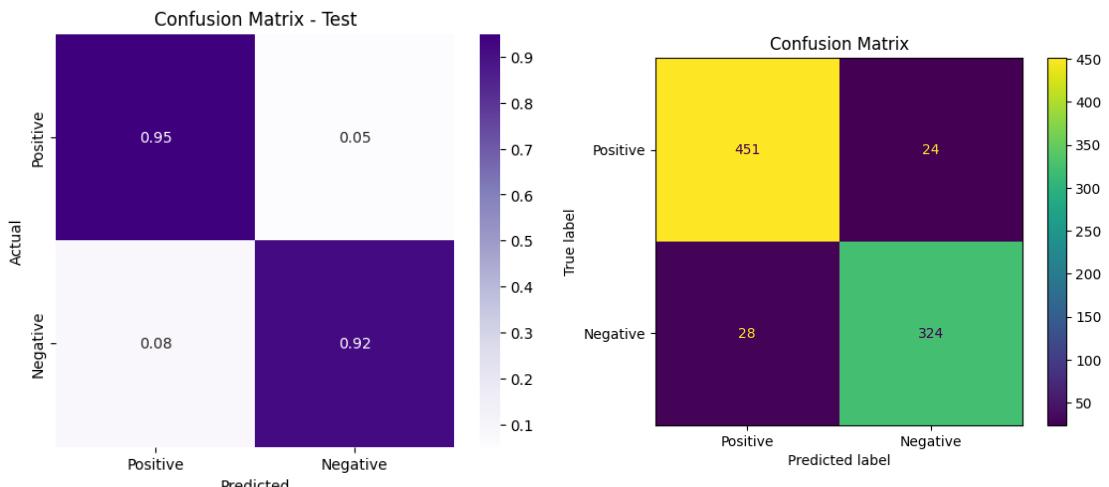


Figura 50: Confusion Matrix.

#### 4.1.2.4 Grad-CAM

Di seguito è riportato un esempio di applicazione della Grad-CAM su un'immagine di test appartenente alla classe *Positive*.

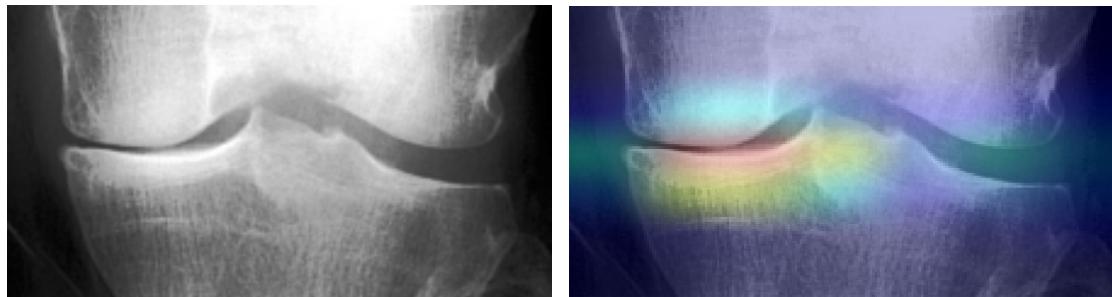


Figura 51: Applicazione della Grad-CAM.

#### 4.1.3 Classificazione Binaria Dataset II

##### 4.1.3.1 Loss e accuracy

Considerata la dimensione del dataset e le risorse computazionali a disposizione, sono state eseguite cinque epoche di addestramento.

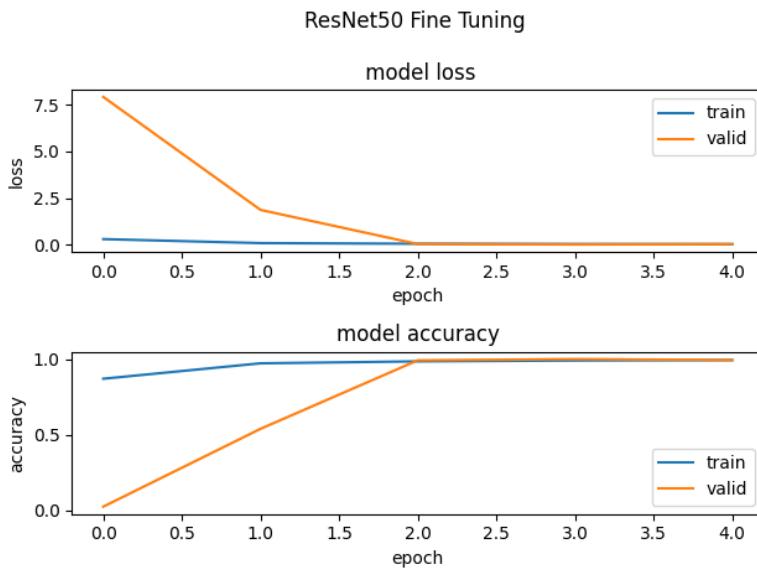


Figura 52: Andamento della loss e dell'accuracy.

Dai grafici in figura 52 emerge chiaramente la **veloce convergenza del modello**.

Il modello ha raggiunto un livello di precisione sul test set eccellente, pari al **99.937%**.

#### 4.1.3.2 Precision, Recall e F1-score

Sono di seguito riportati i valori di *precision*, *recall* e *F1 score* riscontrati:

	Precision	Recall	F1
Positive	0.99900	0.99975	0.99938
Negative	0.99975	0.99900	0.99938
Media	<b>0.99938</b>	<b>0.99938</b>	<b>0.99937</b>

Tabella 5: Risultati di Precision, Recall e F1 per entrambe le classi.

#### 4.1.3.3 Confusion matrix

Le matrici di confusione riportate in figura 53 evidenziano le prestazioni impressionanti del modello. **Su un totale di 32000 immagini di test la rete commette solamente 19 errori**, 6 falsi negativi e 13 falsi positivi.

