



Università degli Studi di Cagliari
Facoltà di Ingegneria e Architettura
Corso di Laurea in Ingegneria Meccanica

**Studio comparativo di pipeline di deep learning per la valutazione automatica del danno
da foratura in compositi CFRP:**

Omografia cross-modale vs. analisi radiografica sequenziale

*“Confronto tra due approcci: una pipeline cross-modale teacher–student con
supervisione radiografica vs. un’analisi radiografica sequenziale”*

Relatore:
Prof. El Methedi
Corelatore:
Prof. Gian Luca MArcialis

Tesi di Laurea di:
Riccardo Venturi

A.A. 2024-2025

ABSTRACT

Questo lavoro di tesi presenta un'analisi comparativa di due pipeline di *deep learning* per la caratterizzazione automatica del danno da foratura in compositi rinforzati con fibra di carbonio (CFRP), con l'obiettivo di sostituire le metodiche manuali e soggettive di ispezione mediante un flusso computazionale riproducibile e metrologicamente coerente.

Sono stati sviluppati e valutati due sistemi end-to-end:

- (A) la pipeline *RPOS* (*Radiograph-Privileged Optical Segmentation*), un framework *teacher-student* cross-modale basato sul paradigma dell'Informazione Privilegiata, in cui dati radiografici supervisionano la segmentazione nel dominio ottico mediante registrazione omografica;
- (B) la pipeline *ROIA* (*Radiograph-Only Integrated Analysis*), interamente radiografica, composta da moduli YOLOv8 per il rilevamento, UNet++ per la segmentazione semantica, un algoritmo di estrazione feature calibrato metricamente e modelli MLP-LSTM per l'imputazione dei dati di forza e la predizione sequenziale dell'evoluzione del danno.

I risultati sperimentali, ottenuti su un dataset di circa 600 fori, mostrano che la pipeline ROIA garantisce stabilità, riproducibilità e coerenza metrologica superiori, grazie alla normalizzazione di scala e all'addestramento ibrido “foveale”. La pipeline RPOS, pur concettualmente innovativa, risulta limitata dalle distorsioni non lineari tra domini modali. Il lavoro fornisce dunque una base metodologica generalizzabile per l'automazione dei controlli non distruttivi su materiali compositi avanzati.

ABSTRACT

This thesis presents a comparative study of two *deep learning* pipelines for the automated characterization of drilling-induced damage in Carbon Fiber Reinforced Polymers (CFRP), aiming to replace manual, operator-dependent inspections with a reproducible and metrically consistent computational workflow.

Two end-to-end systems were designed and evaluated:

- (A) the *RPOS (Radiograph-Privileged Optical Segmentation)* pipeline — a cross-modal *teacher–student* framework based on the *Privileged Information* paradigm, where radiographic data supervise optical segmentation through homographic registration;
- (B) the *ROIA (Radiograph-Only Integrated Analysis)* pipeline — a fully radiograph-domain system combining YOLOv8 detection, UNet++ semantic segmentation, metrically calibrated feature extraction, and MLP–LSTM modules for force data imputation and sequential damage evolution prediction.

Experimental results on a dataset of ~600 drilled holes demonstrate that ROIA achieves superior stability, reproducibility, and metrological consistency due to scale normalization and a hybrid “foveal” training strategy. While the cross-modal RPOS approach proved conceptually innovative, it was constrained by non-linear inter-modal distortions. The proposed framework provides a foundation for automated non-destructive inspection of advanced composite materials.

Indice

INTRODUZIONE.....	1
Capitolo 1: Fondamenti Metodologici e Strumenti Computazionali.....	2
1.1 Scelta Architetturale: LSTM vs. Transformer.....	4
1.2 Architetture di Deep Learning per il Rilevamento e la Segmentazione.....	5
1.2.1 Rilevamento di Oggetti One-Shot: Architettura YOLOv8.....	5
1.2.2 Segmentazione Semantica Encoder-Decoder: Architettura UNet++.....	6
1.2.3 Modellazione Sequenziale: Architettura LSTM.....	7
Capitolo 2: Sviluppo di una Pipeline Automatizzata per l'Acquisizione e la Normalizzazione del Dataset Sperimentale.....	10
2.1. Obiettivo e Sfide dell'Acquisizione Dati.....	10
2.2. Pipeline di Acquisizione Dati: Un Approccio Modulare in Quattro Fasi.....	11
2.2.1. Fase 1: Bootstrap del Dataset tramite Computer Vision Classica.....	12
2.2.2. Fase 2: Tiling e Creazione del Dataset YOLO.....	12
2.3. Addestramento e Valutazione del Modello di Rilevamento.....	13
2.3.1. Analisi delle Performance di Addestramento.....	14
2.3.2. Valutazione Qualitativa delle Predizioni.....	15
2.4. File cucitura e ordinamento dei fori.....	18
2.5 : Ordinamento Bustrofedico tramite K-Means Clustering.....	19
2.5.1 Risultati della Pipeline di Acquisizione Dati.....	19
Capitolo 3: Pipeline A – Indagine su un Framework Cross-Modale (RPOS).....	20
3.1 Premessa: L'ipotesi dell'Informazione Privilegiata.....	20
3.2 Implementazione Tecnica: Registrazione tramite Omografia.....	21
3.3 Risultati e Analisi Critica della Pipeline RPOS.....	22
Capitolo 4: Pipeline B – Analisi Integrata su Dominio Radiografico (ROIA).....	25
4.1. Motivazione e Architettura Generale.....	25
4.2. La Criticità Nelle Scale: Normalizzazione.....	26
4.3 Generazione del Dataset di Pre-Training: La "Mask Factory" Automatica.....	28
4.4 Fase 2: Fine-Tuning sul "Golden Dataset" e Diagnosi del Plateau di Apprendimento.....	30
4.4.1 Fase 2: Fine-Tuning sul "Golden Dataset" e Diagnosi del Plateau di Apprendimento.....	31
4.5 Addestramento Ibrido "Foveale" per l'Analisi dei Dettagli.....	34
4.6 L'Algoritmo.....	34
Capitolo 5: Modulo di Estrazione Features Quantitative.....	38
Capitolo 6: Modellazione Predittiva dell'Evoluzione del Danno.....	39
6.1. Imputazione dei Dati di Forza Mancanti: un'Analisi Diagnostica.....	39

Conclusioni e Sviluppi Futuri.....	44
Considerazioni Critiche e Limiti.....	45
Appendice.....	46
Guida alla Struttura del Repository.....	50
Bibliografia Essenziale.....	51

INTRODUZIONE

Questo lavoro di tesi affronta il problema critico della quantificazione automatizzata del danno indotto dalla foratura (*drilling-induced damage*) nei materiali compositi a fibra di carbonio (CFRP), un ambito di fondamentale importanza nei settori ad alta tecnologia quali l'aerospaziale e l'automotive ****. Le lavorazioni meccaniche su questi materiali eterogenei e anisotropi introducono inevitabilmente difetti sub-superficiali, in particolare la delaminazione, che agiscono come concentratori di stress e possono compromettere drasticamente l'integrità strutturale e la vita a fatica del componente. L'ispezione tradizionale, spesso basata su metodi visivi o analisi manuali di immagini da Controlli Non Distruttivi (NDT), è soggettiva, a bassa riproducibilità, e inadeguata per caratterizzare in modo robusto le complesse morfologie del danno fibroso.

Precedenti studi in questo ambito hanno stabilito una correlazione diretta tra i parametri di processo, come la forza di foratura (Forza_N), e la severità del danno. Basandosi su un dataset sperimentale di circa 600 fori sequenziali, è stata fornita una preziosa base di dati di processo e una prima valutazione geometrica manuale del danno. Tuttavia, tale approccio ha evidenziato limiti significativi in termini di consistenza metrologica, lasciando aperto il problema di una caratterizzazione rapida e affidabile su larga scala.

L'obiettivo di questo lavoro di tesi è lo sviluppo e la valutazione comparativa di due **pipeline computazionali end-to-end**, entrambe innovative, assimilabili a un workflow MLOps (Machine Learning Operations), per la caratterizzazione automatica e la predizione del danno da foratura. Data la natura multimodale dei dati a disposizione, sono stati investigati due percorsi strategici:

1. **Pipeline A - RPOS (Radiograph-Privileged Optical Segmentation):** Un framework *cross-modal* di tipo *teacher-student*, che sfrutta le scansioni radiografiche come "informazione privilegiata" (*Privileged Information*) [1] per addestrare un modello di segmentazione che operi unicamente su immagini ottiche, allineate tramite una trasformazione omografica.
2. **Pipeline B - ROIA (Radiograph-Only Integrated Analysis):** Una architettura di sistema integrata, che opera esclusivamente sul dominio radiografico e concatena il precedente modulo Yolo con altri moduli avanzati: **(a)** una rete **UNet++** per la segmentazione, la cui performance è stata massimizzata tramite una strategia di **training ibrido "foveale"**; **(b)** un algoritmo di estrazione feature con **auto-calibrazione interna**, che garantisce la robustezza metrologica; e **(c)** una rete **MLP** per compensare i buchi nei dati di forza applicata, insieme a una **LSTM** per la modellazione predittiva del danno con MSE loss function.

I risultati di questa tesi contribuiscono al settore con due architetture di pipeline distinte, fornendo un'analisi critica basata sui dati delle sfide e dei vantaggi di entrambi gli approcci in un contesto industriale.

Per l'implementazione degli algoritmi di misura e l'analisi quantitativa dei dati, sono state utilizzate librerie fondamentali dell'ecosistema scientifico Python, tra cui **NumPy** per la manipolazione degli array [2] e **Matplotlib** per la visualizzazione dei risultati [3].

Capitolo 1: Fondamenti Metodologici e Strumenti Computazionali

L'analisi automatizzata del danno in materiali compositi richiede l'orchestrazione di diverse tecniche di Computer Vision. Questo capitolo introduce i fondamenti computazionali alla base degli strumenti selezionati per le pipeline sperimentali, giustificandone la scelta in relazione alle specifiche sfide del problema: dalla localizzazione dei fori alla segmentazione di precisione del danno, fino alla modellazione predittiva della sua evoluzione. Essendo presente ampio repertorio sulle fondamenta matematiche dei modelli e sulle definizioni di segmentazione e rilevamento di oggetti, si è preferito in questo capitolo concentrarsi sulle sfide ingegneristiche e sulla funzionalità di ogni componente.

La caratterizzazione del danno da foratura in campioni di grandi dimensioni, come le scansioni complete analizzate in questo studio, presenta una sfida computazionale significativa. Un approccio ingenuo basato sulla segmentazione semantica dell'intera immagine ad alta risoluzione (es. $> 5000 \times 9000$ pixel) sarebbe proibitivo in termini di memoria GPU e tempo di elaborazione. Per superare questo limite, entrambe le pipeline investigate in questa tesi, la RPOS (Pipeline A) e la ROIA (Pipeline B), adottano una strategia gerarchica e computazionalmente efficiente che orchestra due distinti compiti di computer vision in sequenza, object detection e segmentation.

Il primo stadio di entrambe le pipeline impiega un modello di **Object Detection**, nello specifico una rete YOLO, per eseguire una rapida ed efficiente localizzazione dei fori all'interno della scansione completa individuando le ROI(Region of interest). L'obiettivo di questa fase non è la precisione al pixel, ma l'identificazione di "regioni di interesse" (*Regions of Interest*), definite tramite *bounding boxes*, che isolano ciascun foro dal resto del pannello. Questo approccio riduce drasticamente il carico computazionale, permettendo agli stadi successivi di operare non sull'intera immagine, ma su patch a dimensione fissa (512x512 pixel) molto più gestibili.

Il secondo stadio affronta il compito di analisi fine all'interno di ciascuna patch estratta, passando dal rilevamento alla **segmentazione semantica**. In questa fase, un'architettura UNet++ viene utilizzata per classificare ogni singolo pixel, assegnandolo a una delle tre classi predefinite: sfondo, foro o danno. L'output di questo stadio è una maschera di segmentazione dettagliata, la quale costituisce la base per tutte le successive analisi quantitative.

La differenziazione strategica tra le due pipeline si manifesta nel modo in cui questo stadio di segmentazione viene sfruttato. Nella **Pipeline A (RPOS)**, si è tentato un approccio *cross-modal*, utilizzando le maschere di segmentazione ad alta fedeltà ottenute dal dominio radiografico come *ground-truth* "privilegiato" per addestrare il modello a segmentare il danno

direttamente dalle immagini ottiche, previa registrazione omografica. Nella **Pipeline B (ROIA)**, invece, si è adottato un approccio *single-modality* più diretto, utilizzando le maschere radiografiche stesse come base per l'estrazione di un ricco set di feature ingegneristiche, destinate all'analisi predittiva sequenziale tramite modelli LSTM

Confronto tra Architetture: *One-Shot Detection* vs. *Dense Prediction*

- **YOLOv8 (Detector):** È un'architettura *one-shot* e *single-stage*. Il suo scopo è analizzare l'intera immagine in un unico passaggio ("one look") per predire un insieme **sparso** di output: le coordinate di un numero limitato di *bounding boxes* e le relative confidenze di classe. Questo la rende eccezionalmente veloce ed efficiente, poiché la complessità computazionale non dipende dal numero di pixel dell'immagine, ma dal numero di oggetti da rilevare.[4]
- **UNet++ (Segmenter):** È un'architettura *dense prediction* di tipo Encoder-Decoder. Il suo scopo è produrre un output **denso**, ovvero una maschera di segmentazione in cui **ogni singolo pixel** dell'immagine di input viene classificato. La complessità computazionale, in particolare il consumo di memoria GPU, è direttamente proporzionale al numero di pixel da processare, $O(W \times H)$. Dove W e H sono larghezza e altezza dell'immagine.[5]

Un modello UNet++ richiede che l'intera immagine e le sue mappe di feature intermedie risiedano nella memoria della GPU. Una stima approssimativa del consumo di memoria per il solo primo strato (assumendo un batch_size=1 e un numero di filtri F=64) con float 32 è:

$$Vram \approx W \times H \times (\text{Channels}_{\text{input}} + F) \times 4 \text{bytes/pixel}$$

$$Vram = 9120 \times 5408 \times (3 + 64) \times 4 \approx 13 \text{Gigabytes}$$

Le patches vengono elaborate sequenzialmente, questo vuol dire che ad ogni nuova elaborazione si libera la memoria. Con formato 512x512 di esempio avremmo:

$$Vram = 512 \times 512 \times (3 + 64) \times 4 \approx 70 \text{Mbytes}$$

Una T4 Nvidia ha potenze dell'ordine dei 16gb, rischiando di terminare la Ram di processo.

Il problema aumenterebbe esponenzialmente se si considerassero tutti i layer intermedi del modello.

La stima del consumo di VRAM deve considerare tre componenti principali: la memoria per i **parametri del modello**, quella per i **gradienti** e gli **stati dell'ottimizzatore** (entrambe proporzionali al numero di parametri), e quella per le **mappe di attivazione intermedie**, che è la componente dominante e scala con le dimensioni spaziali dell'input. Mentre i parametri del modello e gli stati dell'ottimizzatore per l'architettura scelta occupano un costo fisso e

gestibile (ordine di centinaia di MB), il requisito di memoria per le attivazioni cresce in modo quadratico. Processando un'immagine intera da 49.3 Megapixel, la memoria richiesta per le sole attivazioni viene stimata essere nell'ordine di decine di Gigabyte (≈ 47 GB), un valore che supera ampiamente le capacità di una GPU commodity (NVIDIA T4, 16 GB). Al contrario, processando una singola patch da 512x512 pixel, questo requisito scende a poche centinaia di MB. Poiché le patch vengono elaborate in modo sequenziale, il consumo di VRAM massimo in ogni istante rimane basso e gestibile, rendendo la strategia gerarchica l'unica ingegneristicamente praticabile.

Per motivazioni analoghe, legate alla scalabilità computazionale e ai requisiti di dati, si è scelto di non investigare architetture basate su **Vision Transformers (ViT)**. Sebbene rappresentino lo stato dell'arte in molti benchmark di classificazione, la loro architettura basata sui meccanismi di auto-attenzione ha una complessità che scala quadraticamente con il numero di patch di input (pixel), e notoriamente richiedono dataset di pre-training di dimensioni enormi (anche centinaia di milioni di immagini) per raggiungere performance competitive, una risorsa non disponibile per questo progetto. L'approccio basato su **CNN** rimane quindi la scelta più pragmatica ed efficace per il dominio di applicazione specifico.

1.1 Scelta Architetturale: LSTM vs. Transformer

Sebbene le architetture **Transformer** siano emerse come lo stato dell'arte in molti compiti di elaborazione del linguaggio naturale (NLP) e stiano guadagnando popolarità anche nell'analisi di serie temporali, per questo studio è stata preferita una rete **LSTM**. [3]

La motivazione è duplice e di natura prettamente ingegneristica:

Complessità Computazionale: Il meccanismo di auto-attenzione al cuore del Transformer ha una complessità computazionale e di memoria che scala **quadraticamente** con la lunghezza della sequenza di input ($n^2 * d$), dove n è la lunghezza della sequenza e d la dimensione delle feature. Al contrario, un'architettura ricorrente come la LSTM scala in modo **lineare** ($n * d^2$). Data la lunghezza delle nostre sequenze (fino a 595 fori), un'architettura Transformer sarebbe stata significativamente più esigente in termini di risorse computazionali.

Requisiti di Dati: Similmente ai Vision Transformers, i modelli Transformer per serie temporali beneficiano enormemente di dataset di grandi dimensioni per apprendere pattern complessi. Le LSTM, avendo un *bias induuttivo* più forte verso la sequenzialità (processano i dati un passo alla volta), sono spesso più efficienti e performanti in scenari *low-data*, come il nostro dataset di alcune centinaia di punti temporali.

Pertanto, l'architettura LSTM rappresenta un compromesso ottimale tra capacità espressiva e fattibilità pratica per modellare le dinamiche di usura e danneggiamento presenti nei nostri dati sperimentali.

1.2 Architetture di Deep Learning per il Rilevamento e la Segmentazione

1.2.1 Rilevamento di Oggetti One-Shot: Architettura YOLOv8

Per la localizzazione iniziale dei fori, è stata impiegata un'architettura **YOLOv8**. [4]

Appartenente alla famiglia dei *detector single-shot*, YOLO (You Only Look Once) tratta il rilevamento di oggetti come un singolo problema di regressione, direttamente dalle coordinate dei pixel dell'immagine alle coordinate delle *bounding boxes* e alle probabilità di classe.

La sua architettura è tipicamente composta da tre parti principali:

1. **Backbone:** Una rete neurale convoluzionale (nel caso di YOLOv8, una C2f basata su CSPDarknet53 [6]) che estrae una gerarchia di *feature maps* a diverse scale.
2. **Neck:** Una serie di strati (come le architetture PAN) che combinano e fondono le *feature maps* provenienti da diversi livelli del *backbone* per arricchire l'informazione semantica e spaziale a tutte le scale.
3. **Head (Testa di Rilevamento):** Un insieme di strati convoluzionali finali che operano sulle *feature maps* raffinate e producono le predizioni finali: per ogni cella della griglia in cui l'immagine viene suddivisa, la "testa" predice le coordinate (x, y, w, h) della *bounding box*, una confidenza di "oggettività" (la probabilità che quella box contenga un oggetto) e le probabilità delle classi.

L'operazione di Loss della architettura Yolo è una loss composita per object detection:

$$L_{YOLO} = \lambda_{box} L_{box} + \lambda_{obj} L_{obj} + \lambda_{cls} L_{cls}$$

L_{obj} = **loss di oggettività** (*objectness*), tipicamente una Binary Cross-Entropy (BCE), che insegna al modello a distinguere tra una regione di interesse contenente un oggetto (un foro) e lo sfondo

L_{box} = **loss di regressione**, che penalizza l'errore sulle coordinate della *bounding box*. Le implementazioni moderne usano una variante basata sulla metrica *Intersection over Union* (es. CIoU), che si è dimostrata più efficace rispetto a una semplice distanza Euclidea

L_{cls} = è la **loss di classificazione** che, nel nostro caso con una singola classe ("foro"), verifica semplicemente la presenza dell'oggetto.