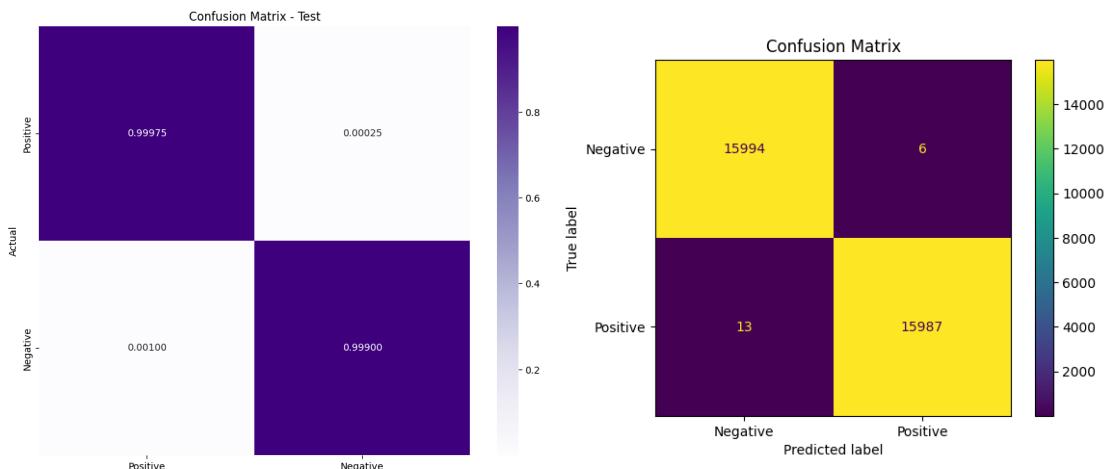


Figura 53: Confusion matrix.

#### 4.1.3.4 Grad-CAM

È di seguito riportato un esempio di applicazione della Grad-CAM.

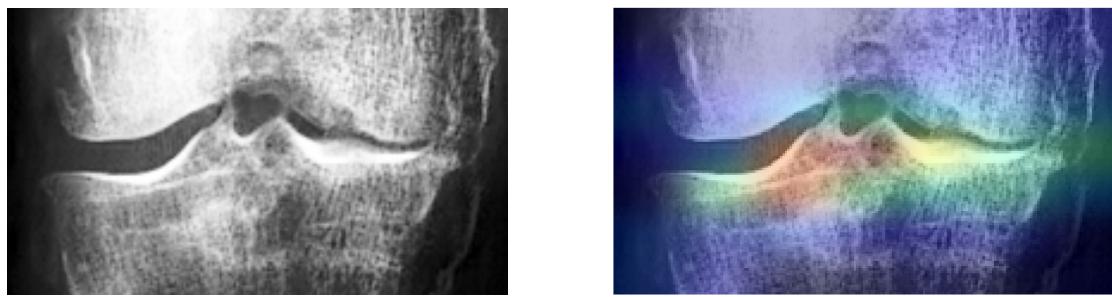


Figura 54: Applicazione della Grad-CAM.

## 4.2 ViT

### 4.2.1 Classificazione Multi-classe

L’analisi delle performance del modello ha evidenziato diversi limiti e aree di potenziale miglioramento. Nonostante il ViT sia una tecnica avanzata di image recognition, i risultati ottenuti non sono stati soddisfacenti, con un’accuracy circa del **52%** e valori di *precision*, *recall* e *F1-score* relativamente bassi.

Durante lo svolgimento di questo progetto sono stati fatti diversi tentativi, sia di modifica ai parametri, sia andando ad intervenire direttamente sulle componenti della rete. Si è poi tentato un approccio con data augmentation, portando il dataset fino a 70’000 immagini, implementando l’aumento in fase di pre-processing, unito a tecniche di dropout per ridurre l’overfitting. Questi

accorgimenti però non hanno portato ai risultati sperati.

Qui sotto vengono riportati i grafici delle relative metriche e le conclusioni che si sono potute trarre da esse. È importante dire che questi mostrano un addestramento fissato a 100 epoch, ma che è stato interrotto alla 62-esima epoca dalla callback *early stopping*.

Cercando dei confronti con le implementazioni dello stato dell'arte, si è scoperto che una CNN basata su ResNet101 ha raggiunto un'accuratezza del **69%** su questo stesso task, suggerendo che il ViT potrebbe non essere la scelta ottimale per l'analisi delle radiografie in questo contesto. Inoltre, sempre rifacendosi allo stato dell'arte, si nota che non sono state raggiunte le performance dello Swin Transformer che aveva raggiunto un'accuratezza del 70.1% in questo task (Figura 9).

#### 4.2.1.1 Loss e accuracy

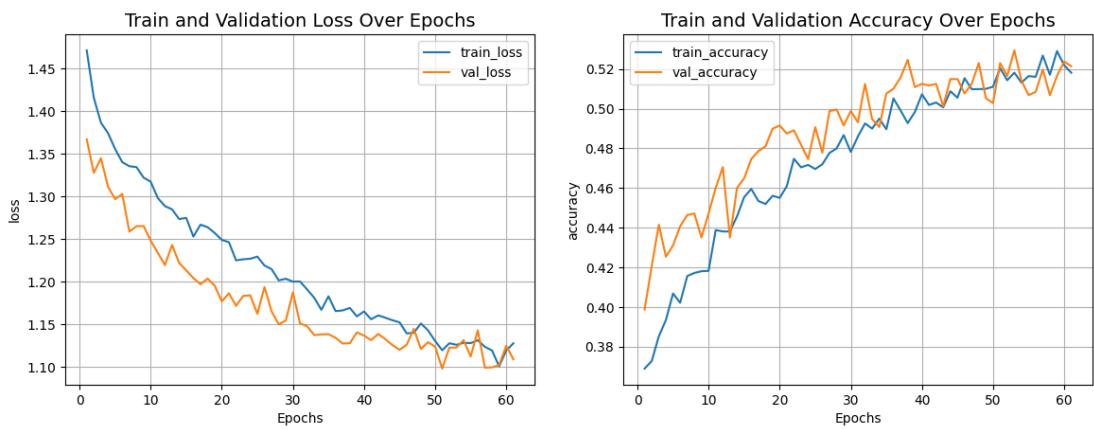


Figura 55: Loss e accuracy

Durante le prime epoch, la loss scende rapidamente, passando da un valore iniziale di 5.87 a 1.24 alla ventesima epoca. Un discorso analogo lo si può fare anche per la validation loss che ha valori che si stabilizzano intorno a 1.15 dopo circa 50 epoch.