La funzione di perdita è una somma pesata di tre termini: (i) un termine di regressione per le coordinate della *bounding box*, implementato mediante una loss IoU-aware (CIoU/DIoU) per migliorare la correlazione con la metrica di valutazione, (ii) un termine di *objectness* (BCE)

che discrimina background/oggetto, e (iii) un termine di classificazione (cross-entropy). L'implementazione utilizzata (Ultralytics YOLOv8) integra anche distribuzioni per la regressione del box (DFL) e moduli di focal/varifocal per gestire squilibri di classe e stabilizzare il training.

1.2.2 Segmentazione Semantica Encoder-Decoder: Architettura UNet++

L'operazione fondamentale delle due architetture Yolo e Unet++ è la convoluzione 2D così definita:

$$(I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

Dove I è l'immagine di input(o la feature map intermedia) e K è un kernel di pesi di addestramento che si comporta da filtro informativo della rete.

L'architettura UNet++ consiste in:

1. **Encoder:** Un percorso di contrazione che utilizza una sequenza di blocchi convoluzionali seguiti da operazioni di MaxPooling per ridurre la dimensione spaziale e aumentare la profondità delle *feature* (catturando il contesto). Per questa pipeline è stato scelto un encoder pre-addestrato della famiglia **EfficientNet-B0** [7]. Questa scelta rappresenta un eccellente per le richieste di VRAM contenute.
2. **Decoder:** Un percorso di espansione che utilizza convoluzioni trasposte (Transposed Convolution) o *upsampling* per aumentare la risoluzione spaziale, recuperando gradualmente le informazioni di localizzazione.
3. **Skip Connections Annidate:** La caratteristica distintiva della UNet++ è l'introduzione di strati convoluzionali intermedi lungo i percorsi delle *skip connections*. Se in una U-Net classica l'output dello strato dell'encoder $X^{(0,j)}$ viene direttamente fuso con l'input dello strato del decoder $X^{(1,j)}$, in una UNet++ ogni nodo $X^{(i,j)}$ della griglia viene calcolato come: [8]

$$X^{(i,j)} = H(\left[[X^{(i,j)}]_{k=0}^{j-1}, U(X^{(i+1)(j-1)}) \right]) \quad \text{per } i > 0$$

H è un operatore di convoluzione e U un'operazione di upsampling e [...] indica una concatenazione lungo l'asse dei canali della mia immagine.

Questa architettura densa riduce il gap semantico tra encoder e decoder, permettendo al modello di catturare dettagli più fini e di migliorare le prestazioni, specialmente su bordi complessi come quelli del danno da delaminazione.

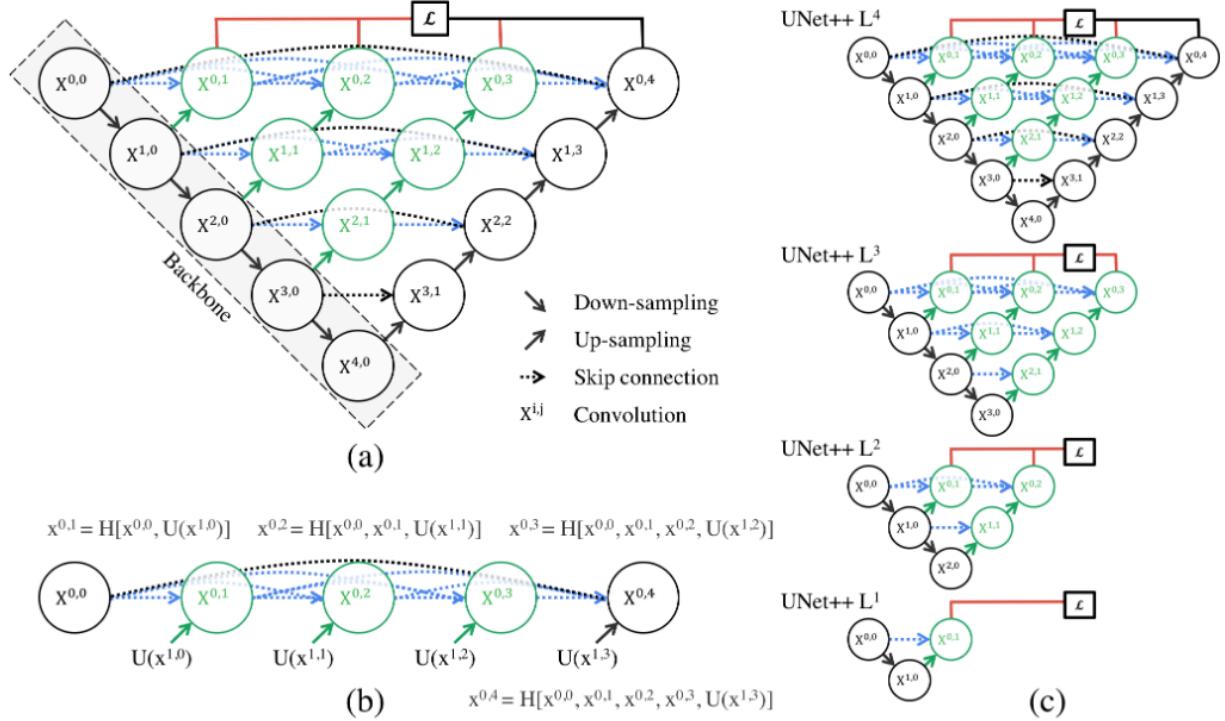


Figura 1.1 – Schema dell’architettura UNet++ (Nested U-Net) Fonte tratta da. [5]

Loss Function Unet++

Per il compito di segmentazione, dove ogni pixel deve essere classificato, è stata scelta una funzione di perdita ibrida, comunemente usata in letteratura per la sua robustezza.

$$L_{\text{Segmentazione}} = 0.5 \cdot L_{\text{BCE}} + 0.5 \cdot L_{\text{Dice}}$$

L_{BCE} = (Binary Cross-Entropy) opera a livello di singolo pixel, fornendo un segnale di gradiente stabile e preciso.

L_{Dice} = (Dice Loss) è una metrica basata sulla sovrapposizione delle aree. È particolarmente efficace in presenza di forte **sbilanciamento tra le classi** (pochi pixel di danno rispetto a un vasto sfondo) e aiuta il modello a delineare meglio i contorni degli oggetti, un aspetto cruciale per la successiva estrazione di feature metriche

1.2.3 Modellazione Sequenziale: Architettura LSTM

Per modellare l’evoluzione temporale del danno in funzione delle forze di processo, è stata impiegata una rete **Long Short-Term Memory (LSTM)**, [9] un tipo avanzato di Rete Neurale Ricorrente (RNN).

Una cella LSTM standard è governata da un insieme di equazioni che regolano il flusso di informazione attraverso tre "porte" (*gates*) principali: la *forget gate*, l'*input gate* e l'*output gate*. Dato un input x_t , lo stato nascosto precedente h_{t-1} , lo stato di cella precedente c_{t-1} , lo stato della cella c_t , il nuovo stato nascosto h_t , al tempo t:

1. **(f_t) forget gate:** Decide quale informazione dello stato di cella precedente deve essere scartata.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2. **(i_t) input gate:** Decide quali nuove informazioni devono essere memorizzate nello stato di cella

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

3. **Aggiornamento della cella di stato C(t):**

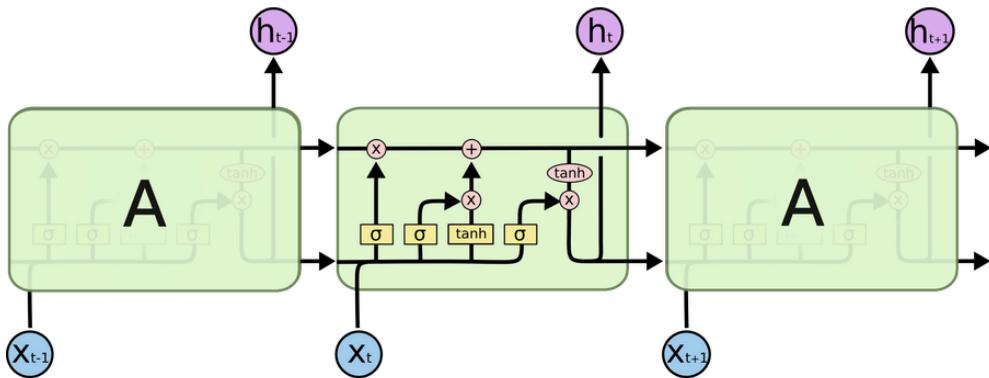
$$C_t = f_t * C_{t-1} + i_t * C$$

4. **Output gate $\sigma(t)$:** Decide quale parte dello stato di cella deve essere usata per produrre l'output

$$\sigma_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = \sigma_t * \text{atanh}(C_t)$$

Dove W è la matrice pesi, b matrice dei bias, σ è la funzione sigmoide. Questo meccanismo di *gating* permette alla LSTM di mantenere e propagare selettivamente le informazioni attraverso lunghi intervalli temporali, rendendola ideale per catturare il trend di usura graduale e la sua correlazione con la morfologia del danno nel nostro dataset sequenziale



The repeating module in an LSTM contains four interacting layers.

Figura 1.2 – Schema dell'architettura LTSM Tratto da. Olah [10]

Per superare i limiti della tradizionale *Mean Squared Error* (MSE), che tende a predire il valore medio ignorando i picchi, per la rete LSTM è stata implementata una **Quantile Loss**, nota anche come *pinball loss*.

$$L_\tau(y, \hat{y}) = \begin{cases} \tau \cdot (y - \hat{y}) & \text{se } y \geq \hat{y} \\ (\tau - 1) \cdot (y - \hat{y}) & \text{se } y < \hat{y} \end{cases}$$

Dove y è il valore reale, \hat{y} è la predizione e τ è il quantile target (es. $\tau = 0.9$ per il 90° percentile). Questa funzione di perdita permette di addestrare modelli specifici per stimare non solo la tendenza centrale (con $\tau = 0.5$), ma anche i limiti inferiori e superiori dell'intervallo di predizione, fornendo una stima del rischio molto più completa e adatta a catturare gli eventi estremi di danneggiamento

Recap:

(a) Lambda in YOLO

Nella formulazione della loss di detection i coefficienti λ sono pesi scalari utilizzati per bilanciare i termini di regressione delle bounding box L_{box} , objectness L_{obj} e classificazione L_{cls} . La scelta di questi pesi è ingegneristica e dipende dalla scala numerica delle singole componenti e dagli obiettivi sperimentali; alternative automatiche includono la stima della varianza omoschedastica (Kendall et al.) o GradNorm.

(b) UNet++ hybrid loss

Per la segmentazione con UNet++ si è adottata la loss ibrida $L = \alpha \cdot L_{BCE} + (1 - \alpha) L_{Dice}$ (tipicamente $\alpha = 0.5$). BCE fornisce stabilità

pixel-wise, Dice mitiga lo sbilanciamento foreground/background e migliora i contorni della maschera.

(c) Quantile loss per LSTM

Per la modellazione sequenziale si è impiegata la Quantile (pinball) Loss L_τ per predire quantili condizionati della variabile target (es. $\tau = 0.9$ per i picchi di forza). La quantile loss è asimmetrica ed evita l'effetto di "regressione alla media" tipico dell'MSE, fornendo stime robuste delle code della distribuzione condizionata.

Capitolo 2: Sviluppo di una Pipeline Automatizzata per l'Acquisizione e la Normalizzazione del Dataset Sperimentale

2.1. Obiettivo e Sfide dell'Acquisizione Dati

Il prerequisito fondamentale per qualsiasi analisi basata su Machine Learning è la disponibilità di un dataset vasto, coerente e correttamente indicizzato. L'obiettivo primario di questa fase del lavoro è stato lo sviluppo di una pipeline computazionale in grado di processare le scansioni grezze, sia in modalità ottica che radiografica, e generare un dataset di "patch" – immagini ritagliate 512x512 pixel – normalizzate e centrate su ogni singolo foro. Questo processo, apparentemente semplice, ha presentato sfide ingegneristiche non banali.

I dati di scansione grezzi, come esemplificato in **Figura 2.1**, erano caratterizzati da orientamenti inconsistenti dovuti a metadati EXIF variabili, un ordinamento dei fori non standard che variava da scansione a scansione, e in alcuni casi, dati corrotti o incompleti (es. scansioni divise in più file che richiedevano una "cucitura" digitale). Era quindi necessario un metodo automatico e robusto per superare queste criticità e produrre un dataset affidabile per gli stadi successivi di analisi.

Altro problema che ha portato al secondo metodo alternativo è la mancanza di scan ottiche da usare, infatti il dataset offerto era incompleto.

Altra problematica è stata l'inconsistenza delle dimensioni delle scan e dei dpi fra ottica e radio; questo ha portato nella pipeline alternativa a un problema delle scale di foro, non apparsa durante lo studio omografico.

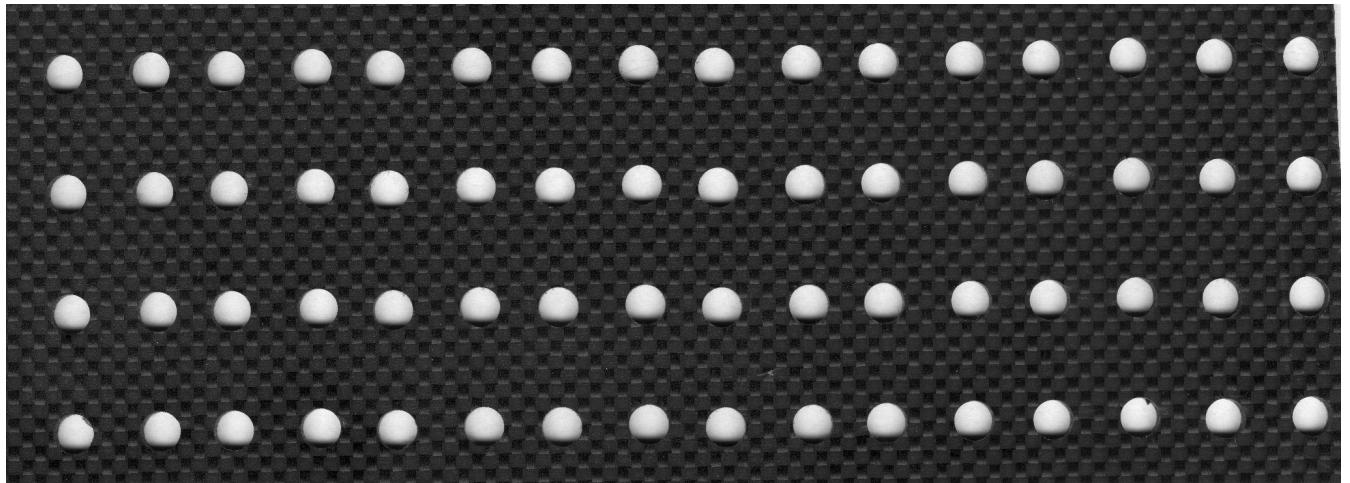


Figura 2.1 -scan ottica iniziale di esempio dei primi 64 fori in ingresso: size 9400x3400 circa

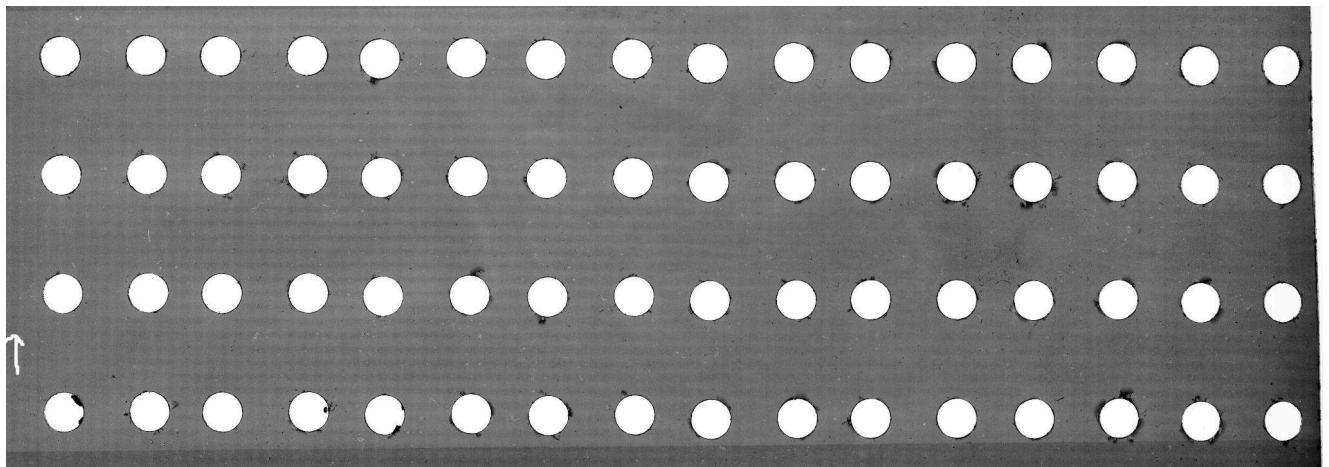


Figura 2.2 - Scan Radio. Dimensioni 4700x1660 circa

2.2. Pipeline di Acquisizione Dati: Un Approccio Modulare in Quattro Fasi

Per superare queste sfide, è stata sviluppata una pipeline sequenziale in Python(google colab), descritta di seguito. Il codice sorgente completo è disponibile nel repository GitHub del progetto.

Appendice E.1 - riferimento

Come primo passo, è stato impiegato un modello di *Object Detection* YOLOv8 per localizzare i fori all'interno delle scansioni complete. Per accelerare la creazione del dataset di training per YOLO, è stata utilizzata una tecnica di *bootstrap*: l'algoritmo cv2.HoughCircles di OpenCV è stato applicato a un'immagine campione per generare un primo set di annotazioni grezze (bounding boxes), evitando un lungo processo di etichettatura manuale. Il modello YOLOv8 nano è stato quindi addestrato su *tile* (tasselli) da 700x700 pixel estratti dalle scansioni.

L'inferenza è stata eseguita sulla risoluzione nativa delle immagini per massimizzare la precisione del rilevamento. Gli overlay visivi confermano l'efficacia del metodo, con il modello in grado di rilevare la quasi totalità dei fori con un'elevata confidenza.

2.2.1. Fase 1: Bootstrap del Dataset tramite Computer Vision Classica

Per generare il dataset di training iniziale per YOLOv8 senza un oneroso lavoro manuale, è stato adottato un approccio semi-automatico. Si è sfruttata la libreria OpenCV per creare le prime annotazioni grezze.

La **Trasformata Circolare di Hough** (cv2.HoughCircles), [11] applicata a una scansione di riferimento, ha permesso di localizzare i centri e i raggi della maggior parte dei fori. Le coordinate di questi cerchi sono state quindi convertite in *bounding boxes* e salvate come *ground-truth* iniziale per il modello YOLOv8.

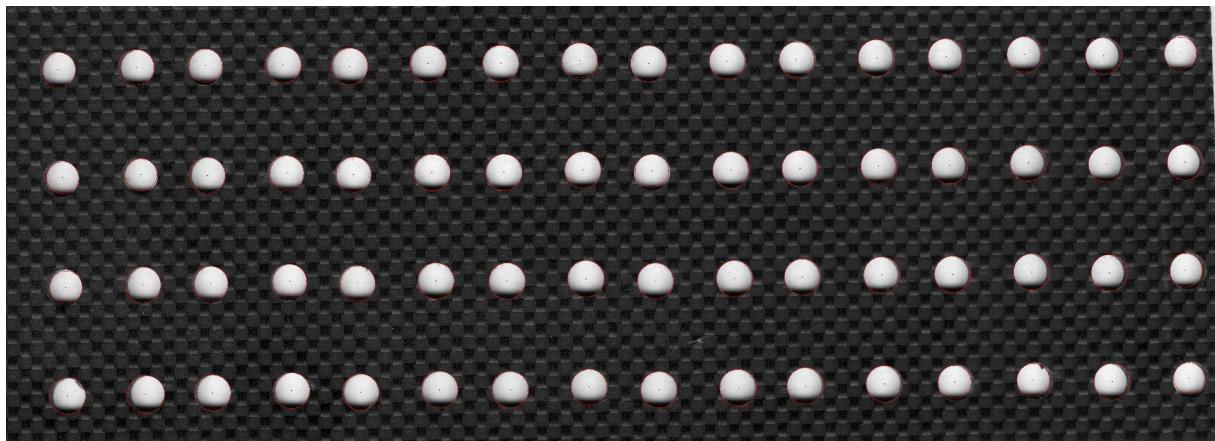


Figura 2.3 - fori segnati dall'algoritmo

2.2.2. Fase 2: Tiling e Creazione del Dataset YOLO

Yolo lavora su una sola immagine. Pertanto, la scansione completa e le *bounding boxes* globali sono state suddivise in "tile" (**tasselli**) di 704x704 pixel con una sovrapposizione del 50%. Questo approccio, noto come *tiled processing*, ha permesso di:

1. Creare un dataset di immagini di dimensioni gestibili (N campioni di 704x704).
2. Garantire, tramite l'overlap, che i fori situati ai bordi dei tasselli fossero presenti per intero in almeno un campione.
3. Generare, per ogni *tile*, un file di etichette .txt in formato YOLO, con le coordinate delle *bounding boxes* normalizzate rispetto alle dimensioni del *tile*.

La scelta dei pixel è stata motivata dal fatto che Unet lavori su multipli di 32, si è poi preferito nelle radiografie, con dpi dimezzato, diminuire le dimensioni a 512x512 sia per risparmiare memoria vram in esecuzione che per lavorare con le dimensioni standard usate nelle unet.

Il dataset di tasselli risultante è stato quindi suddiviso in set di training (80%) e validazione (20%) e configurato tramite un file data.yaml per l'addestramento con il framework Ultralytics.

2.3. Addestramento e Valutazione del Modello di Rilevamento

Il modello **YOLOv8-Nano** è stato addestrato per 50 epochhe sul dataset di tasselli (704x704 pixel) precedentemente generato, utilizzando un *batch size* di 8. L'addestramento è stato eseguito in un ambiente Google Colab equipaggiato con una GPU NVIDIA T4. Il framework Ultralytics applica di default una pipeline di *data augmentation* in tempo reale, che include trasformazioni come flip orizzontali e variazioni di colore, per migliorare la robustezza del modello.

La scomposizione delle scansioni in tasselli (tiling) è stata automatizzata per garantire la gestione della memoria VRAM; i dettagli della logica di mappatura delle coordinate globali sono consultabili nel **Listato A.1 dell'Appendice A**

Tabella 2.1 - esempio di coordinate di una box da file txt

Foro True/False	x1	y1	x2	y2
0	0.5828	0.71142	0.3857142	0.3857

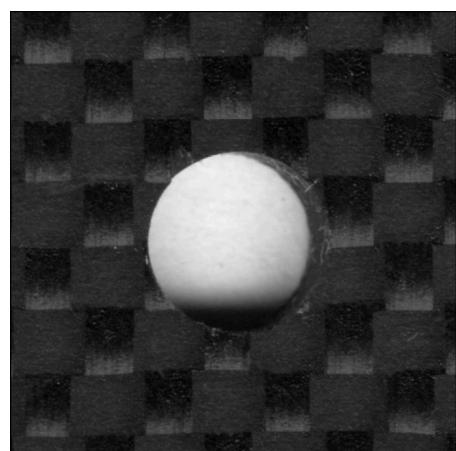
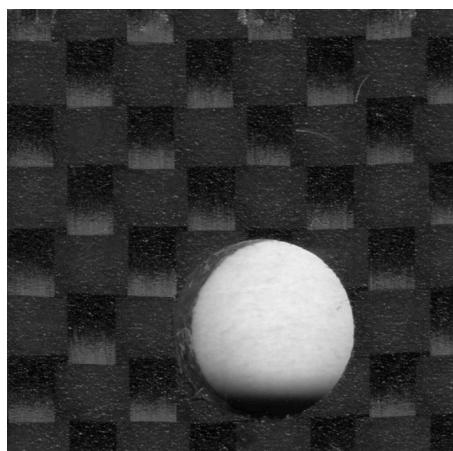


Figure 2.4 - Esempio tile ricavate dalle coordinate di un file txt

2.3.1. Analisi delle Performance di Addestramento

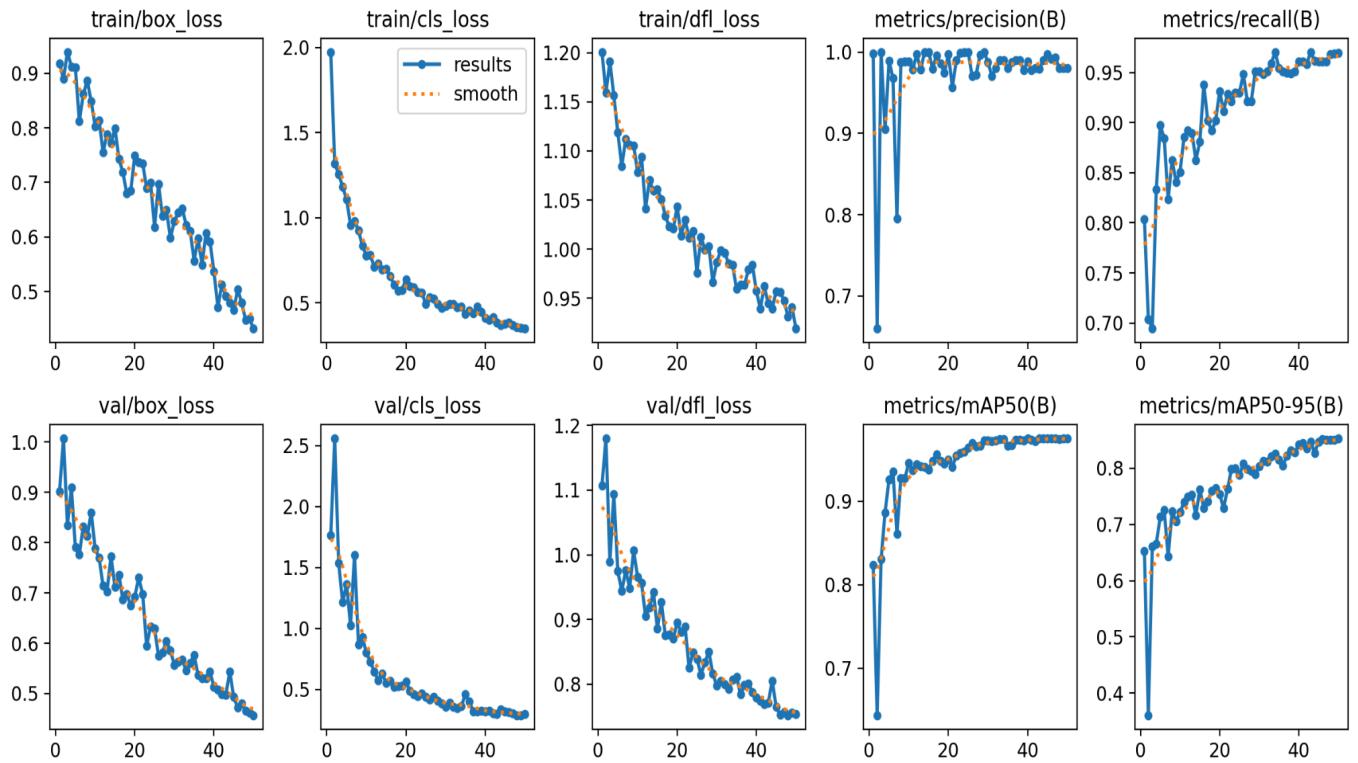


Figura 2.4.1 - Curve di apprendimento del modello YOLOv8-Nano. La prima riga mostra le metriche sul training set, la seconda sul validation set. Si osserva una convergenza stabile e una progressiva riduzione della loss (errore) sia per la localizzazione (box_loss) che per la classificazione (cls_loss), accompagnata da un costante aumento delle metriche di accuratezza (precision, recall, mAP). Si nota una convergenza

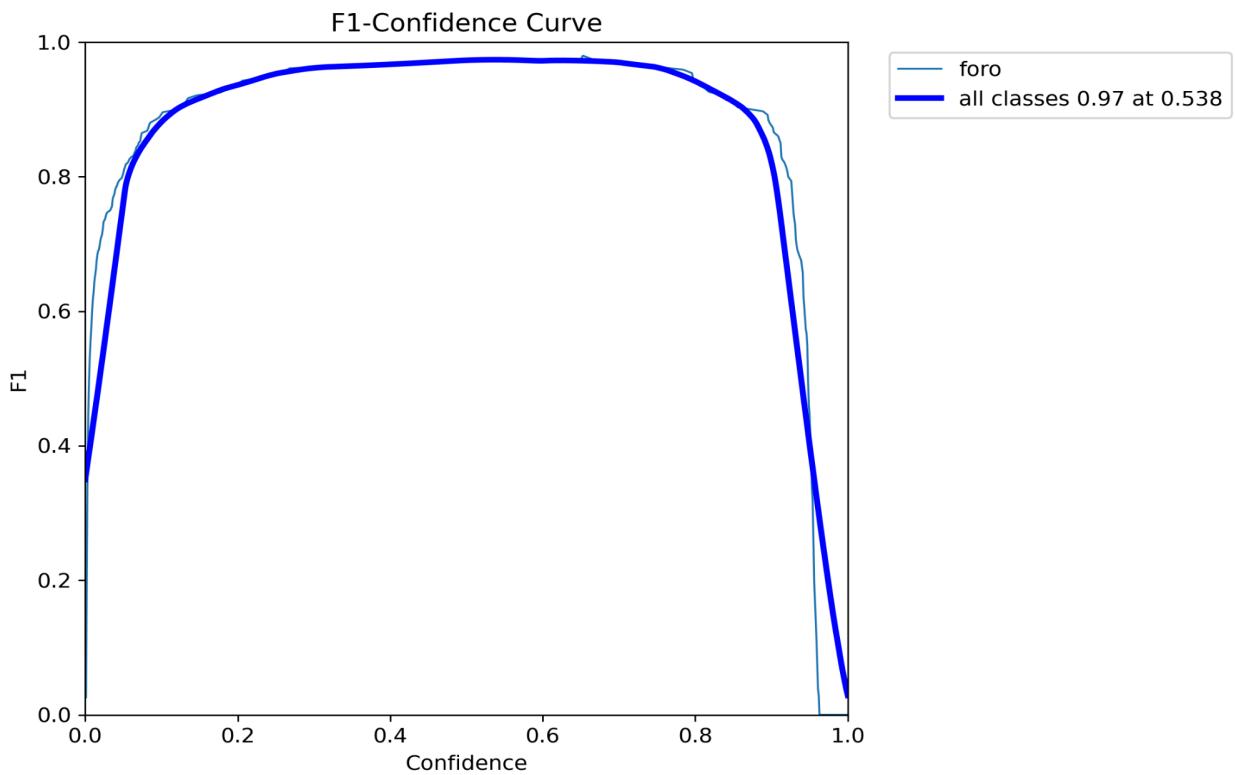


Figura 2.4.2 - In particolare, la curva **F1-Confidence** (Figura 2.Y) mostra che il modello raggiunge il suo picco di performance (un F1-Score di 0.97) a una soglia di confidenza di 0.538. Questo dato è fondamentale in quanto conferma la validità della soglia di confidenza utilizzata durante l'inferenza ($\text{conf}=0.79$), che si posiziona in una regione di alta precisione e alto recall, minimizzando sia i falsi positivi che i falsi negativi

2.3.2. Valutazione Qualitativa delle Predizioni

Oltre all'analisi quantitativa, è stata condotta una valutazione qualitativa per ispezionare visivamente il comportamento del modello sui dati di validazione. Come mostrato in Figura 2.Z, il modello dimostra un'elevata robustezza, identificando correttamente i fori anche in condizioni difficili, come quelli parzialmente occlusi ai bordi dei tasselli (un risultato diretto dell'uso dell'overlap durante la creazione del dataset). La confidenza delle predizioni si attesta costantemente tra 0.8 e 1.0, confermando la solidità del rilevatore.

I log di esecuzione hanno confermato l'efficacia del metodo, con una stima di 86 fori su 84 attesi nella Scansione 5 e 133 su 132 nella Scansione 4, indicando una minima e accettabile sovrastima.

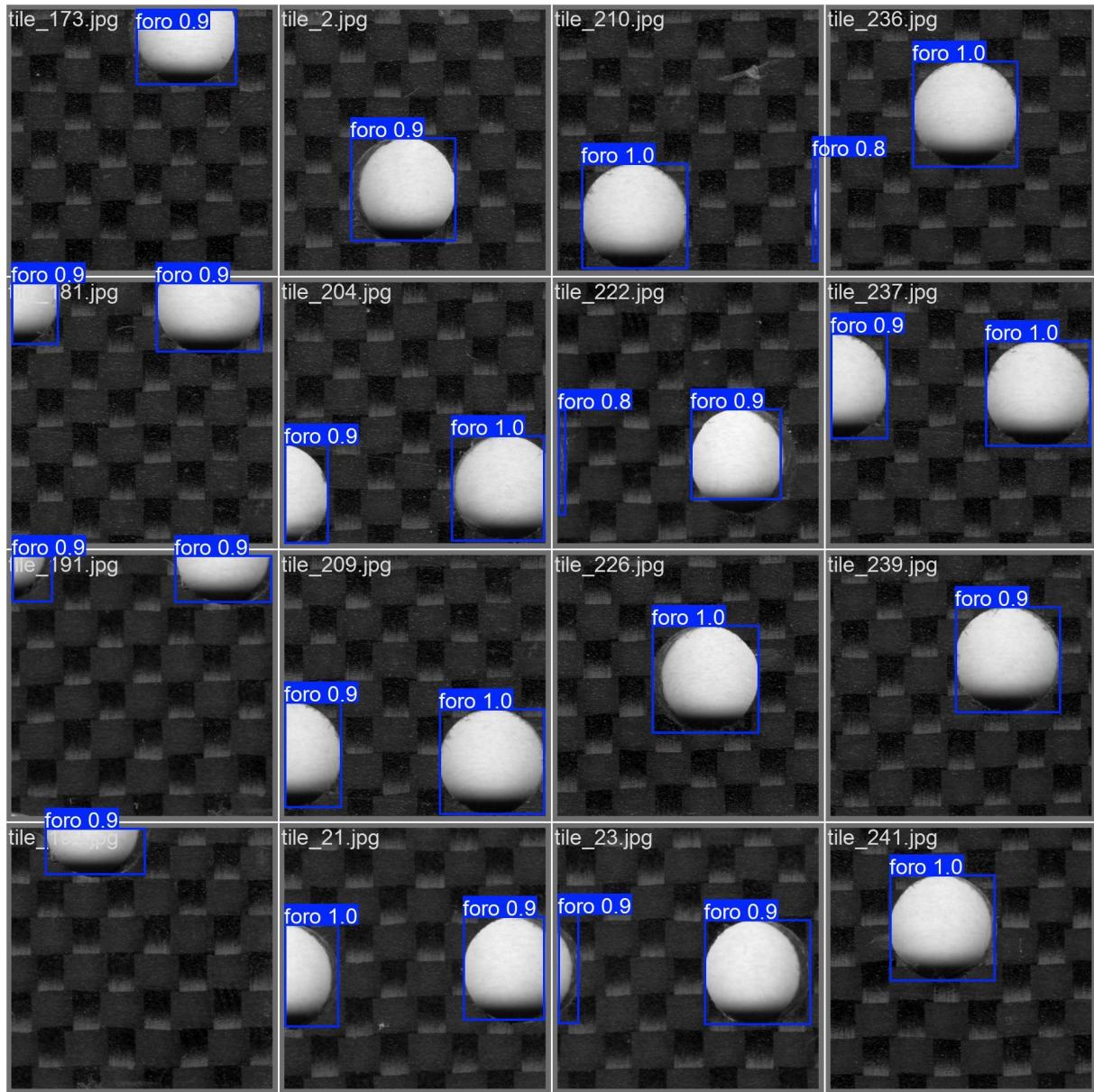


Figura 2.5: Griglia di esempi di predizioni del modello YOLOv8 sul validation set. Le bounding boxes blu indicano i fori rilevati, con il relativo punteggio di confidenza. Si noti la precisione anche sui fori parzialmente visibili.

Questi risultati confermano che il primo stadio della nostra pipeline è affidabile e produce localizzazioni dei fori di alta qualità, costituendo una base solida per le successive fasi di ordinamento e segmentazione di precisione. Si è anche testato con le radiografie ottenendo risultati analoghi, o in alcuni casi superiori.

LOG Completo

Log completo consultabile come `results_yolotrain.csv`.

estratto esempio visibile come **tabella 2.2**:

epoch	time	train /box _loss	train /cls_ loss	train/dfl_lo ss	metric s/precision(B)	metri cs/re call(B)	metric s/mA P50(B)	metrics /mAP5 0-95(B)	val/ box_ loss	val/cl s_los s	val/dfl _loss
1	8.1686	0.91	1.97	1.200	0.997	0.803	0.823	0.6527	0.90	1.761	1.107
	4	771	408	65	87	92	36	2	235	81	39
2	15.495	0.88	1.31	1.159	0.659	0.704	0.643	0.3607	1.00	2.557	1.179
	4	976	959	19	95	02	56	4	703	79	86
3	21.254	0.93	1.25	1.190	1	0.694	0.830	0.6604	0.83	1.537	0.989
	3	906	752	79		39	36	6	429	66	54
4	28.293	0.91	1.18	1.156	0.904	0.833	0.886	0.6659	0.90	1.213	1.093
	3	225	306	29	83	33	56		9	83	73
5	34.104	0.91	1.10	1.118	0.989	0.897	0.925	0.7136	0.79	1.362	0.974
	003	889	73	19	32	7	1	067	39	66	66

Tabella 2.2

Listato A.2 – Estrazione patch ROI con bounding box YOLO

Si è usato imgsz=(H,W) come parametro di inferenza dell’immagine. Per evitare il resize automatico del modello che portava alla perdita della rilevazione dei fori.