L'incremento dell'accuratezza su entrambi i set di dati suggerisce che il modello stia migliorando nella classificazione delle immagini. Tuttavia, la vicinanza dei valori di accuratezza di addestramento e validazione indica che la rete potrebbe essere vicina alla sua **capacità massima di apprendimento con i dati disponibili**.

Come si vede dal grafico, l'incremento dell'accuratezza su entrambi i set di dati ci fa capire che il modello ha migliorato via via nella classificazione delle immagini.

Questa metrica raggiunge, però, solo il **51.58%**, suggerendo come anche in questo caso il modello potrebbe essere vicino alla sua capacità massima di apprendimento.

L'uso di tecniche di **data modification on-the-fly**, come la rotazione casuale (*RandomRotation*) e lo zoom casuale (*RandomZoom*), ha aiutato a prevenire l'overfitting e migliorare la capacità di generalizzazione del modello. Inoltre, la normalizzazione (*Normalization*) e il ridimensionamento delle immagini (*Resizing*) hanno garantito che tutte fossero standardizzate, migliorando ulteriormente la convergenza. Nonostante questi sforzi, l'accuratezza finale del modello suggerisce che potrebbero essere **necessari ulteriori dati per migliorare le performance**.

#### 4.2.1.2 Precision, Recall e F1-score

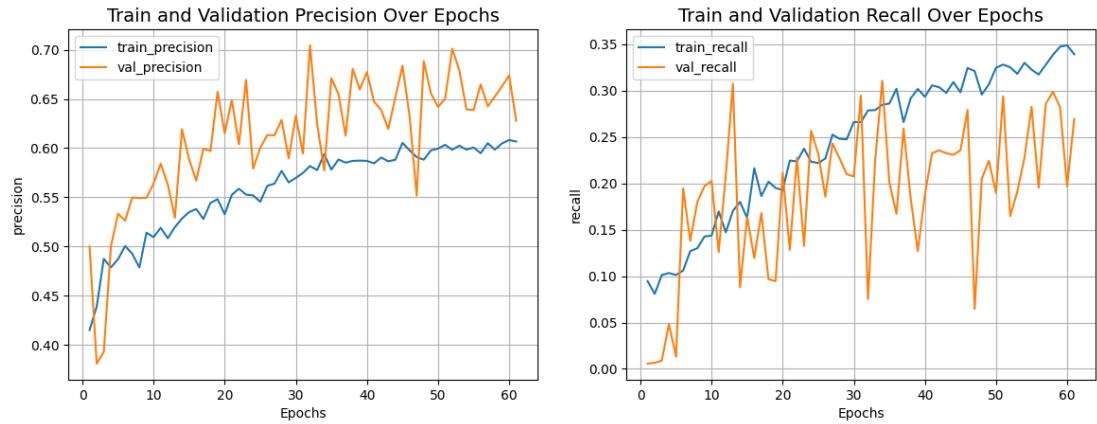


Figura 56: Precision e Recall

Durante l'addestramento del modello, la precisione ha mostrato un miglioramento costante. Come evidenziato nel grafico, questa metrica segue una tendenza crescente ma, soprattutto nella seconda, con una notevole variabilità nella precisione di validazione. La precisione iniziale di addestramento era 0.28, crescendo fino a un valore finale di test di 0.66. Questo fa intuire che il modello migliora nel minimizzare i falsi positivi.

Nonostante ciò, l'andamento a dente di sega delle curve di precisione suggerisce problemi di stabilità e indica che la rete potrebbe beneficiare di una migliore calibrazione e di ulteriori interventi per ridurre le oscillazioni nei risultati. L'uso delle callback `ModelCheckpoint` e `EarlyStopping` ha tentato di migliorare il livello di precisione, anche qui possiamo dire che sarebbe possibile attuare ulteriori miglioramenti.

La recall ha avuto un incremento durante l'addestramento, con il valore iniziale di 0.20 che è cresciuto fino a circa 0.35 per l'addestramento e 0.27 per la validazione.

Il valore finale di recall sul test è stato di 0.18, indicando difficoltà nel rilevare tutte le clas-

si positive. La variabilità e le oscillazioni nei valori di recall possono essere sintomo di problemi di generalizzazione e di instabilità nelle performance del modello su diverse epoche.

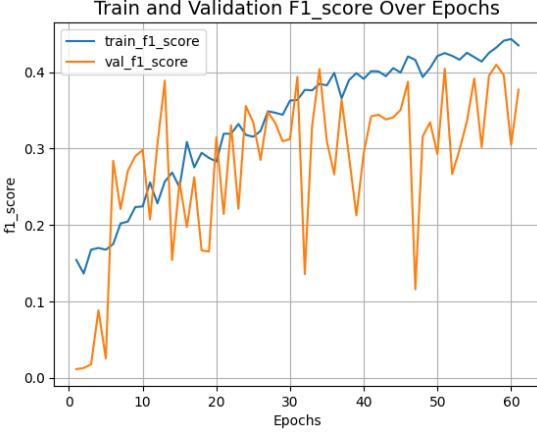


Figura 57: F1-score

L’F1-score presenta molte fluttuazioni, facendo pensare che il modello ha difficoltà a generalizzare bene su dati non visti. Infatti guardando i risultati di test relativi a questa metrika, pari a 0.28, si capisce che il modello ha ancora margine di miglioramento nel bilanciare precision e recall. Nonostante il miglioramento progressivo durante l’addestramento, il valore finale suggerisce che il modello non riesce ancora a bilanciare efficacemente le due metriche. L’utilizzo di tecniche come il bilanciamento dei dati e l’ottimizzazione delle soglie di decisione potrebbe contribuire a migliorare ulteriormente l’F1-score, portando a predizioni più accurate e bilanciate.

Il risultato ottenuto risulta più basso di quello dello stato dell’arte, relativo però a Swin Transformer, che era di 0.67 (Figura 9).