Figura 2.6 - Visione d'insieme delle Bounding Boxes predette su una scan completa. Il modello mantiene una confidenza > 0.90 sulla quasi totalità dei 600 fori, validando la robustezza dello stadio di localizzazione

2.4. File cucitura e ordinamento dei fori

Si è proceduto ad unire le scan divise orizzontalmente per ottenere delle scan da cui poter fare inferenza diretta senza rompere l'algoritmo di ordinamento kmeans:

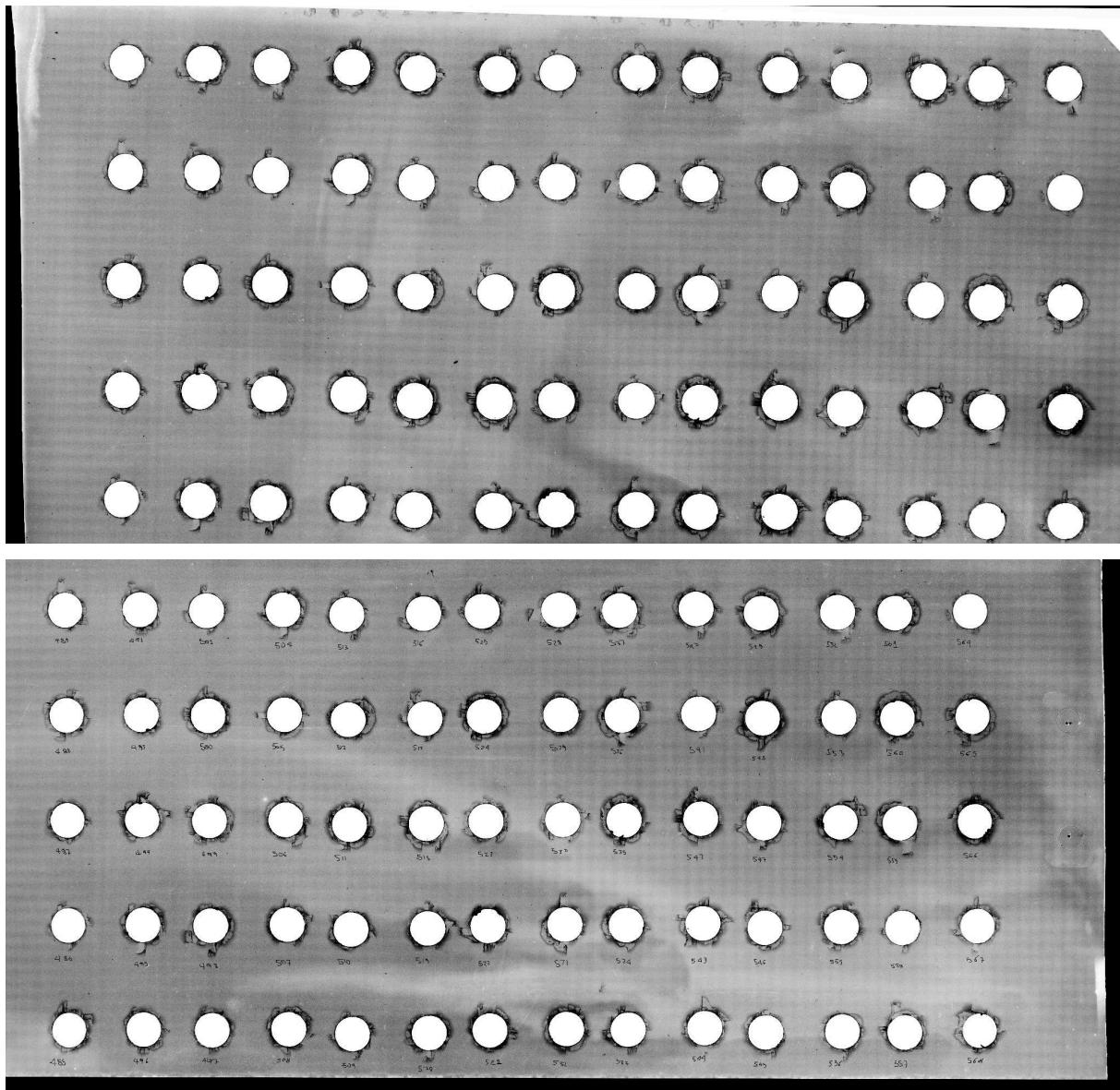


Figura 2.7.1 - 2.7.2 Sopra: le due scan (N.5 lato inferiore-superiore), in cui sono sovrapposte una parte delle file di fori.

Pagina seguente il risultato finale, si è prestata attenzione a non fare upscaling per non rompere l'algoritmo di ordinamento successivo. Per la cucitura si è proseguito in locale con uno script py listato **Appendice E.2**

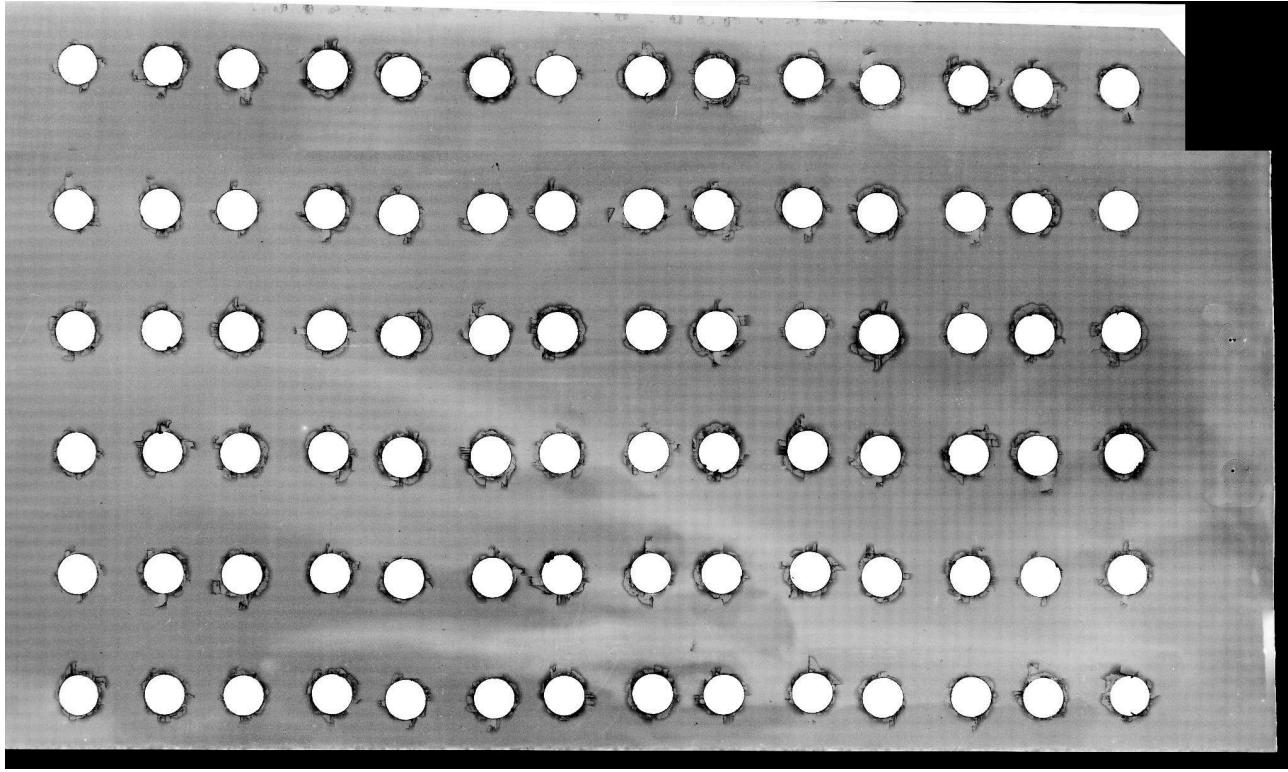


Figura 2.7.3 - Scan completa Cucita ottenuta dalle precedenti

2.5 : Ordinamento Bustrofedico tramite K-Means Clustering

Per imporre un indice sequenziale riproducibile, è stato implementato un algoritmo di ordinamento bustrofedico, uno dei contributi originali di questa fase. I centri delle *bounding boxes* vengono raggruppati in k colonne verticali tramite `sklearn.cluster.KMeans`. [12] Le colonne vengono quindi ordinate da sinistra a destra, e i fori all'interno di ciascuna colonna vengono indicizzati con un percorso a serpentina, partendo dal basso nelle colonne dispari e dall'alto in quelle pari, emulando il percorso di una macchina CNC

è stata considerata anche la casistica speculare, visto che i provini erano stati girati per il fronte retro dei provini in modo casuale, ma per semplicità si mostra uno snippet solo del primo caso, usato anche nelle radiografie.

Per simulare il percorso di una macchina CNC, è stato implementato un ordinamento bustrofedico basato su clustering K-Means (algoritmo dettagliato nel **Listato B.1 dell'Appendice B**).

2.5.1 Risultati della Pipeline di Acquisizione Dati

La pipeline sviluppata si è dimostrata robusta ed efficace. Nonostante la complessità e le imperfezioni dei dati grezzi, il sistema ha prodotto in output dei set di dati usabili per le fasi

successive è possibile vedere in **figura 2.8** della scan 6 con i box ordinati, si sono prodotti anche dei csv con le coordinate dei vari fori in valori assoluti da usare nei crop.

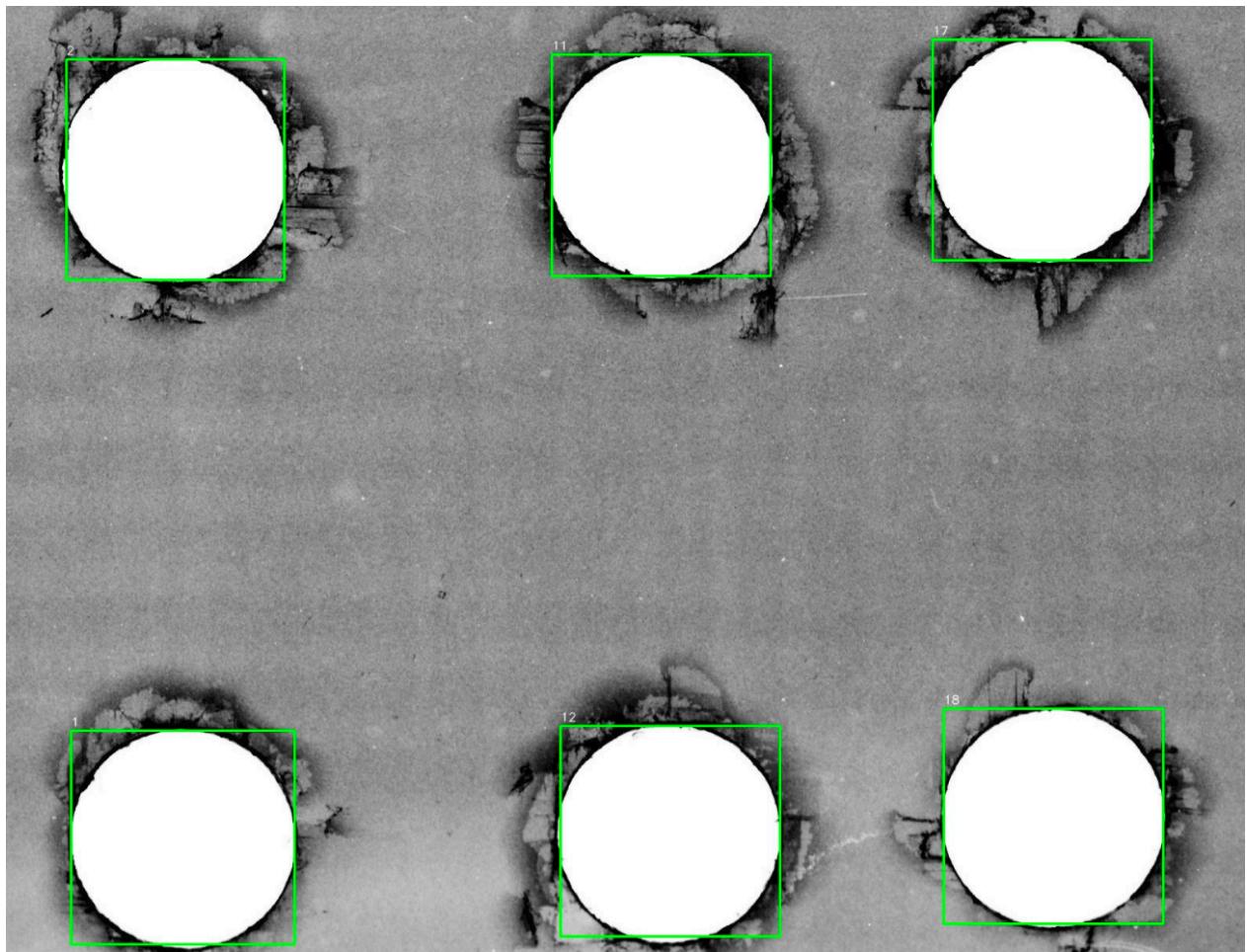


Figura 2.8 - Esempio ingrandito applicato alla scan 6, si vede l'ordine seguito

Capitolo 3: Pipeline A – Indagine su un Framework Cross-Modale (RPOS)

3.1 Premessa: L'ipotesi dell'Informazione Privilegiata

Listato Appendice E.3

L'obiettivo è quello di sfruttare la ricchezza di informazioni del dominio radiografico (la Ground Truth sul danno sub-superficiale) come *informazione privilegiata* per addestrare un modello di segmentazione che, una volta preparato, operi esclusivamente sulle più economiche e facilmente reperibili immagini ottiche.

Questo approccio, noto in letteratura come *Learning Using Privileged Information* (LUPI), avrebbe il vantaggio industriale di permettere una valutazione del danno accurata in contesti

dove le ispezioni radiografiche non sono praticabili, utilizzando un modello addestrato a Rilevare il danno nascosto attraverso le sue manifestazioni superficiali visibili.

Il prerequisito tecnico fondamentale per questo framework *teacher-student* è la capacità di stabilire una corrispondenza geometrica precisa tra il dominio ottico e quello radiografico, un processo noto come registrazione di immagini.

3.2 Implementazione Tecnica: Registrazione tramite Omografia

Per allineare le due modalità di immagine, è stata impiegata una trasformazione omografica. Un'omografia è una trasformazione proiettiva 2D che mappa i punti da un piano all'altro ed è rappresentata da una matrice H di 3×3 . La matrice H è stata calcolata utilizzando la funzione `cv2.findHomography`, che stima la trasformazione ottimale a partire da una serie di punti di corrispondenza tra le due immagini.

I punti di corrispondenza sono stati derivati automaticamente dai dataset di coordinate generati dalla pipeline di rilevamento descritta nel Capitolo 2: per ogni foro, il centroide della bounding box nell'immagine ottica è stato associato al centroide della bounding box corrispondente nell'immagine radiografica.

L'applicazione di un'unica omografia globale (H_{glob}) calcolata su tutti i fori si è rivelata insufficiente. Le analisi visive degli overlay hanno mostrato disallineamenti significativi, in particolare ai bordi dei campioni, indicando la presenza di distorsioni locali non lineari non catturabili da una singola trasformazione.

Per superare questo limite, è stata implementata una strategia di **omografia locale adattiva**:

1. **Selezione dei Vicini:** Per ogni foro, vengono identificati i suoi k vicini più prossimi (nel nostro caso, $k=9$) nel dominio radiografico.
2. **RANSAC Sweep:** Viene stimata un'omografia locale (H_{loc}) utilizzando solo questo sottoinsieme di punti. Per aumentare la robustezza, sono state testate iterativamente diverse soglie per l'algoritmo RANSAC (da 1.0 a 7.0 pixel), scegliendo quella che massimizzava il rapporto di *inlier*.
3. **Filtro di Qualità:** Ogni allineamento locale è stato validato tramite due metriche: lo **shift** (errore di riproiezione del centro del foro, in pixel) e lo **Structural Similarity Index (SSIM)**, che misura la somiglianza strutturale tra la patch ottica e quella radiografica allineata.

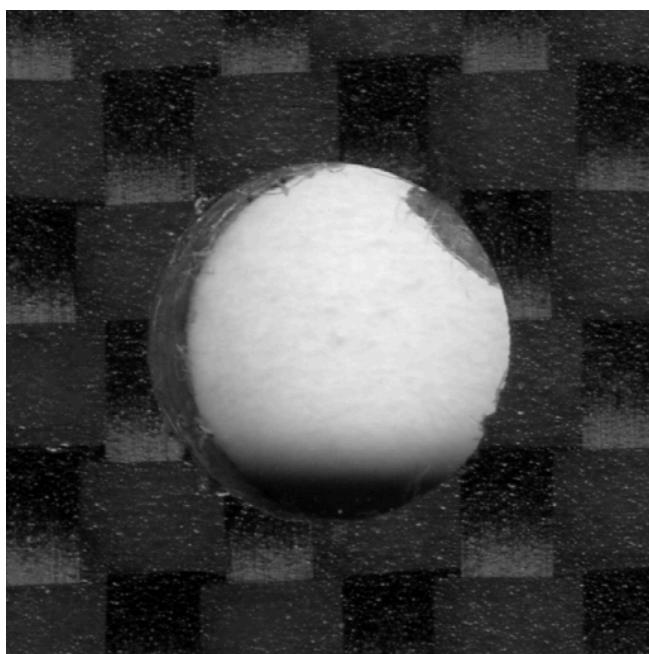
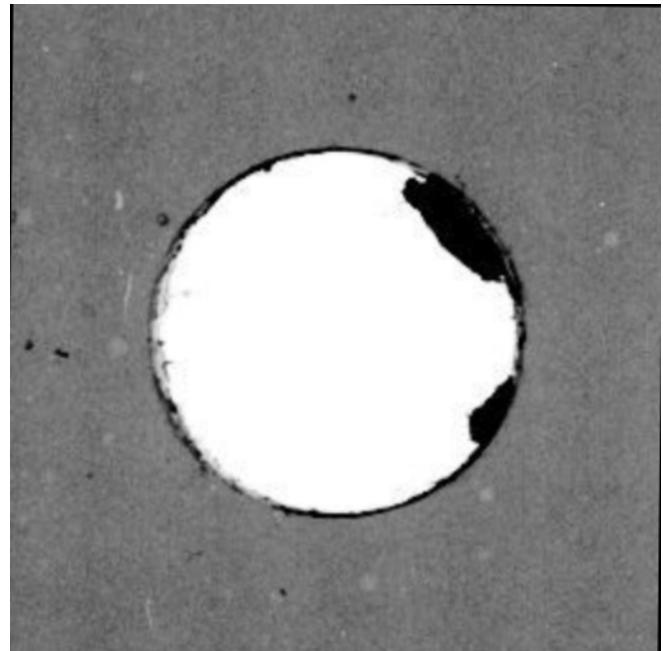
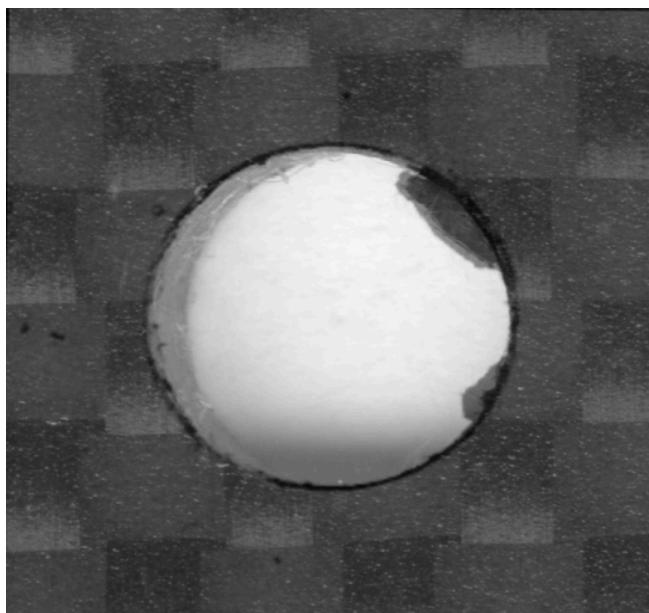


Figure 3.1.1 - 3.1.2 - in alto, scan ottica a sinistra, a destra scan radiologica Ground Truth

Figure 3.1.3 - In basso Overlay Radiografia e Ottica

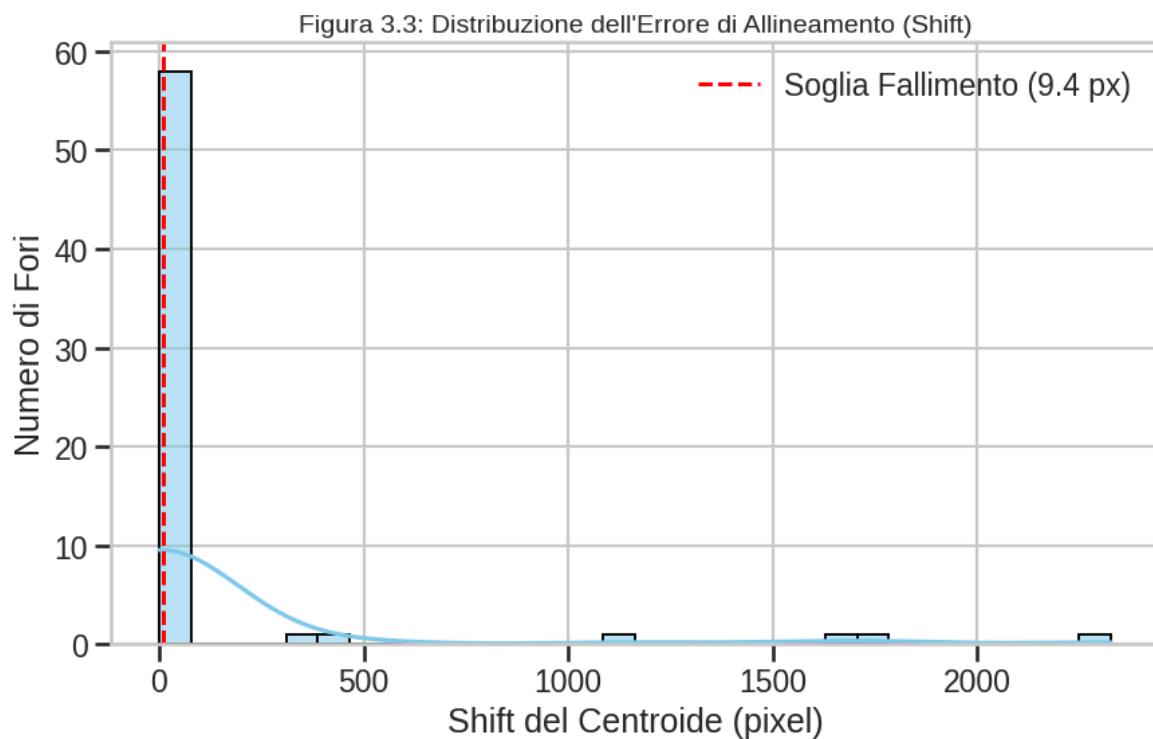
3.3 Risultati e Analisi Critica della Pipeline RPOS