#### 4.2.1.3 AUC

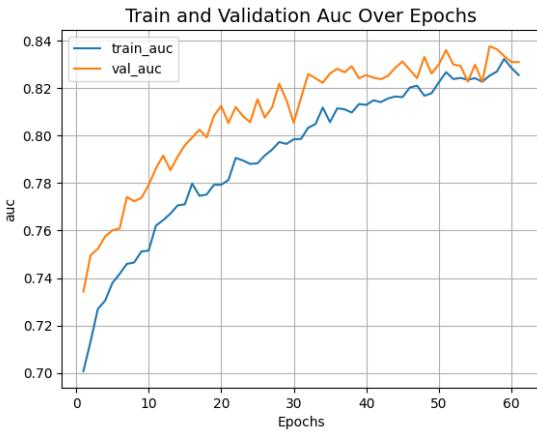


Figura 58: Area Under Curve

L'AUC (Area Under the Curve) è una metrica che misura la capacità del modello di distinguere tra le classi, con valori più alti che indicano una migliore discriminazione. Durante l'addestramento, l'AUC è aumentata costantemente, partendo da 0.58 e raggiungendo circa 0.82 per l'addestramento e 0.83 per la validazione, come evidenziato dai grafici. Questo indica un miglioramento nella capacità del modello di distinguere correttamente tra le classi. L'AUC finale del test è 0.82, denotando che il modello ha una buona capacità di discriminazione tra le classi.

Nel contesto del nostro problema a 5 classi, l'AUC viene tipicamente calcolato utilizzando un approccio *one-vs-rest*. Questo metodo implica la creazione di curve ROC per ciascuna classe trattando quella classe come positiva e tutte le altre come negative. Successivamente, si calcola l'AUC per ciascuna di queste curve e si fa una media per ottenere un valore complessivo. Nonostante il valore elevato, infatti, l'accuratezza finale del modello è rimasta bassa, attorno al 51%. Questo apparente paradosso può essere spiegato dalla natura della metrica AUC che valuta la capacità del modello di ordinare correttamente le classi piuttosto che fare predizioni esatte.

Il fatto che la metrica in esame sia elevata mentre l'accuratezza è relativamente bassa ci può far capire che, sebbene il modello sia in grado di distinguere bene tra le classi, potrebbe non essere sufficientemente preciso nelle sue predizioni finali per ognuna di esse. Questo può accadere quando il modello ha una buona capacità di riconoscere la classe corretta tra le alternative, ma le sue predizioni non sono abbastanza precise per massimizzare l'accuratezza. In altre parole, il modello può essere bravo a ordinare correttamente le classi in termini di probabilità, ma non altrettanto bravo a fare la scelta giusta in modo netto.

#### 4.2.1.4 Confusion matrix

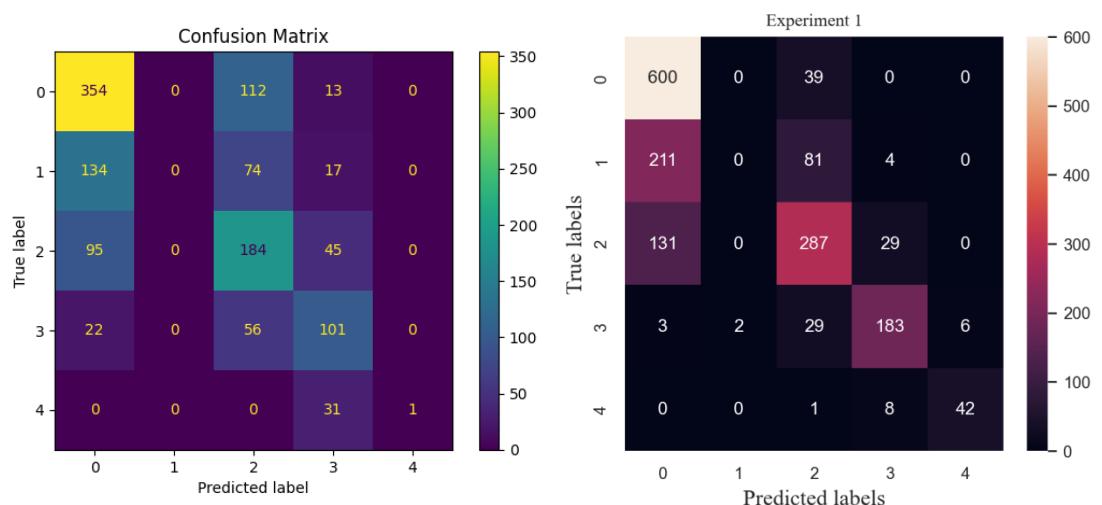


Figura 59: Differenza tra confusion matrix ottenuta e quella dello stato dell'arte

Osservando la confusion matrix, possiamo dedurre che il modello ha una buona capacità di riconoscere la classe "0", ma ha notevoli difficoltà con le altre classi. In particolare, la confusion matrix risulta essere molto simile a quella ottenuta nello stato dell'arte, seppur in quel caso sia stato usato uno Swin Transformer. Inoltre, è importante notare come entrambi i modelli fatichino ad individuare le classi "1" e "4". Ciò può essere dovuto a vari fattori, tra cui la somiglianza delle caratteristiche delle classi "1" e "0", e una mancanza di dati per quanto riguarda la classe "4".

Riguardo a quest'ultima supposizione, si è tentato di utilizzare tecniche di sovra-campionamento e bilanciamento dei dati. Questo però, seppur migliorando leggermente la confusion matrix, ha peggiorato l'accuratezza, portandola al 48%. Per questo motivo si è deciso di non mantenere il sovra-campionamento.

#### 4.2.1.5 Grad-CAM

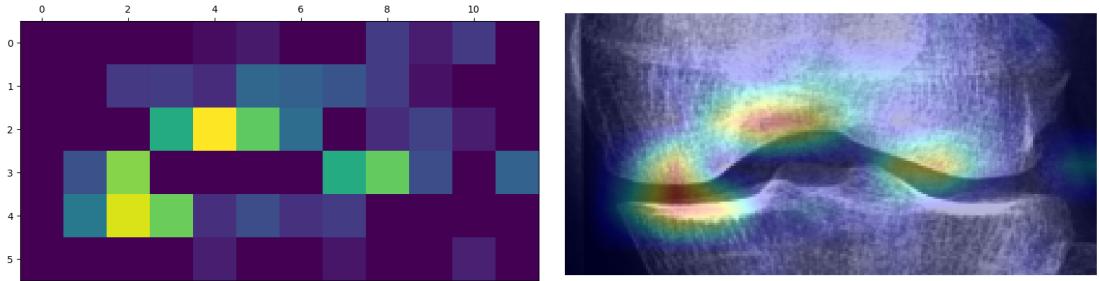


Figura 60: Output della Grad-CAM

Analizzando l'output del modello con la Grad-CAM, riusciamo a comprendere quali aree dell'immagine di input sono più importanti per la predizione del modello.