Nonostante le ottimizzazioni, l'analisi quantitativa ha rivelato che la pipeline RPOS non era sufficientemente robusta per i dati a disposizione. Ogni allineamento è stato validato tramite due metriche:

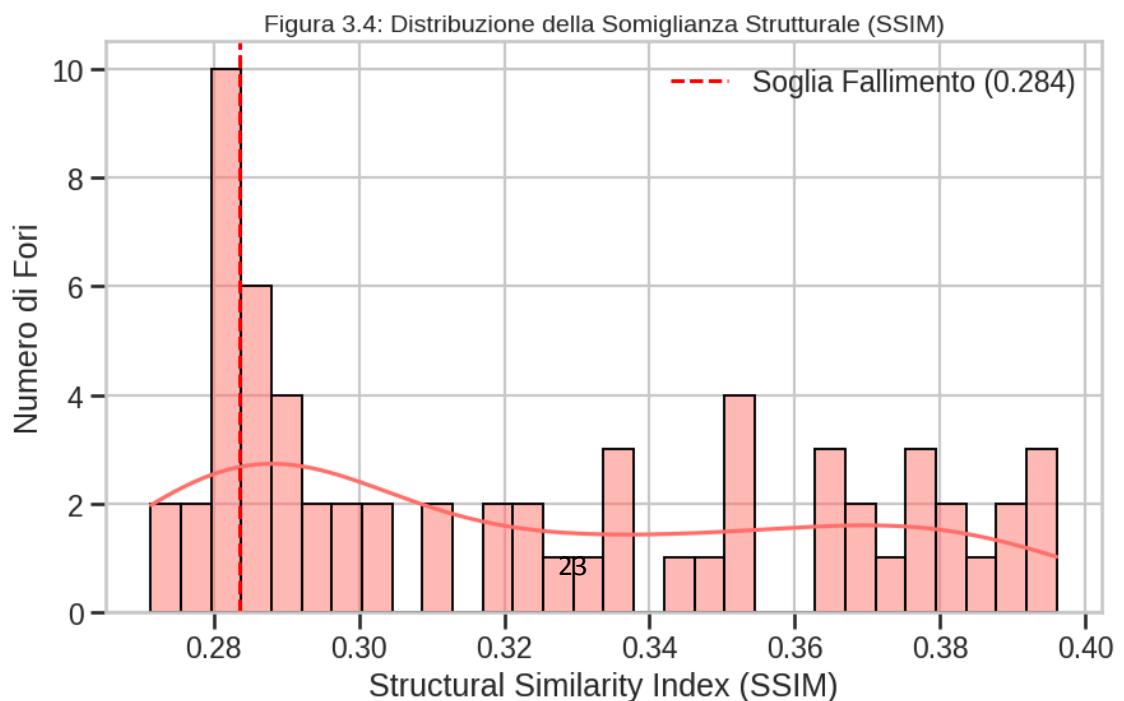
- Shift di riproiezione: l'errore Euclideo (in pixel) tra la posizione prevista e quella reale del centroide di un foro dopo la trasformazione. Valori alti indicano un cattivo allineamento geometrico.
- Structural Similarity Index (SSIM): un indice tra -1 e 1 che misura la somiglianza strutturale tra le due patch allineate. Valori bassi indicano differenze significative di forma, scala o rotazione.

Le distribuzioni di queste metriche su tutti i 64 fori sono mostrate nelle **Figure 3.3 e 3.4**.

Figura 3.3: Distribuzione dell'Errore (Shift)":



Si vede dal grafico che c'è un sottogruppo estremamente fuori allineamento **Figura 3.4**



L'analisi quantitativa ha portato a una conclusione inequivocabile. Definendo come "falliti" gli allineamenti nel 20% peggiore per ciascuna metrica ($\text{shift} > 14.2 \text{ px}$ o $\text{SSIM} < 0.28$), il tasso di successo complessivo della pipeline è risultato essere solo del 59.4%. Un totale del 40.6% dei campioni ha fallito almeno uno dei controlli di qualità, un valore inaccettabile per la creazione di un dataset di training affidabile. L'ispezione visiva dei casi falliti (es. Figura 3.5) conferma che le deformazioni non lineari tra le modalità erano troppo severe per essere corrette da una trasformazione omografica.

L'analisi quantitativa dei risultati (Figura 3.3 e 3.4) ha rivelato i limiti intrinseci dell'approccio cross-modale per il dataset in esame. Sebbene l'omografia locale adattiva sia riuscita ad allineare correttamente un sottoinsieme dei campioni, un'analisi aggregata ha evidenziato un tasso di fallimento significativo.

Come riportato dalle statistiche finali, su un totale di 64 fori analizzati, ben il 32.8% dei campioni è stato scartato perché non rispettava le soglie minime di qualità ($\text{shift} > 9.4 \text{ pixel}$ o $\text{SSIM} < 0.284$). Questo ha portato a un tasso di successo finale di solo il 67.2%, producendo un dataset "accettabile" di soli 43 fori per lo scan 1A.

Proiettando questo tasso di successo sull'intero dataset di ~600 fori, si otterrebbe un numero di campioni validi insufficiente per un addestramento robusto di una rete di segmentazione profonda. Inoltre, la complessità nel gestire i casi speculari ("uscite") avrebbe ulteriormente ridotto questo numero.

Si è pertanto concluso che, sebbene metodologicamente promettente, la pipeline RPOS basata su omografia era ingegneristicamente insostenibile ("not viable"), a causa della magnitudine delle distorsioni non-lineari tra le due modalità di scansione. Questa conclusione ha motivato la decisione strategica di abbandonare l'approccio cross-modale e di focalizzare lo sviluppo sulla Pipeline B, un'architettura più robusta e diretta operante esclusivamente all'interno del dominio radiografico

Si mostra un esempio di disallineamento in **figura 3.5**

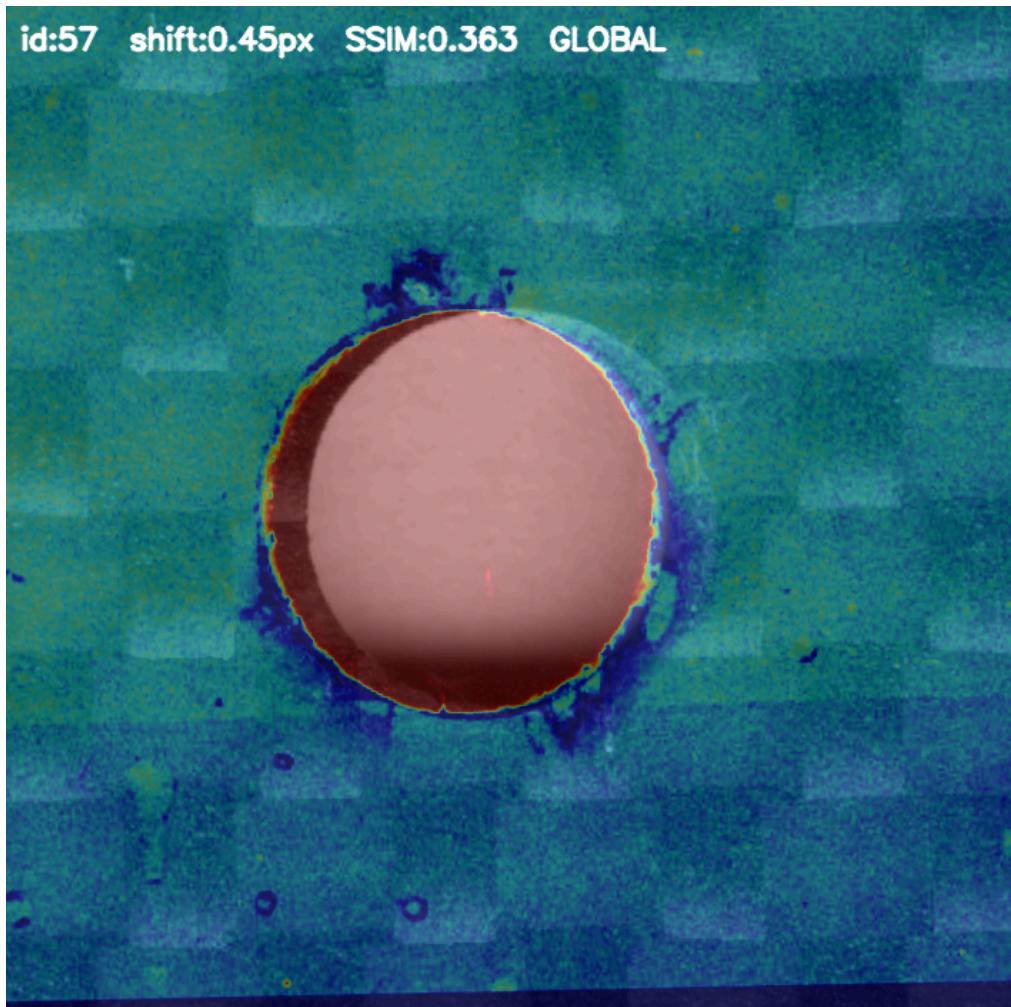


Figura 3.5 - Visualizzazione del disallineamento (shift) tramite overlay cromatico. Si nota come la distorsione non lineare del dominio ottico rispetto a quello radiografico renda l'omografia globale inapplicabile per fini metrologici.

Capitolo 4: Pipeline B – Analisi Integrata su Dominio Radiografico (ROIA)

4.1. Motivazione e Architettura Generale

A seguito delle sfide incontrate nell'approccio cross-modale, si è scelto di sviluppare una pipeline integrata operante esclusivamente sul dominio radiografico. L'architettura ROIA (Radiograph-Only Integrated Analysis) è un sistema sequenziale progettato per trasformare le scansioni grezze in predizioni temporali sul danneggiamento, concatenando moduli specializzati in un workflow MLOps end-to-end:

1. **Rilevamento e Normalizzazione (Modulo YOLO):** La pipeline di acquisizione dati descritta nel Capitolo 2 viene utilizzata come primo stadio per processare le scansioni

grezze, localizzare i fori, imporre un ordine bustrofedico e generare un dataset di **patch 512x512 con scala normalizzata**.

2. **Segmentazione del Danno (Modulo UNet++):** Le patch normalizzate vengono fornite a una rete UNet++ per eseguire la segmentazione semantica, producendo maschere a tre classi (sfondo, foro, danno).
3. **Estrazione Feature Ingegneristiche (Modulo di Calcolo):** Un algoritmo analizza le maschere e calcola un set di descrittori quantitativi (Area Delaminata, Dmax, Momenti di Hu), convertendoli in unità fisiche (mm, mm²) tramite una **scala canonica fissa**, resa possibile dalla normalizzazione a monte.
4. **Imputazione Dati Mancanti (Modulo MLP):** I dati di forza di processo, incompleti per alcuni campioni, vengono completati tramite un modello MLP addestrato a predire i valori mancanti.
5. **Modellazione Predittiva (Modulo LSTM):** Il dataset finale, completo e coerente, viene utilizzato per addestrare un modello LSTM in grado di predire l'evoluzione delle feature di danno in funzione della storia pregressa.



Figura 4.1.0 - Pipeline completa B

4.2. La Criticità Nelle Scale: Normalizzazione

Un'analisi preliminare del dataset ha rivelato un'inconsistenza critica: fori nominalmente identici (\varnothing 6 mm) presentavano dimensioni in pixel drasticamente diverse tra le scansioni, variando da ~120 px a ~300 px. Questa eterogeneità, dovuta a procedure di acquisizione non standardizzate, rendeva impossibile qualsiasi analisi metrica basata su una scala fissa.

Per risolvere questa criticità alla radice, il problema è stato affrontato **nella fase di generazione delle patch**. È stato implementato un algoritmo di cropping con **normalizzazione di scala integrata**, che assicura che in ogni patch di output da 512x512 pixel il foro abbia sempre una dimensione target costante (\approx 290 pixel).

Carico Computazionale in Vram(Gb): scala log

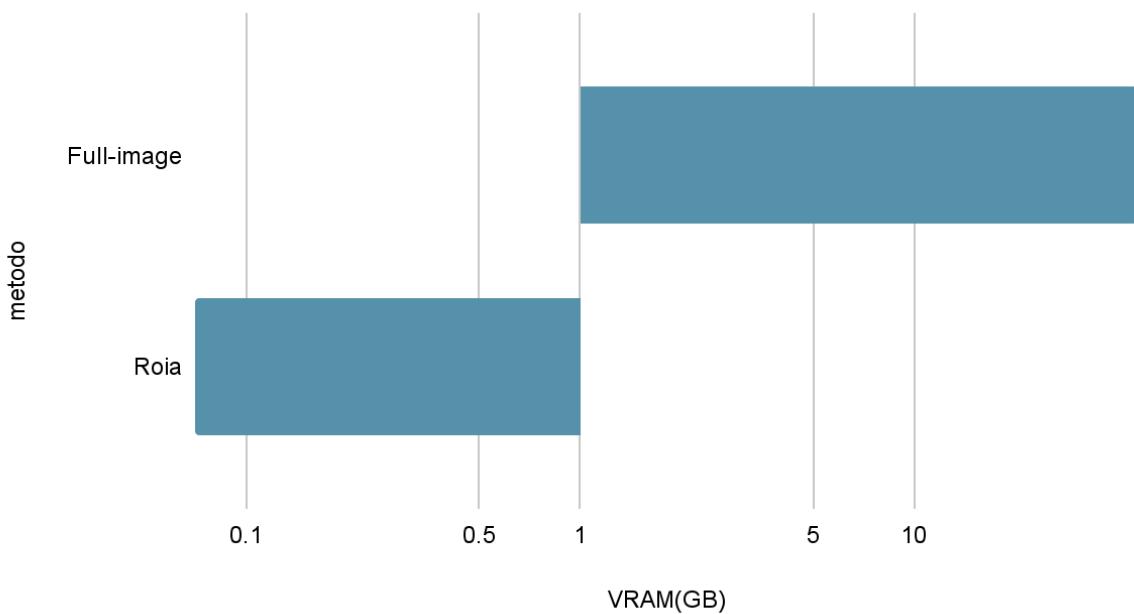


Grafico 4.1: Analisi dell'efficienza computazionale (scala logaritmica). Il confronto evidenzia l'insostenibilità dell'approccio a risoluzione nativa, che richiederebbe hardware di classe Enterprise (VRAM > 40 GB), rispetto alla pipeline ROIA ottimizzata che opera stabilmente in circa 70 MB di VRAM.

Si mostra poi sotto una tabella dei valori ricavati da script colab

Scan	Risoluzione immagine (px)	Diametru o foro medio (px)	Scala implicit a (px / mm)
Carbon Textile 1A	4780 × 1662	≈ 120 px	≈ 20.0
Carbon Textile 3B	4399 × 1231	≈ 116 px	≈ 19.3

Scan	Risoluzione immagine (px)	Diametro foro medio (px)	Scala implicita (px / mm)
scan4_cucita_senza_linea	12497 × 5820	≈ 300 px	≈ 50.0
scan5_shift_0610	9100 × 5397	≈ 260 px	≈ 43.0
T_0_90_6	3585 × 4820	≈ 145 px	≈ 24.0

Tabella 4.1 - Valori ricavati dai CSV YOLO (media dei bounding-box) – la variabilità è > 150 %.

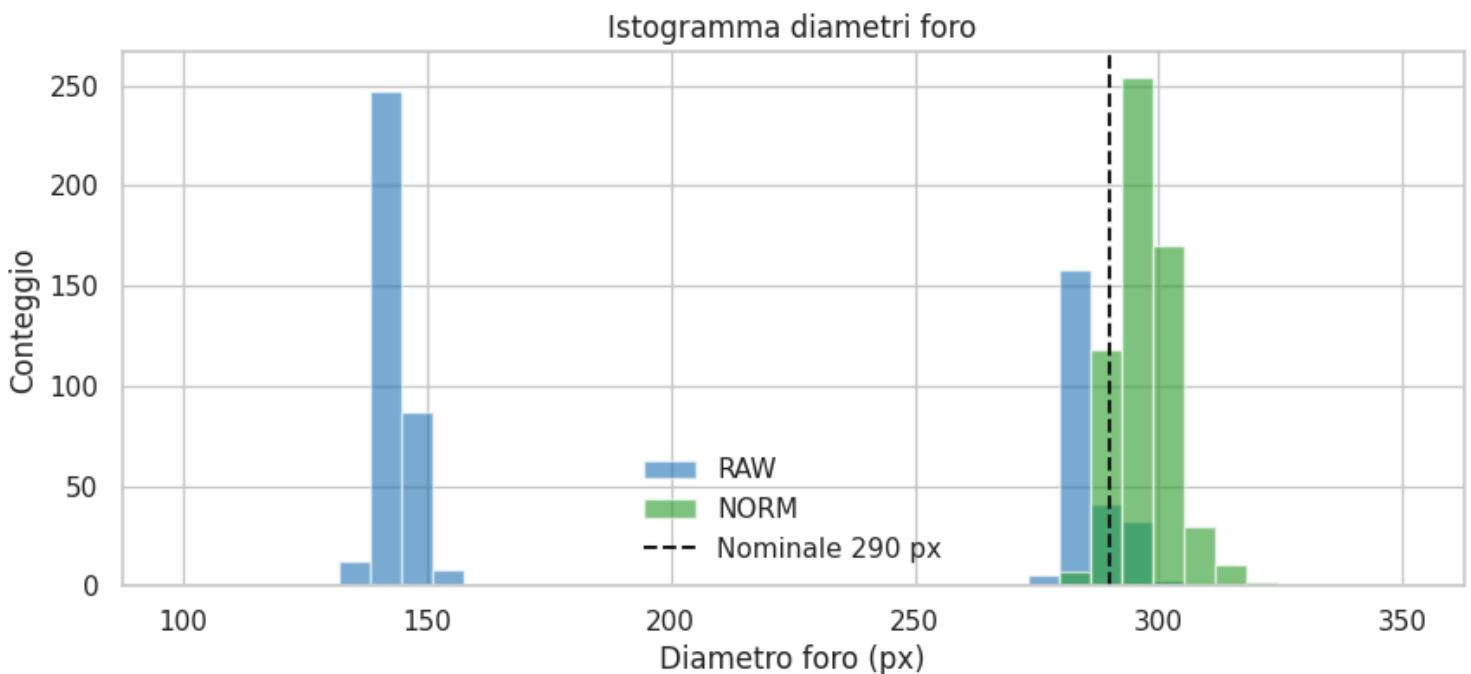


Figura 4.2 - I grafici a istogramma PRIMA e DOPO che mostrano l'effetto della normalizzazione

Questa normalizzazione a monte è un passaggio chiave che garantisce la robustezza dell'intera pipeline, permettendo alla UNet++ di addestrarsi su esempi consistenti e al modulo di

estrazione feature di utilizzare una scala canonica fissa per una conversione affidabile in unità fisiche

4.3 Generazione del Dataset di Pre-Training: La "Mask Factory" Automatica

Per addestrare efficacemente una rete di segmentazione profonda come la UNet++, è necessario un vasto dataset. Data l'impossibilità di etichettare manualmente centinaia di campioni, è stata sviluppata una pipeline automatizzata, denominata "Mask Factory", per generare un dataset di pre-addestramento di circa 600 maschere a partire dalle patch normalizzate.