Nell'analisi si nota che i colori più caldi (rosso e giallo) indicano le zone di maggiore importanza. Queste aree si concentrano principalmente intorno alle articolazioni del ginocchio, che è esattamente dove ci si aspetterebbe che il modello trovi le informazioni più critiche per determinare la necessità di una protesi o la gravità dell'artrosi.

La Grad-CAM ottenuta risulta essere molto simile a quella ottenuta dallo stato dell'arte (Figura 11), dove le zone rilevate sono molto simili.

#### 4.2.2 Classificazione Binaria

In questa sezione, come citato già nella progettazione, si è passati ad un nuovo dataset composto da 320.000 immagini sintetiche. La rete ha quindi potuto sfruttare un dataset bilanciato di 160.000 immagini per la classe 0 (gradi KL 0 e 1), e di 160'000 immagini per la classe 1 (gradi

KL 2, 3 e 4). Data l'enorme dimensione di questo dataset sono bastate 3/4 epoche per ottenere risultati **molto significativi**.

```
Test accuracy: 99.33%
Test AUC: 99.98%
Test Precision: 98.91%
Test Recall: 99.75%
Test F1 Score: 99.33%
Epoch 1
- train_loss: 0.36729660630226135
- val_loss: 0.07115290313959122
- train_accuracy: 0.8441205620765686
- val_accuracy: 0.9806567430496216
Epoch 2
- train_loss: 0.1347983479499817
- val_loss: 0.0366654098033905
- train_accuracy: 0.9486250281333923
- val_accuracy: 0.9888870120048523
Epoch 3
- train_loss: 0.09345336258411407
- val_loss: 0.020894426852464676
- train_accuracy: 0.9654687643051147
- val_accuracy: 0.9930857419967651
```

Figura 61: Risultati della classificazione binaria

#### 4.2.2.1 Loss e accuracy

La metrica di loss ha mostrato una significativa riduzione. In particolare, la loss di addestramento è scesa da un valore iniziale di 0.37 a 0.09 alla fine dell'ultima epoca. La loss di validazione ha seguito una tendenza simile, diminuendo da 0.07 a 0.02. L'utilizzo di un dataset molto più grande, ha sicuramente contribuito a ottenere questo rapido calo.

L'accuratezza del modello binario ha mostrato miglioramenti sostanziali durante l'addestramento. Partendo da un'accuratezza iniziale di 0.84, si è raggiunta un'accuratezza finale di 0.97 sul training set. L'accuratezza di validazione è passata da 0.98 a 0.99. Il test finale ha riportato un'accuratezza del 99.33%, indicando che il modello è altamente efficace nel distinguere le due classi. Questo risultato è significativamente migliore rispetto al modello multi-classe, dove l'accuratezza finale era attorno al 51%, dimostrando che la semplificazione del problema in una classificazione binaria, unita all'uso di un dataset di dimensione maggiore, ha portato ad ottenere una migliore performance.

L'elevata accuratezza può potenzialmente significare che il modello potrebbe essere testato in ambito clinico per aiutare i chirurghi a prendere decisioni più informate, riducendo il rischio di errore umano.

#### 4.2.2.2 Precision, Recall e F1-Score

La precisione e la recall del modello binario hanno mostrato risultati impressionanti. Alla fine dell'addestramento, la precisione di test è stata del 98.91% e il recall del 99.75%, con un F1-score

di 99.33%. Questi valori indicano che il modello è molto accurato nel predire le classi positive e ha una capacità quasi perfetta di identificare tutte le istanze positive.

Un valore di F1-Score elevato come quello ottenuto indica che il modello è ben bilanciato nel predire le classi positive senza trascurare nessuna istanza.

Anche questo risultato è molto superiore rispetto al modello multi-classe, dove la metrica era significativamente più bassa, riflettendo la difficoltà di bilanciare precisione e recall in un contesto con più classi.

#### 4.2.2.3 AUC

L'AUC è arrivata ad un valore finale eccezionale di 0.9998, sia per l'addestramento che per il test. Questo valore elevato indica che la rete ha una capacità quasi perfetta di distinguere tra le classi positive e negative.

#### 4.2.2.4 Confusion matrix

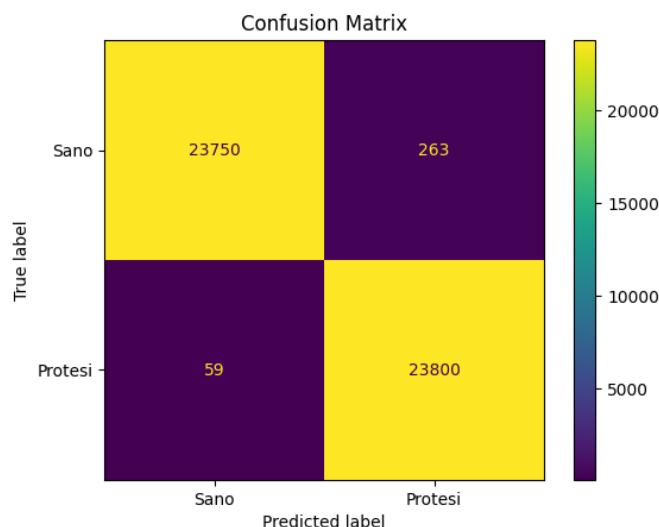


Figura 62: Confusion matrix

La confusion matrix del modello binario mostra una chiara capacità di distinguere tra le classi. Con 23750 predizioni corrette per la classe *Sano* e 23800 per la classe *Protesi*, il modello ha commesso pochissimi errori, con soli 263 falsi positivi e 59 falsi negativi. Questo risultato sottolinea l'efficacia del modello nel compito di classificazione binaria.