L'approccio iniziale, basato su soglie di luminosità fisse, si è rivelato inefficace. Sebbene riuscisse a segmentare il foro (classe 1), falliva sistematicamente nel distinguere il danno a bassa intensità ("ragnatela") dalla texture di fondo del materiale composito.

Si è quindi implementato un algoritmo più robusto, basato sull'analisi della densità dei bordi, la cui logica è la seguente:

1. Localizzazione Geometrica del Foro: Il foro viene prima localizzato con precisione tramite la trasformata di Hough (cv2.HoughCircles), definendo un'ancora geometrica.
2. Definizione di una Regione di Interesse (ROI): Viene creata una "ciambella" attorno al foro, escludendo l'area del foro stesso ma includendo una regione circostante dove è probabile che si trovi il danno.
3. Rilevamento dei Bordi: All'interno della ROI, viene applicato il rilevatore di bordi di Canny per identificare le discontinuità di texture associate al danno.
4. Creazione di una Mappa di Densità: I bordi rilevati vengono "addensati" tramite un'operazione morfologica di chiusura (cv2.morphologyEx con MORPH_CLOSE) per creare una mappa in cui le aree ad alta densità di bordi corrispondono alle zone di danno.
5. Segmentazione Finale: La mappa di densità viene binarizzata per generare la maschera finale del danno (classe 2).

Questo approccio basato sulla texture si è dimostrato significativamente più efficace nel catturare il danno complesso e filiforme.

Prima dell'addestramento, è stata generata una base di maschere tramite una pipeline di pre-segmentazione algoritmica basata su operatori morfologici e rilevamento di Canny (implementazione nel **Listato C.1 dell'Appendice C**)

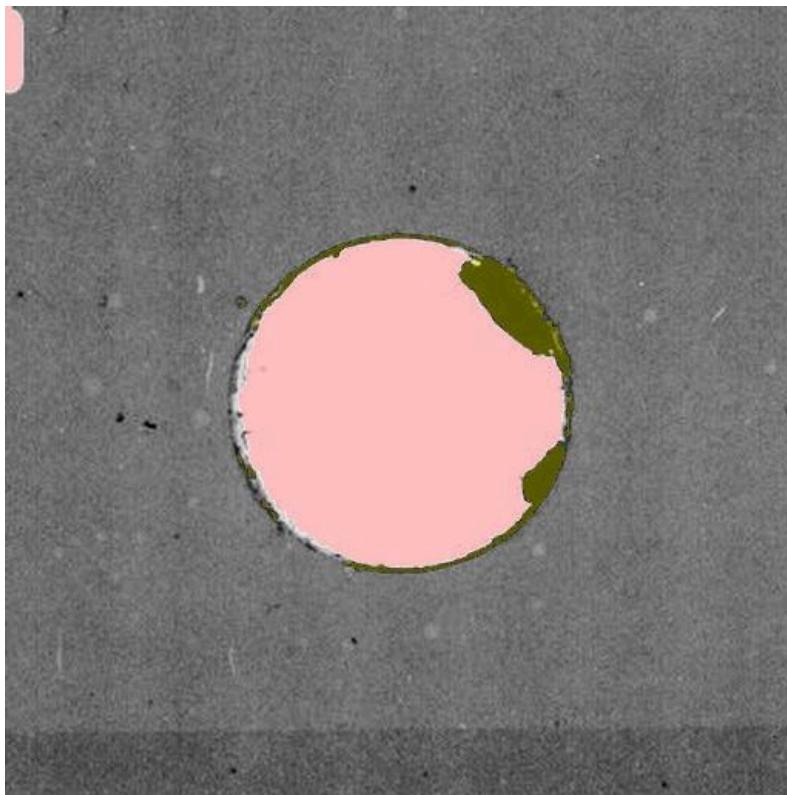


Figura 4.3 - esempio su foro 1

È importante notare che le maschere prodotte da questa pipeline, sebbene efficaci visivamente, tendono ad essere poco **conservative**, talvolta sottostimando l'estensione reale del danno. Tuttavia, hanno fornito un dataset di pre-addestramento sufficientemente vasto e coerente per permettere al modello UNet++ di apprendere le feature fondamentali del problema, come verrà descritto nella sezione successiva.

4.4 Fase 2: Fine-Tuning sul "Golden Dataset" e Diagnosi del Plateau di Apprendimento

Una volta ottenuta una base di conoscenza solida con il pre-training, il passo successivo è stato quello di specializzare il modello sui dati di massima qualità. Il modello **best_model_pre-training.pth** è stato quindi sottoposto a un processo di fine-tuning, utilizzando il "golden set" di 108 maschere verificate manualmente.

La strategia adottata per questa fase è stata la seguente:

- **Scongelamento completo:** Tutti i layer del modello, incluso l'encoder, sono stati resi addestrabili (`param.requires_grad = True`) per permettere alla rete di affinare ogni suo parametro in base ai dati di alta qualità.
- **Tasso di Apprendimento Ridotto:** Per evitare di perdere la conoscenza acquisita durante il pre-training con aggiornamenti troppo aggressivi, è stato utilizzato un *learning rate* molto basso, pari a $1\text{e}-5$.

L'addestramento è stato condotto in modo **incrementale**, aumentando progressivamente il numero di campioni "golden" (da 10, a 20, fino a 50) per osservare la reazione del modello a un dataset di alta qualità in crescita.

Tabella 4.2: Sintesi delle performance del training

Epoch	Train Loss	Val Loss	LR	Note
18	0.0180	0.0317	0.000232	Nuovo modello migliore – pesi salvati
19	0.0163	0.0227	0.000181	Nuovo modello migliore – pesi salvati
20	0.0165	0.0190	0.000136	Nuovo modello migliore – pesi salvati
21	0.0153	0.0197	0.000095	—
22	0.0153	0.0220	0.000062	—
23	0.0147	0.0198	0.000035	—
24	0.0154	0.0181	0.000016	Nuovo modello migliore – pesi salvati
25	0.0146	0.0182	0.000004	fine train

4.4.1 Fase 2: Fine-Tuning sul "Golden Dataset" e Diagnosi del Plateau di Apprendimento

Una volta ottenuta una base di conoscenza solida con il pre-training, il passo successivo è del processo di fine-tuning, utilizzando un "golden set" di maschere verificate manualmente in CVAT, per correggere le imprecisioni sistematiche delle maschere automatiche.

La strategia adottata per questa fase è stata la seguente:

- Scongelamento completo: Tutti i layer del modello, incluso l'encoder, sono stati resi addestrabili (param.requires_grad = True) per permettere alla rete di affinare ogni suo parametro in base ai dati di alta qualità.
- Tasso di Apprendimento Ridotto: Per evitare di perdere la conoscenza acquisita con aggiornamenti troppo aggressivi, è stato utilizzato un *learning rate* molto basso, pari a 1e-5.

L'addestramento è stato condotto in modo incrementale, iniziando con un piccolo sottoinsieme di 11 campioni "golden" (8 per il training, 3 per la validazione) per osservare il comportamento del modello.

Risultato Diagnostico: Identificazione di un Plateau di Performance

I log di addestramento (esempio tabella 4.3) hanno rivelato un comportamento critico e informativo: il modello raggiungeva molto rapidamente un plateau di apprendimento, smettendo di migliorare dopo la primissima epoca. Questo fenomeno indicava che, a causa delle ridotte dimensioni del dataset di fine-tuning, il modello stava iniziando a memorizzare gli esempi di training (overfitting) invece di apprendere feature generalizzabili.

Tabella 4.3

Tabella di log sessione 1 su 10 immagini aggiuntive			
Epoca	Validation Loss (Dice)	Δ vs Epoca Precedente	Interpretazione
1	0.7021	–	Rapido adattamento iniziale, si raggiunge il minimo
2	0.7026	+0.0005	Peggioramento, segnale di inizio overfitting
3	0.7021	-0.0005	Ritorno al minimo precedente
4	0.7021	0.0000	Stagnazione completa

Tabella di log sessione 1 su 10 immagini aggiuntive

			(plateau)
5	0.7026	+0.0005	Stagnazione e potenziale overfitting

Dopo un primo, rapido adattamento alla prima epoca, il modello non è più in grado di estrarre nuova informazione significativa dal dataset. La loss di validazione "rimbalza" attorno a un valore minimo senza una chiara tendenza al miglioramento.

Questa diagnosi è stata fondamentale. Ha dimostrato che, sebbene la pipeline di fine-tuning fosse tecnicamente funzionante, il segnale di apprendimento fornito dalle sole immagini intere era insufficiente. Il modello non riusciva a specializzarsi sulle micro-strutture del danno, portando alla decisione di sviluppare una strategia di addestramento ibrida e più mirata, descritta nella fase successiva.

Nonostante il plateau numerico, il confronto visivo ha comunque mostrato un leggero miglioramento nella coerenza delle maschere rispetto al modello di pre-training, confermando che la direzione era corretta, sebbene la strategia necessitasse di un'ulteriore evoluzione.

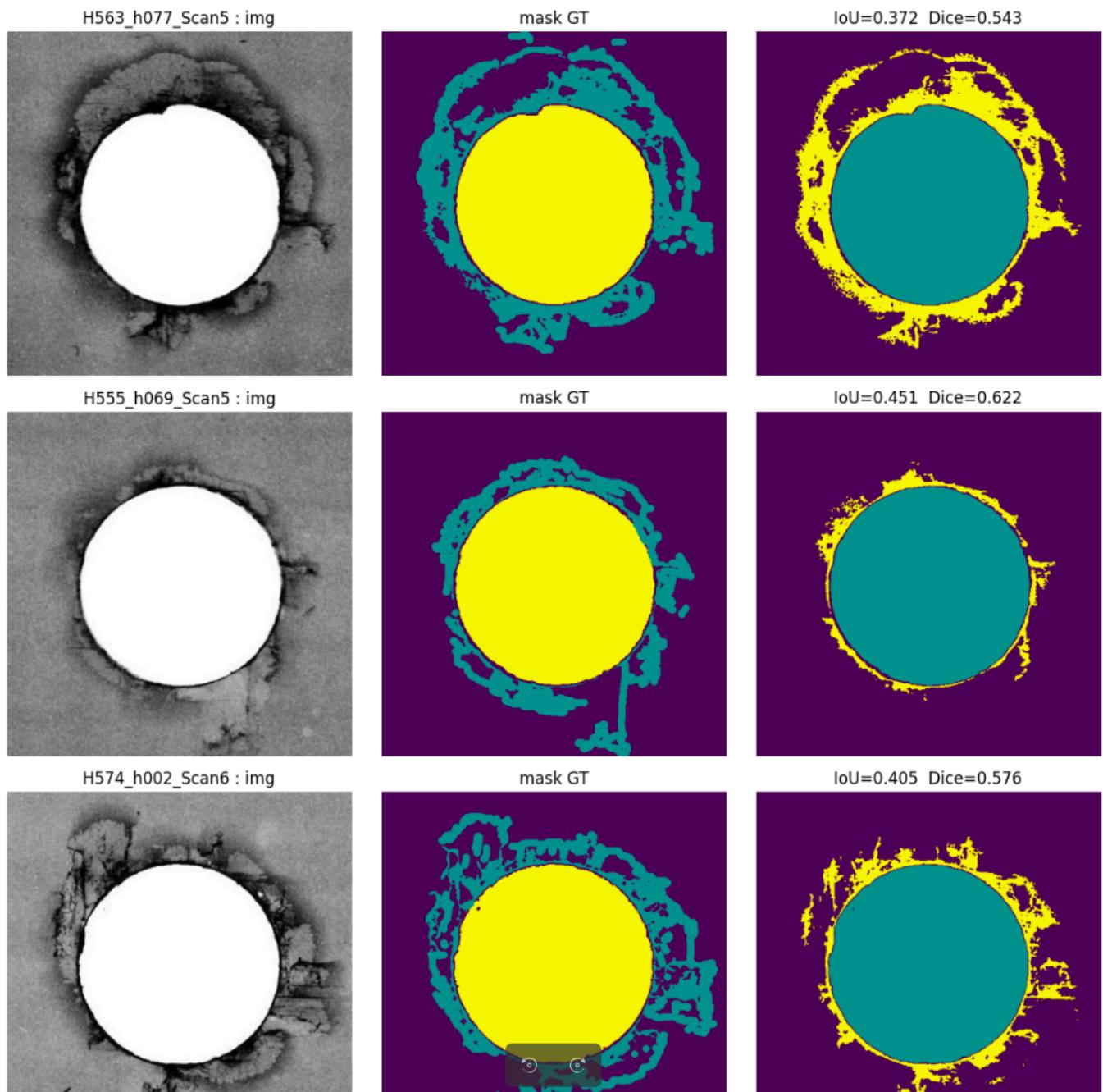


Figura 4.5 - Confronto tra maschera *Ground Truth* e inferenza *UNet++*, confermano la capacità della rete di delineare correttamente anche le delaminazioni a bassa intensità

4.5 Addestramento Ibrido "Foveale" per l'Analisi dei Dettagli

La motivazione del nome risiede nell'occhio, capace di distinguere sia i dettagli che l'insieme dell'informazione visiva:

Infatti si nota che il modello ha difficoltà a rilevare i danni a ragnatela con variazioni di gradiente di contrasto cromatico poco netta.

Si è quindi preposto dalle maschere precedenti a ottenere un sottoinsieme di dettagli delle delaminazioni a ragnatela da passare al modello durante l'addestramento, per velocizzare il processo e rompere velocemente il plateau raggiunto precedentemente.

L'approccio si basa sulla creazione di due dataset distinti che vengono utilizzati simultaneamente:

1. **Dataset di Contesto (Immagini Intere):** Le solite immagini 512x512. Il loro compito è continuare a insegnare al modello il "**cosa**" e il "**dove**": la forma generale del foro, la texture del materiale, e la localizzazione approssimativa del danno. Questo previene la "dimenticanza catastrofica" e mantiene la comprensione del quadro generale.
2. **Dataset Foveale (Patch di Dettagli):** Un nuovo dataset composto da centinaia di piccole patch (es. 128x128px) estratte **specificamente dalle regioni più complesse** delle maschere di training (i bordi delle ragnatele). Queste patch vengono poi "zoomate" (upscale) alla dimensione di input del modello (512x512). Il loro compito è insegnare al modello il "**come**": come è fatta la micro-struttura del danno, come tracciare linee sottili e come riconoscere pattern a bassa visibilità.

Inviando al modello, nello stesso ciclo di training, un mix di questi due tipi di dati, lo costringiamo a diventare un esperto sia nella visione d'insieme che nell'analisi dei dettagli microscopici. È un "allenamento incrociato" che mira a creare un modello più robusto e con un approccio più assertivo nella segmentazione delle zone di danno.

4.6 L'Algoritmo

La sfida principale era isolare in modo automatico le regioni di danno più complesse. I metodi di Computer Vision classici si sono rivelati inefficaci, in quanto le texture fibrose del danno a "ragnatela" sono difficilmente distinguibili dallo sfondo e i bordi stessi del foro venivano spesso erroneamente classificati.

Si è quindi sviluppato un **algoritmo di esclusione**, denominato "Blackout", che inverte il problema: invece di cercare il danno, identifica ed esclude l'area in cui il danno complesso è assente per definizione, ovvero il foro.

Fondamentalmente si esclude l'area centrale di foro e si generano delle patch randomiche attorno al foro dove abbiamo le maggiori variazioni di contrasto.

La maschera del foro viene dilatata tramite un'operazione morfologica, utilizzando un kernel ellittico di grandi dimensioni (**BLACKOUT_MARGIN**). Questo crea una vasta "zona di esclusione" o `blackout_zone` che copre l'intero foro e un ampio margine di sicurezza attorno ad esso.

A questo punto si procede alla verifica manuale delle patches e si selezionano le migliori, subito dopo si effettua un crop automatico da maschere Radiologiche e binarie.

Listato C.2 - Creazione patch foveali

Isolamento del Danno Remoto: La blackout_zone viene sottratta dalla maschera totale del danno. I pixel rimanenti (surviving_damage_mask) rappresentano esclusivamente le aree di danno lontano dal foro, ovvero le regioni di "ragnatela" più interessanti.

Listato C.3 - Preselezione e campionamento delle patches

Campionamento e Cropping Foveale: Vengono campionate N coordinate casuali dai pixel della surviving_damage_mask. Ognuna di queste coordinate funge da centro per estrarre una patch di 128x128 pixel sia dall'immagine originale sia dalla maschera "golden".

Listato C.4 - Campionamento e estrazione delle patches

Questo processo ha generato un set di centinaia di patch-candidato. Da queste, un sottoinsieme di **24 patch**, giudicate visivamente le più rappresentative delle tipologie di danno complesso, è stato selezionato manualmente per costituire il "Dataset Foveale" finale, archiviato nel file GoldenPatches_v1.zip.

Figure 4.6.1 - 4.6.2: due esempi di selezione di sottopatches 128x128

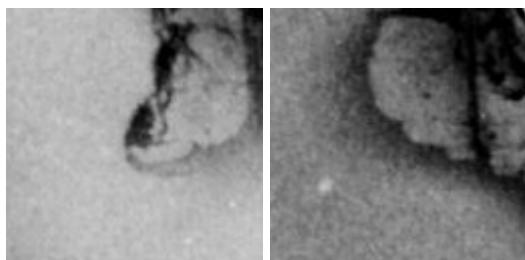


Figura 4.7.1 sotto - Ispezione delle inferenze visive, maschere, pre-foveale, post-foveale

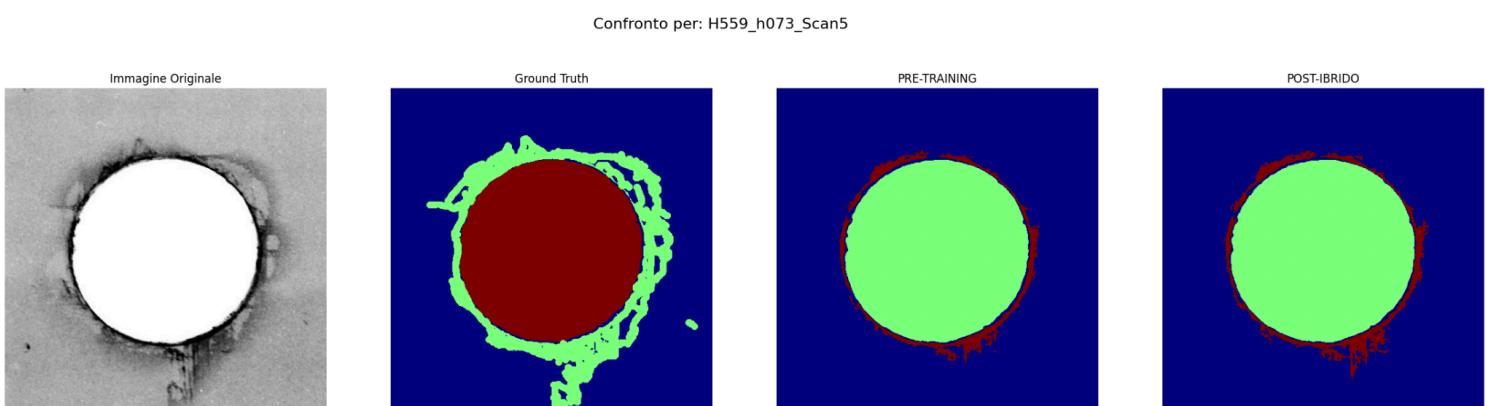
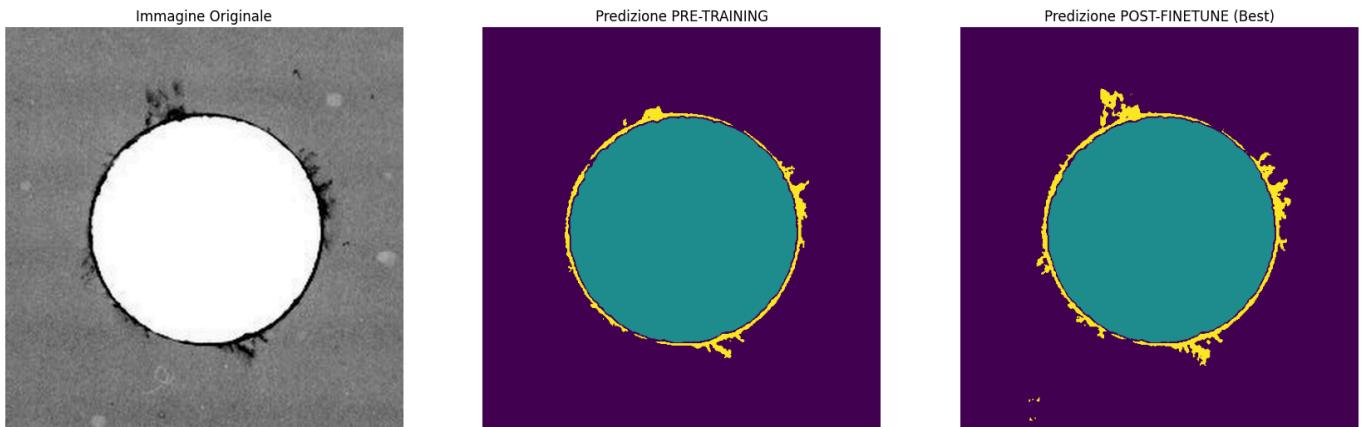


Figura 4.7.2 sotto - Fonte, pre addestramento foveale, post addestramento foveale

Confronto per: H019_h019_Carbon Textile 1A.jpg

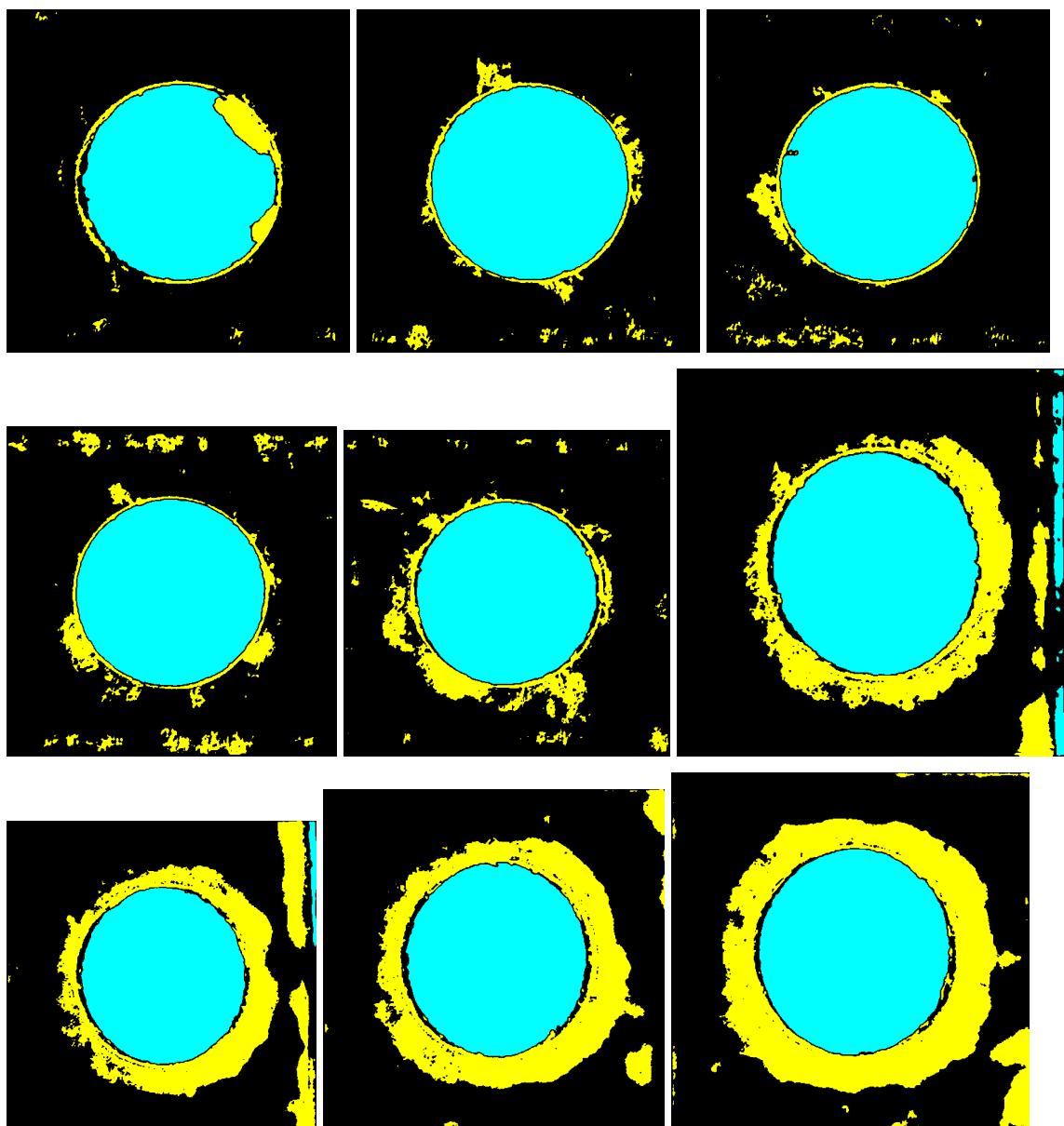


Si mostrano due esempi nelle immagini 4.7.1 e 4.7.2, in cui si può notare la progressione nella capacità di rilevazione del danno da Unet++ con confronti fra il pre addestramento foveale e post, nella immagine 4.7.1 in particolare si può osservare anche la maschera usata come elemento di addestramento.

In alcuni esempi successivi si può notare come in alcune patches il modello Unet++ si è rilevato un evidente overfitting, in cui il modello ha iniziato a generare artefatti inesistenti (allucinazioni) basati sulla memorizzazione del training set, si tratta di un numero ridotto di elementi, che potranno in futuro essere aggiustati con operazioni in software quali Qupath o Cvat.

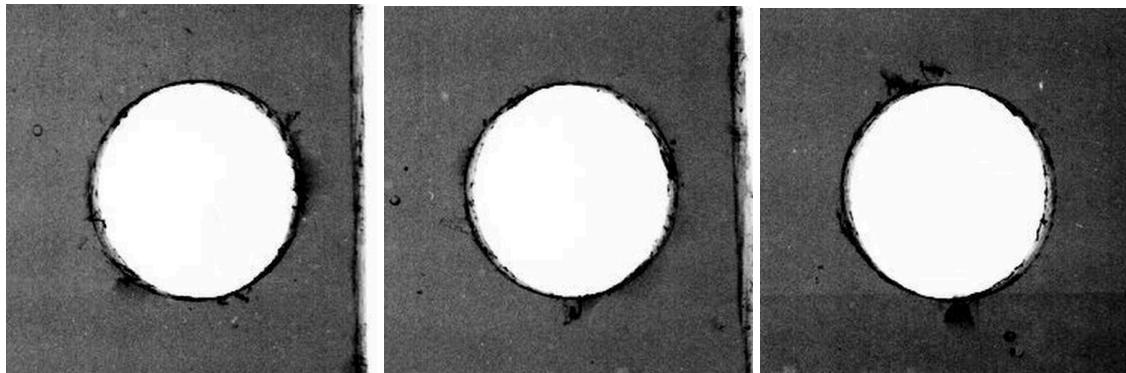
In alternativa si propone anche una esclusione morfologica ellittica o circolare al di fuori di un raggio di interesse definito tramite algoritmo.

Figura 4.8 - Serie di immagini di maschere post inferenza Unet++, rilevate in modo da selezionare quelle con artefatti più prominenti



Si nota dalla serie di immagini in figura 4.8 che la sezione di bordo delle lastre sia la più critica per il modello; è stato interpretato lo sfondo come foro e la variazione cromatica come danno aggiuntivo(non esistente in realtà). Si può interpretare ciò dalle immagini associate delle radiografie nella serie in figura 4.9

Figura 4.9 - Serie di immagini di Radiografie da cui risultano associati più artefatti in seguito alla inferenza di Unet++



Capitolo 5: Modulo di Estrazione Features Quantitative

Listato appendice E.4

Avendo ricevuto dal modello UNet++ una maschera di segmentazione di alta qualità per ciascuno dei 596 fori, il passo successivo è stata la traduzione di queste informazioni visive in un set di descrittori quantitativi e ingegneristicamente significativi. Il calcolo dei descrittori quantitativi è stato affidato a un modulo Python dedicato (**Listato E.4, Appendice E**), che automatizza la conversione pixel-millimetro sfruttando il diametro nominale del foro come riferimento metrico invariante: **EstrattoreFeatures.py**

La criticità fondamentale in questo processo è la conversione da pixel a unità fisiche (mm^2). Per affrontare questa sfida, la pipeline ROIA implementa una **strategia di calibrazione a due livelli**:

1. **Normalizzazione a Monte (livello Dataset):** Come descritto nel Capitolo 2, le patch di input fornite alla UNet++ sono già state generate tramite un **cropping adattivo** che normalizza la dimensione apparente del foro. Questo garantisce che il modello di segmentazione operi su dati dimensionalmente coerenti.
2. **Auto-Calibrazione a Valle (livello Feature):** Per massimizzare la precisione metrica e annullare qualsiasi residua variabilità di scala, l'estrattore di feature implementa un algoritmo di **auto-calibrazione per-immagine**. Questo approccio non si fida di una scala teorica fissa, ma la ricalcola dinamicamente per ogni singola maschera.

La logica si basa sull'utilizzo del foro stesso, il cui diametro fisico è noto a priori (6 mm), come "righello interno" (internal ruler).

La robustezza metrologica è garantita da un modulo di autocalibrazione per-immagine che ricalcola la scala pixel/mm sfruttando il diametro noto del foro (si veda **Listato D.1, Appendice D**)

Questo approccio a doppio livello garantisce una robustezza metrologica eccezionale, disaccoppiando l'accuratezza delle misure finali dalle inevitabili piccole variazioni nei processi di acquisizione e cropping a monte.

L'esecuzione di questo script su tutte le 596 maschere ha generato il file **features_finali_auto-calibrate.csv**, un dataset strutturato e metricamente robusto. Questo file rappresenta l'input fondamentale per l'ultima fase della pipeline ROIA: la modellazione predittiva.

Capitolo 6: Modellazione Predittiva dell'Evoluzione del Danno

Listato Appendice E.5

Con un dataset di feature ingegneristiche robusto e metricamente coerente, l'obiettivo finale della pipeline ROIA è lo sviluppo di un modello in grado di predire l'evoluzione sequenziale del danno. Questa fase si articola in due passaggi critici: la preparazione di un dataset temporale completo, che richiede l'imputazione dei dati di forza mancanti, e lo sviluppo di un modello predittivo basato su una rete neurale ricorrente LSTM.

6.1. Imputazione dei Dati di Forza Mancanti: un'Analisi Diagnostica

Il dataset sperimentale di riferimento, derivato dalla tesi di Melis (2018) [13], presentava una lacuna: per 16 fori sequenziali, i valori della forza di processo (Forza_N) non erano disponibili. Poiché una serie temporale ininterrotta è un prerequisito fondamentale per la modellazione con LSTM, si è resa necessaria una strategia di imputazione per stimare questi valori mancanti.

È stata condotta un'analisi metodica per determinare l'approccio più affidabile, confrontando due ipotesi principali.

La prima ipotesi assumeva che le caratteristiche geometriche del danno (Area, Dmax, Momenti di Hu), essendo una conseguenza della forza applicata, contenessero informazioni sufficienti per predire la forza stessa. Per testare questa ipotesi, è stata addestrata una rete neurale Multi-Layer Perceptron (MLP) utilizzando l'intero set di feature disponibili (geometriche + NumeroForo) come input per predire Forza_N.

L'addestramento ha mostrato una buona convergenza, come evidenziato dalle curve di apprendimento in **Figura 6.1**, con la loss di validazione che si è stabilizzata, indicando l'assenza di overfitting significativo.

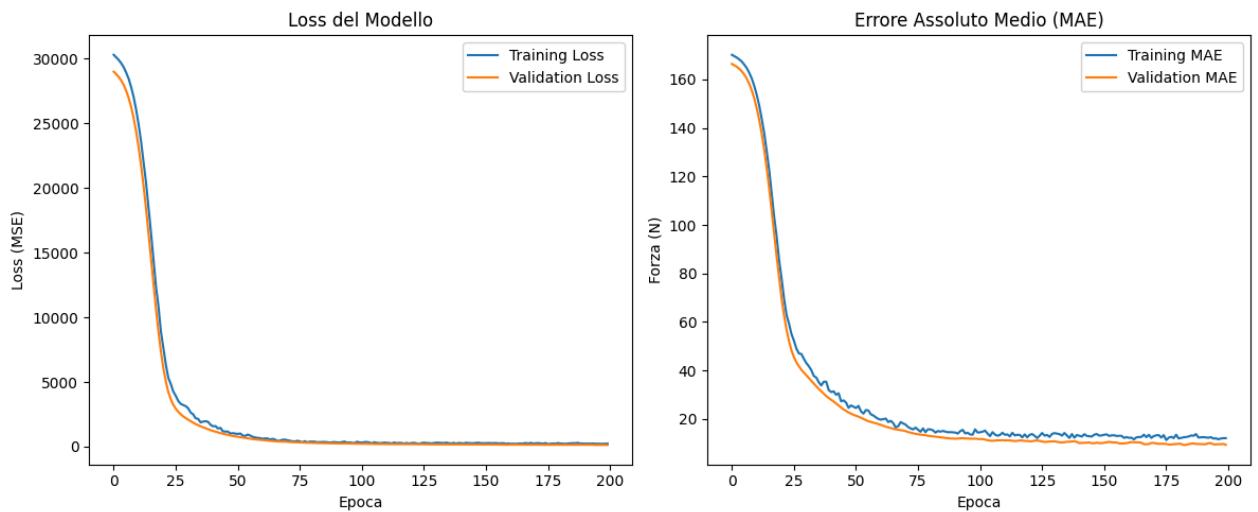


Figura 6.1 -

Tuttavia, l'analisi qualitativa delle predizioni (descritta più avanti) ha rivelato una tendenza del modello a "smussare" le stime, producendo valori plausibili ma con una variabilità ridotta rispetto ai dati reali.

Per investigare in modo quantitativo quali variabili fossero realmente predittive, è stato utilizzato un modello Gradient Boosting (LightGBM) [14] per la sua capacità di fornire una metrica chiara di "Feature Importance".

L'analisi ha prodotto un risultato, mostrato in **Figura 6.2** di gerarchia di importanza delle feature in relazione alla forza predetta e all'ordine di foratura temporale.

I risultati hanno dimostrato che il **NumeroForo** è il singolo predittore più potente, spiegando da solo la maggior parte della varianza della forza. Le complesse feature geometriche, sebbene correlate, hanno un contributo predittivo marginale una volta che l'effetto dell'usura è noto.

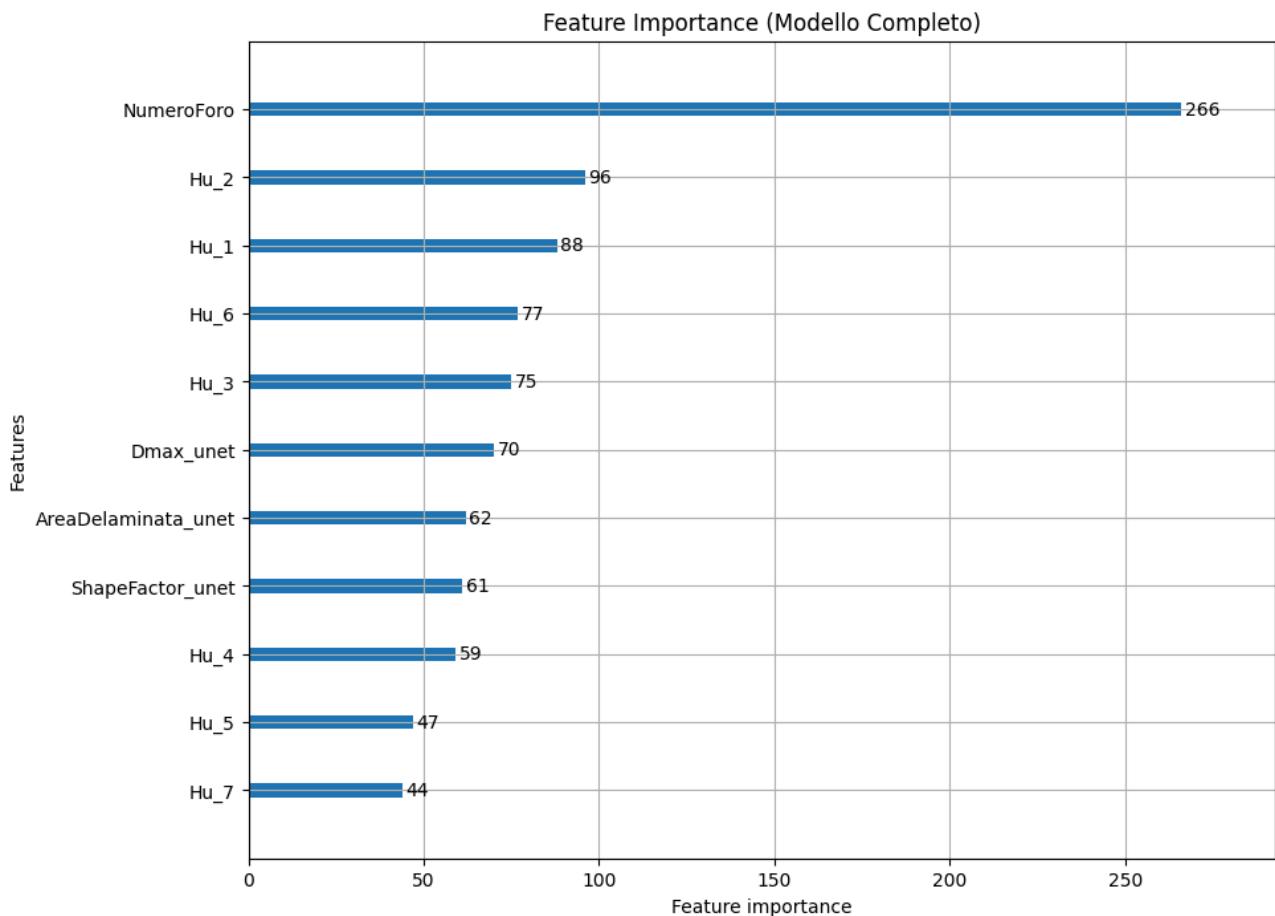


Figura 6.2 - Importanza delle feature per la predizione della Forza_N. Il NumeroForo spiega la quasi totalità della varianza, suggerendo che le feature geometriche aggiungono informazioni ridondanti.

Dal confronto emerge che il modello LGBM-Base, ottimizzando per l'errore globale, produce una stima che non cattura la dinamica locale del processo. L'MLP, pur essendo meno accurato su scala globale, sfrutta le informazioni secondarie delle feature geometriche per generare una stima localmente più plausibile e fisicamente coerente.

Conclusione: Questa analisi evidenzia un punto cruciale nella modellazione di dati sperimentali: le metriche globali possono essere fuorvianti. La valutazione qualitativa ha dimostrato la superiorità contestuale del modello MLP per questo specifico compito di imputazione. Pertanto, si è scelto di procedere utilizzando il **dataset completato tramite il modello MLP**.