Nonostante nel modello multi-classe, come detto, si aveva tentato di migliorare i risultati mediante tecniche di sovraccampionamento, abbiamo visto che queste non hanno mostrato miglioramenti

significativi, qui invece, con che la semplificazione del problema a due classi e l'utilizzo di un dataset molto più ampio si sono ottenute performance molto superiori.

#### 4.2.2.5 Grad-CAM

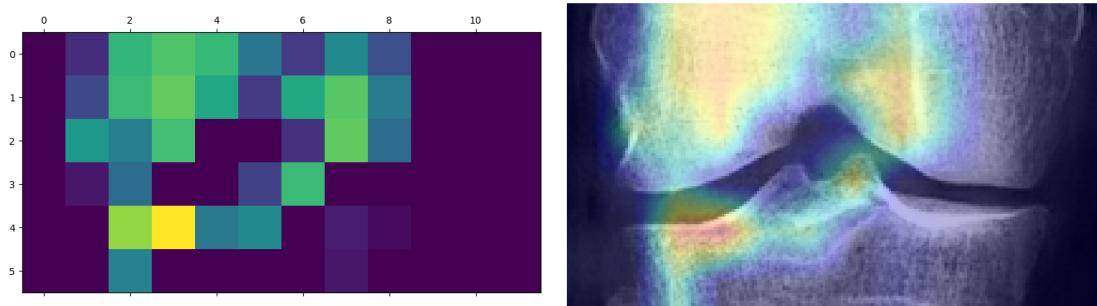


Figura 63: Output della Grad-CAM

Nella Grad-CAM sovrapposta alla radiografia, si possono osservare chiaramente le aree che il modello considera più rilevanti per la sua decisione.

Si può notare che le zone evidenziate sono principalmente localizzate intorno alle articolazioni e ai bordi ossei, suggerendo che il modello identifica segni di usura della cartilagine e deformità ossee come criteri fondamentali per la diagnosi.

Il fatto che queste zone siano simili su diverse immagini rinforza la fiducia nella capacità del modello di generalizzare le sue decisioni cliniche. Ciò significa che il modello non solo è accurato, ma è anche robusto nel riconoscere i segni distintivi che indicano la necessità di una protesi.

## 4.3 WGAN-GP

### 4.3.1 Potenziali miglioramenti e limiti

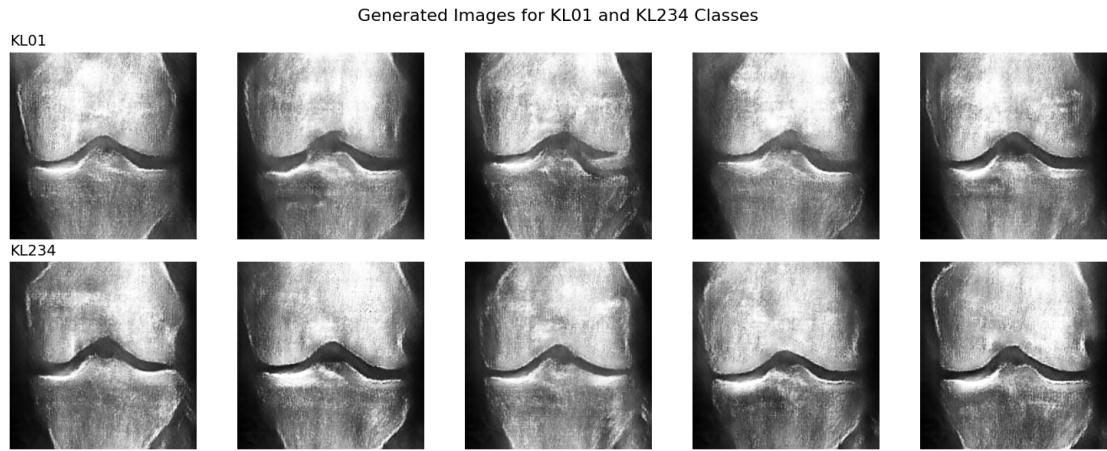


Figura 64: Esempi di immagini generate dalla WGAN-GP per le due classi.

Le immagini ottenute sono in generale di buona qualità, nonostante si possa notare dai risultati che la rete ha riscontrato maggiori difficoltà nella generazione di immagini di classe KL234. Questo è dovuto al fatto che queste immagini presentavano maggiori differenze tra di loro, avendo *più cause diverse* che determinano l’alterazione del ginocchio rispetto a uno sano, e anche perché erano in numero **minore**. Inoltre, le immagini di classe 2 si discostano significativamente da quelle di classe 4: nel primo caso, le immagini possono essere definite sane con qualche imperfezione, mentre quelle di grado 4 presentano **deformazioni importanti**. Questa grande differenza, unita al fatto che le immagini di grado 4 fossero davvero poche, ha portato la rete a non prendere quasi in considerazione deformazioni così gravi. Di conseguenza, le immagini di grado KL234 rappresentano maggiormente quelle di classe 2 e risultano quindi **più simili** alle immagini generate di classe KL01.

Inoltre, il modello ha eseguito relativamente poche epoche di allenamento (soprattutto rispetto allo stato dell’arte dove il numero di epoche era 1000), il che ha portato alla generazione, ogni tanto, di alcune immagini **completamente errate**.

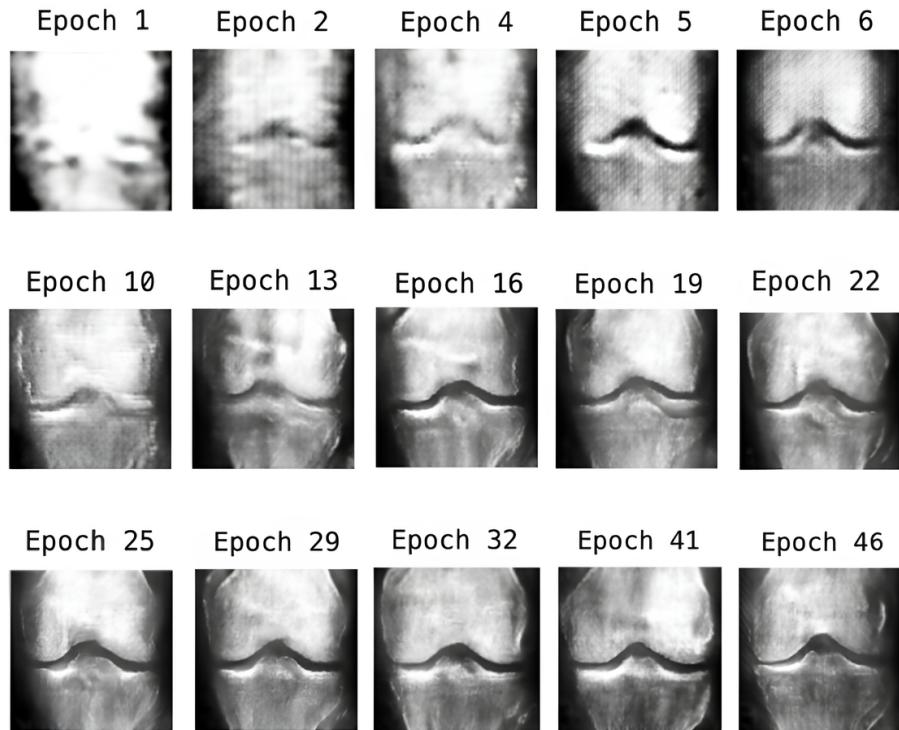


Figura 65: Immagini generate durante le epoche.

È importante notare che la qualità delle immagini migliora **significativamente** e in modo abbastanza *graduale* nel tempo. All'inizio, con learning rate **più alti**, il miglioramento è più *pronunciato* per poi stabilizzarsi lentamente.

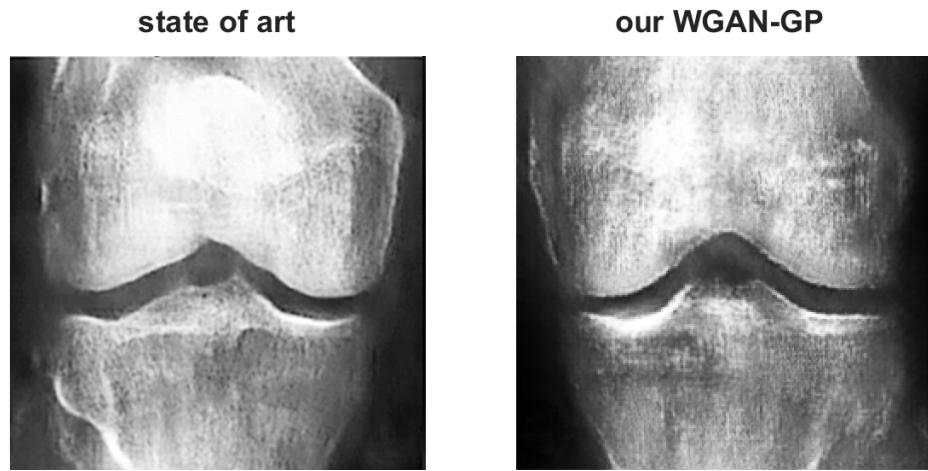


Figura 66: Esempi di immagini generate dal paper [12] considerato come stato dell'arte e dalla WGAN-GP di questo progetto a parità di epoche (34).

Il team non ha superato lo stato dell'arte principalmente a causa dell'impossibilità di eseguire un numero sufficiente di epoche. Questo si può notare in figura 66 dove le immagini generate sono addirittura superiori a parità di epoche rispetto allo stato dell'arte. Si sarebbe potuto migliorare ulteriormente aumentando la complessità del modello e testando vari parametri. Ad esempio, si sarebbe potuto usare un `batch_size` di 8, ma le limitazioni della memoria RAM lo hanno impedito. Anche l'aumento del numero di dati per il training avrebbe potuto migliorare le performance ma è stata messa in primo piano la qualità rispetto alla quantità, come dimostrato con l'operazione di focus filtering che ha escluso metà delle immagini ritenute inadatte o che potevano ingannare la rete. Inoltre, un aumento significativo di immagini avrebbe reso ancora più lento l'addestramento.

Queste osservazioni suggeriscono che con un maggior numero di epoche e possibilmente più dati per la classe KL234, le prestazioni del modello potrebbero ulteriormente migliorare, portando a immagini generate di **qualità superiore**.

L'elenco completo delle epoche di addestramento è reperibile nel file esterno `training_epochs.pdf`.

#### 4.3.2 Analisi delle loss

Le curve delle loss durante l'addestramento non sono state l'indicatore principale della qualità del modello. Sebbene le loss siano importanti, nel contesto della generazione di immagini, l'aspetto visivo delle immagini stesse è spesso un indicatore più significativo.

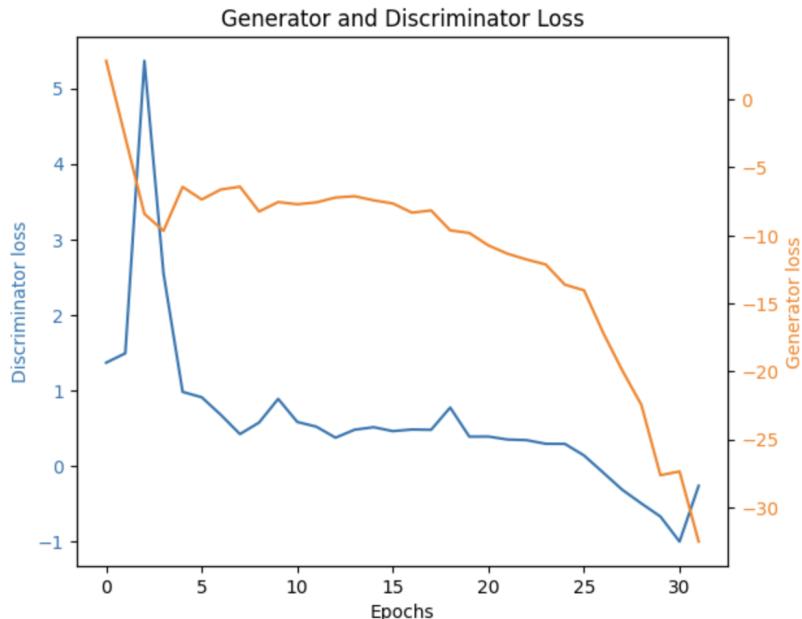


Figura 67: Andamento delle loss di generatore e discriminatore durante le epoche.

## 5 Conclusioni e considerazioni finali

A fronte di quanto era lo scopo del progetto, ovvero poter stabilire quale architettura di classificazione fosse più appropriata per l'analisi di radiografie, si è deciso di fornire di seguito quelle che sono le principali considerazioni finali:

- **ResNet50 ha prestazioni molto superiori a ViT nel problema multi-classe:** per questo tipo di applicazioni, la scelta riguardante quale metodo di intelligenza artificiale utilizzare è pressochè obbligata.
- **Nel problema binario, ViT e ResNet hanno prestazioni molto simili:** questo può portare l'operatore all'utilizzo sia di una che dell'altra per avere una conferma delle decisioni da prendere nella parte pre-intervento.
- **ViT ha un tempo di addestramento molto più rapido:** questo fattore può portare a preferire ViT nel caso del problema binario, preferendolo a ResNet se le risorse computazionali e il tempo di sviluppo sono limitate.

Modello	Classificazione	Test Accuracy	Precision	Recall	F1
<b>ResNet50</b>	Multi-classe	88.3%	0.8826	0.8823	0.8823
<b>ViT</b>	Multi-classe	51.1%	0.6446	0.1493	0.2425
<b>ResNet50</b>	Binaria	99.9%	0.9994	0.9994	0.9994
<b>ViT</b>	Binaria	99.3%	0.9891	0.9975	0.9933

Tabella 6: Comparazione finale delle performance tra ResNet50 e ViT.

Per quanto riguarda la WGAN-GP, si può affermare che il modello ha il potenziale per generare immagini affidabili se il numero di epoche viene aumentato. Questo approccio può risolvere il problema della privacy, che era l'**obiettivo principale del progetto**. In generale, questi risultati suggeriscono che l'implementazione della WGAN-GP sia stata un successo.

## Riferimenti bibliografici

- [1] Behzad Heidari. Knee osteoarthritis prevalence, risk factors, pathogenesis and features: Part i. *Caspian Journal of Internal Medicine*, 2(2):205–212, 2011.
- [2] J. H. Kellgren and J. S. Lawrence. Radiological assessment of osteo-arthrosis. *Annals of the Rheumatic Diseases*, 16(4):494–502, 1957.
- [3] Fabio Garcea, Alessio Serra, Fabrizio Lamberti, and Lia Morra. Data augmentation for medical imaging: A systematic literature review. *Computers in Biology and Medicine*, 152:106391, 2023.
- [4] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.
- [5] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning, 2013.
- [6] S. H. S. Almansour, R. Singh, S. M. H. Alyami, N. Sharma, M. S. A. Reshan, S. Gupta, and A. Shaikh. A convolution neural network design for knee osteoarthritis diagnosis using x-ray images. *International Journal of Online and Biomedical Engineering (iJOE)*, 19(07):125–141, 2023.
- [7] A. S. Mohammed, A. A. Hasanaath, G. Latif, and A. Bashar. Knee osteoarthritis detection and severity classification using residual neural networks on preprocessed x-ray images. *Diagnostics (Basel)*, 13(8):1380, Apr 2023.
- [8] S. S. Abdullah and M. P. Rajasekaran. Automatic detection and classification of knee osteoarthritis using deep learning approach. *La Radiologia medica*, 127(4):398–406, 2022.
- [9] D. R. Sarvamangala and R. V. Kulkarni. Grading of knee osteoarthritis using convolutional neural networks. *Neural Process Lett*, 53:2985–3009, 2021.
- [10] J Li, J Chen, Y Tang, C Wang, BA Landman, and SK Zhou. Transforming medical imaging with transformers? a comparative review of key properties, current progresses, and future perspectives. *Medical Image Analysis*, 85:102762, 2023. Epub 2023 Jan 31.
- [11] Aymen Sekhri, Marouane Tliba, Mohamed Amine Kerkouri, Yassine Nasser, Aladine Che-touani, Alessandro Bruno, and Rachid Jennane. Automatic diagnosis of knee osteoarthritis severity using swin transformer. *Title of Journal*, Volume Number(Issue Number):Page Range, Year.
- [12] Paloneva J. Pölönen I. et al. Prezja, F. Deepfake knee osteoarthritis x-rays from generative adversarial neural networks deceive medical experts and offer augmentation potential to automatic classification. *Sci Rep 12, 18573*, 2022.

- [13] Fabi Prezja, Juha Paloneva, Ilkka Pölönen, Esko Niinimäki, and Sami Äyrämö. Deepfake knee osteoarthritis x-rays from generative adversarial neural networks deceive medical experts and offer augmentation potential to automatic classification. *Scientific Reports*, 12:18573, 2022.
- [14] Rahul Gomes, C. Kamrowski, Jordan Langlois, Papia Rozario, Ian Dircks, Keegan Grottodden, Matthew Martinez, Wei Tee, Kyle Sargeant, Corbin LaFleur, and Mitchell Haley. A comprehensive review of machine learning used to combat covid-19. *Diagnostics*, 12:1853, 07 2022.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [16] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.
- [17] Martin Arjovsky (New York University) Vincent Dumoulin (Université de Montréal) Aaron Courville (U. Montreal) Ishaan Gulrajani (Google), Faruk Ahmed (MILA). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, 2017.
- [18] Thomas Unterthiner Sepp Hochreiter(LIT AI Lab / University Linz) Bernhard Nessler (Johannes Kepler University Linz) Martin Heusel, Hubert Ramsauer. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2018.
- [19] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 09–15 Jun 2019.
- [20] Thorsten Herrig. Fixed points and entropy of endomorphisms on simple abelian varieties. In *Journal of Algebra*, volume 517, pages Pages 95–111. ISSN 0021-8693, 2019.
- [21] Zahra Mirikhraji and Ghassan Hamarneh. Star shape prior in fully convolutional networks for skin lesion segmentation. *CoRR*, abs/1806.08437, 2018.
- [22] Jonas Adler and Sebastian Lunz. Banach wasserstein gan. In *Neural Information Processing Systems*, 2018.