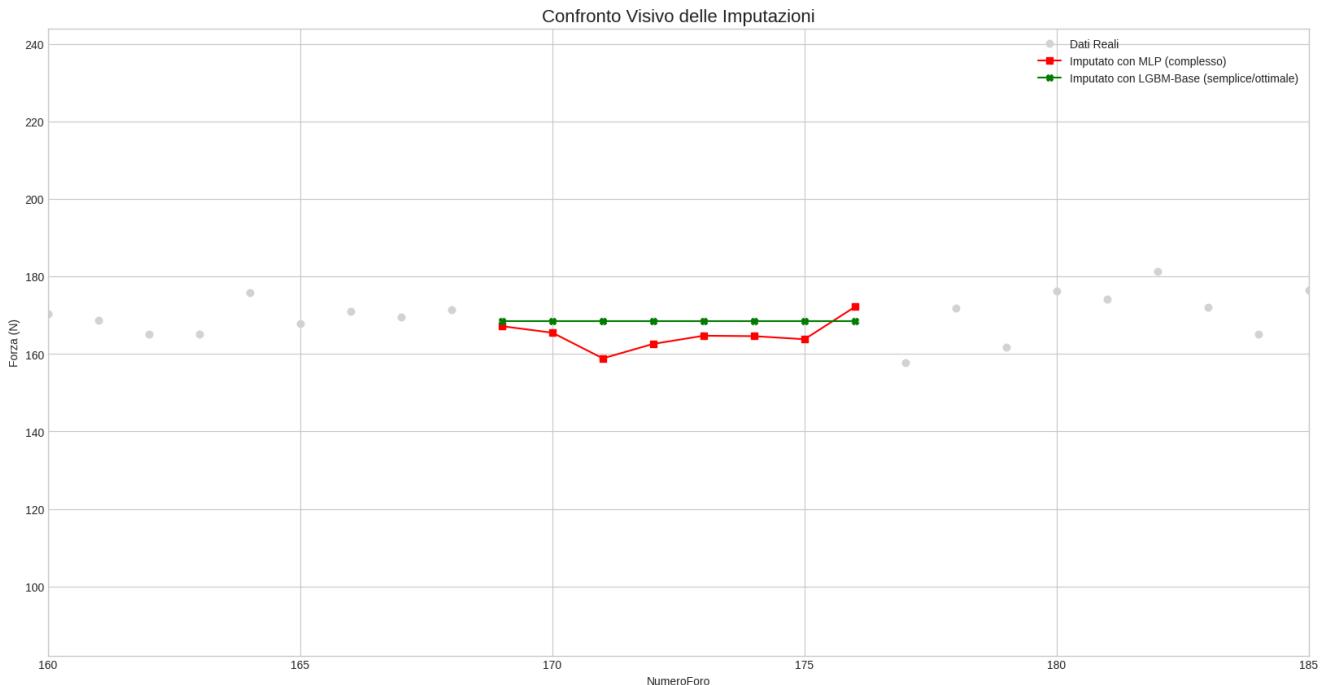


Figura 6.3. - Modellazione Sequenziale con Rete LSTM

Dataset di riferimento segnato come **LSTM_master_dataset.csv**

L'obiettivo di questa fase finale è predire l'evoluzione delle principali feature del danno (AreaDelaminata_mm2, Dmax_mm) in funzione della sequenza storica delle forze e delle feature precedenti, utilizzando il dataset completo e imputato.

Il dataset è stato trasformato in sequenze temporali tramite una finestra mobile (sliding window) di **n_steps=15**. Ogni campione di addestramento è quindi composto da 15 osservazioni consecutive, utilizzate per predire i valori del 16° foro.

Un'analisi preliminare, utilizzando un modello LSTM standard addestrato con la tradizionale Mean Squared Error (MSE) come funzione di perdita, ha evidenziato una limitazione nota: pur catturando il trend generale, il modello tendeva a "smussare" le predizioni, sottostimando sistematicamente i picchi e gli eventi estremi del danneggiamento.

Per superare questo limite e ottenere una stima non solo del valore medio ma anche dell'intervallo di incertezza, è stata implementata una Quantile Loss. A differenza della MSE, questa funzione di perdita permette di addestrare il modello a predire uno specifico quantile q della distribuzione futura del target.

Sono stati quindi addestrati tre modelli LSTM distinti, ciascuno ottimizzato per un quantile differente: $q=0.1$ (limite inferiore), $q=0.5$ (mediana) e $q=0.9$ (limite superiore). L'architettura impiegata è una rete LSTM a due layer con layer di Dropout per la regolarizzazione, come definito nel seguente snippet Keras in **Appendice D listato D.2**

Ogni modello è stato addestrato per 50 epochhe con un meccanismo di *early stopping* per prevenire l'overfitting, l'implementazione di questi modelli sequenziali è stata realizzata utilizzando il framework **TensorFlow** con la sua API Keras [15]

L'utilizzo della Quantile Loss ha permesso di costruire un **intervallo di predizione** che incapsula la maggior parte dei valori reali, fornendo una stima non solo del valore più probabile (la mediana), ma anche del potenziale "worst-case scenario" (il 90° percentile). I risultati per la metrica Dmax_mm sono mostrati in **Figura 6.4 - 6.5**.

--- Addestramento modello per quantile q=0.1 ---

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py
:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
super().__init__(**kwargs)
```

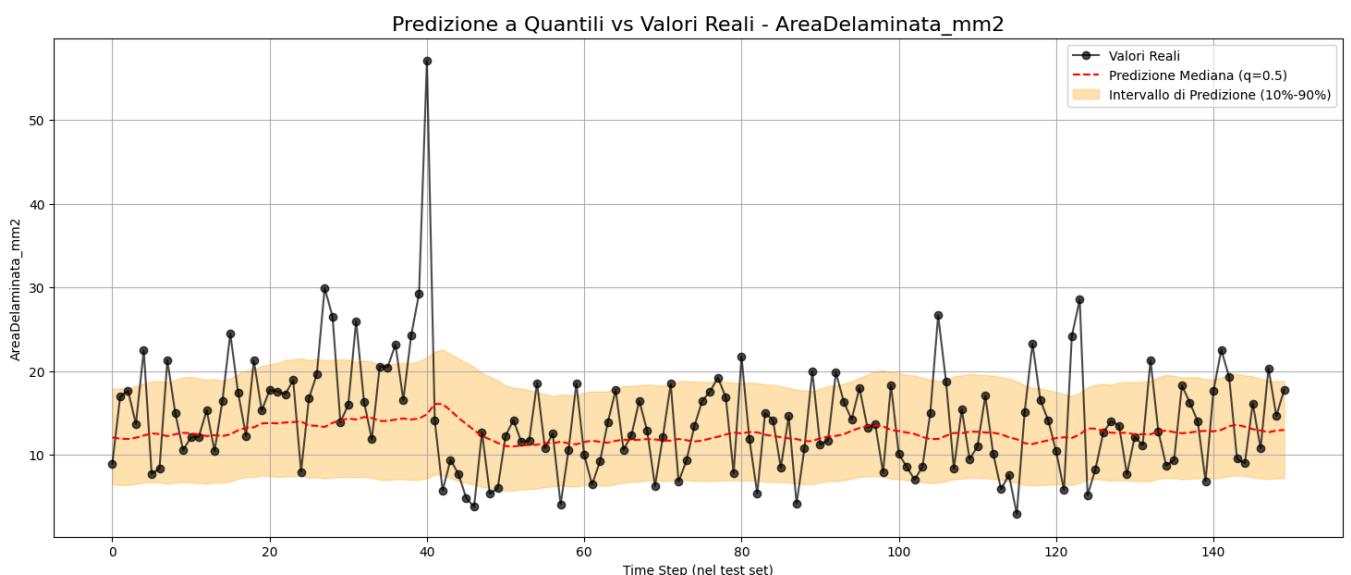
Addestramento per q=0.1 completato. Val Loss finale: 0.0149

--- Addestramento modello per quantile q=0.5 ---

Addestramento per q=0.5 completato. Val Loss finale: 0.0492

--- Addestramento modello per quantile q=0.9 ---

Addestramento per q=0.9 completato. Val Loss finale: 0.0225



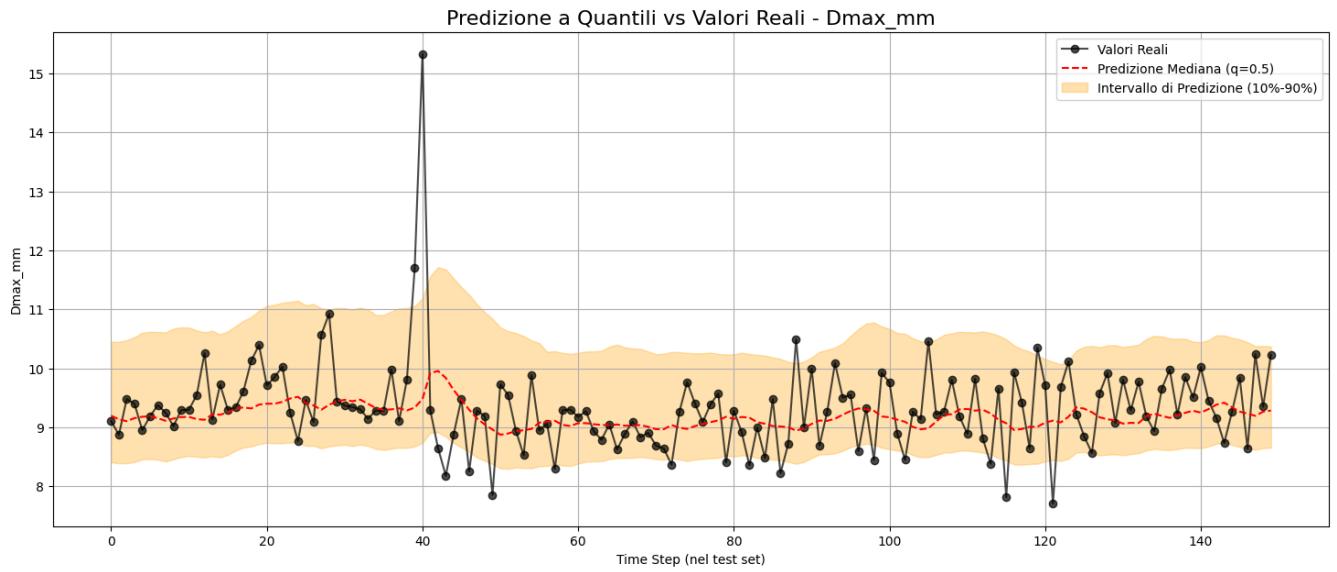


Figura 6.4 - 6.5: Confronto tra i valori reali (nero) e le predizioni a quantili per D_{max_mm} sul test set. La linea rossa tratteggiata rappresenta la stima della mediana ($q=0.5$), mentre l'area arancione definisce l'intervallo di predizione tra il 10° e il 90° percentile.

Analisi: Dal grafico emerge come l'intervallo di predizione (area arancione) riesca a contenere la quasi totalità dei dati reali, inclusi i picchi più elevati (es. al timestep 40). Il limite superiore dell'intervallo, corrispondente alla predizione del modello addestrato con $q=0.9$, si alza in corrispondenza dei picchi reali, agendo come un sistema di allarme precoce per eventi di danneggiamento anomali.

Questo approccio, quindi, non fornisce una semplice predizione puntuale (stima della media), ma un intervallo di predizione che funge da strumento per la stima del rischio e di conseguenza per il monitoraggio del processo in un contesto industriale. La pipeline ROIA nel suo complesso, ha dimostrato la sua capacità di gestire dati complessi e incompleti per produrre un modello predittivo degno di interesse per ricerche future.

Conclusioni

Questo lavoro di tesi ha affrontato il problema della quantificazione automatizzata e della previsione del danno da foratura in laminati compositi CFRP, sviluppando e valutando una pipeline computazionale end-to-end denominata ROIA (Radiograph-Only Integrated Analysis).

Sintesi dei Risultati:

L'architettura ROIA ha dimostrato la sua efficacia nell'orchestrare una sequenza di moduli di Machine Learning e Computer Vision per trasformare dati grezzi e inconsistenti in analisi predittive. I contributi e i risultati chiave possono essere così riassunti:

1. Acquisizione e Normalizzazione Dati: È stata sviluppata una pipeline robusta per il rilevamento (**YOLOv8**), l'ordinamento (K-Means bustrofedorico) e il cropping dei fori da scansioni radiografiche eterogenee. L'introduzione di un algoritmo di cropping con normalizzazione di scala si è rivelato un passaggio cruciale per garantire la coerenza metrica del dataset di input, risolvendo a monte le significative variazioni di DPI e risoluzione presenti nei dati grezzi.
2. Segmentazione del Danno: Il modello di segmentazione **UNet++**, addestrato con una strategia ibrida "foveale", ha dimostrato una capacità superiore nel delineare i dettagli del danno complesso rispetto a un fine-tuning standard. Questo approccio, che combina immagini a pieno campo con patch focalizzate sulle micro-strutture del danno, ha permesso di superare il plateau di apprendimento e di generare maschere di segmentazione di alta qualità.
3. Estrazione Quantitativa e Imputazione: L'implementazione di un algoritmo di auto-calibrazione basato sul diametro noto del foro ha garantito la conversione affidabile delle misure da pixel a unità fisiche. La successiva analisi diagnostica per l'imputazione dei dati di forza mancanti ha evidenziato come un modello più semplice, basato sull'usura (**Numeroforo**), fosse quantitativamente più accurato, ma un modello più complesso (**MLP**) producesse risultati qualitativamente più plausibili e dinamicamente coerenti, motivandone la scelta finale.
4. Modellazione Predittiva: Il modello finale, una rete neurale LSTM addestrata con una Quantile Loss, si è dimostrato capace non solo di prevedere il trend evolutivo del danno, ma anche di stimare un intervallo di predizione. Questa tecnica, a differenza della tradizionale MSE, permette di catturare la variabilità e gli eventi estremi del processo, fornendo una stima del rischio più completa e industrialmente rilevante.

Considerazioni Critiche e Limiti

Nonostante i risultati positivi, è importante sottolineare i limiti di questo studio. La performance di ogni modello a valle è intrinsecamente legata alla qualità del "golden set" di maschere utilizzato per il fine-tuning e ai dati iniziale delle scan. Sebbene l'approccio foveale abbia migliorato significativamente le performance, ulteriori raffinamenti, magari attraverso l'etichettatura di un numero maggiore di campioni complessi, potrebbero portare a ulteriori miglioramenti. Inoltre, il modello predittivo LSTM è stato addestrato e validato su un'unica serie storica. La sua capacità di generalizzare su provini con geometrie o materiali differenti rimane da investigare.

Il dataset limitato e la mancanza di validazione incrociata multiesemplare costituiscono il principale limite sperimentale.

Sviluppi Futuri

Sviluppi futuri di particolare interesse:

- l'estensione multimodale del paradigma proposto della pipeline ROIA; con integrazione nella pipeline di segnali acustici, termografici e vibrazionali in un framework auto-supervisionato. Costruendo una rappresentazione unificata del danno basata su spazi latenti coerenti tra domini fisici differenti.
- l'applicazione in tempo reale della pipeline ROIA, con l'ottimizzazione embedded dei modelli per piattaforme a basso consumo (Es. Jetson, Edge TPU ...) mediante pruning e quantizzazione adattiva, al fine di consentire applicazioni in-line su impianti produttivi reali.
- Dal punto di vista teorico, l'approccio RPOS si potrebbe formalizzare come morfismo continuo tra categorie di feature space \mathcal{C}_{rad} e \mathcal{C}_{opt} , aprendo la strada a una trattazione geometrica della trasferibilità dell'informazione nel contesto dei controlli non distruttivi.

Appendice

I pesi addestrati dei modelli **UNet++**, **MLP-LSTM** sono pubblicamente disponibili al seguente indirizzo:

R.Venturi *CFRP Weights* <https://huggingface.co/Riccardo99999/CFRP-ROIA-weights> [16]

Appendice A: Frammenti di Codice Salienti

Listato A.1

```
# ...
global_boxes = np.load("/content/global_boxes.npy") # Carica le annotazioni da Hough
for y in range(0, H, stride):
    for x in range(0, W, stride):
        tile_img = img[y:y+tile_size, x:x+tile_size]
        # ... (logica per mappare le coordinate globali a quelle locali del tile)
        # ... (salvataggio del tile e del relativo file .txt di etichette)
# ...
```

Listato A.2 - Estrazione patch ROI con bounding box YOLO

```
# Esegue la predizione sull'immagine singola passando il PERCORSO
model.predict(
    source=path,           # ← passiamo il **percorso** dell'immagine singola
    imgsz=(H,W),          # ← risoluzione nativa ⇒ nessun resize aggressivo di YOLO
    conf=0.79, iou=0.80, agnostic_nms=True, # Parametri di confidenza e NMS
    save_txt=True, save_conf=True, save=True, # Salva file .txt, confidenza e immagini box
    project="runs/detect", name=RUN, # Definisce dove salvare i risultati
    device=device, verbose=False # Specifica il device e sopprime output dettagliato
)
```

Appendice B

Listato B.1 - logica di clustering K-Means

```
# ... (logica di clustering K-Means)
for j, col in enumerate(cols):
    # ...
    idxs_sorted = sorted(
        idxs,
        key=lambda i: -((df.y1.iat[i] + df.y2.iat[i]) / 2) # Ordinamento per Y decrescente
```

```

if (j % 2 == 0) :           # se la colonna ha indice pari

else ((df.y1.iat[i] + df.y2.iat[i]) / 2): # o crescente altrimenti

```

Appendice C

Listato C.1 -Rilevamento Bordi e creazione mappa di Densità: Pre-segmentazione algoritmica

--- 1. Rilevamento Bordi e Creazione Mappa di Densità ---

```

# Migliora il contrasto locale per esaltare i dettagli del danno
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
img_enhanced = clahe.apply(img_gray)

# Rileva i bordi solo all'interno dell'area di interesse a "ciambella" (danno_search_area)
edges = cv2.Canny(img_enhanced, CANNY_THRESHOLD_LOW, CANNY_THRESHOLD_HIGH)
edges_in_roi = cv2.bitwise_and(edges, danno_search_area)

# "Addensa" i bordi per creare una mappa di densità delle feature di danno
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (KERNEL_SIZE, KERNEL_SIZE))
density_map = cv2.morphologyEx(edges_in_roi, cv2.MORPH_CLOSE, kernel,
iterations=ITERATIONS)

```

--- 2. Segmentazione e Pulizia ---

```

# Binarizza la mappa di densità per ottenere la maschera grezza del danno
_, potential_damage = cv2.threshold(density_map, DENSITY_THRESHOLD, 255,
cv2.THRESH_BINARY)

```

```

# Rimuove piccoli artefatti/rumore tramite analisi dei componenti connessi
num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(potential_damage)
final_damage_mask = np.zeros_like(potential_damage)
for i in range(1, num_labels):
    if stats[i, cv2.CC_STAT_AREA] >= MIN_AREA_PULIZIA_DANNO_PX:
        final_damage_mask[labels == i] = 255

```

--- 3. Assemblaggio Maschera Finale ---

Unisce la maschera del foro (classe 1) e la maschera del danno pulita (classe 2)

```

final_mask = np.zeros_like(img_gray, dtype=np.uint8)
final_mask[foro_mask == 255] = 1
final_mask[final_damage_mask == 255] = 2

```

... (*salvataggio della maschera finale*) ...

Listato C.2 - Creazione patch foveali

```

# --- Snippet 1: Creazione della Zona di Esclusione ---
hole_mask = np.uint8(mask == 1)
damage_mask = np.uint8(mask == 2)

# Dilata la maschera del foro per creare una "kill zone" invalicabile
kernel = cv2.getStructuringElement(
    cv2.MORPH_ELLIPSE,
    (Config.BLACKOUT_MARGIN * 2, Config.BLACKOUT_MARGIN * 2)
)
blackout_zone = cv2.dilate(hole_mask, kernel, iterations=1)

```

Listato C.3 - Preselezione e campionamento delle patches

```

# --- Snippet 2: Applicazione del Blackout ---
surviving_damage_mask = damage_mask.copy()
# Annulla tutti i pixel di danno che cadono all'interno della zona
di esclusione
surviving_damage_mask[blackout_zone > 0] = 0

```

Listato C.4 - Estrazione delle Patch

```

# --- Snippet 3: Campionamento e Estrazione delle Patch ---
surviving_points_yx = np.argwhere(surviving_damage_mask > 0)
# Campiona casualmente N punti (es. 50) dalle aree di danno puro
num_to_sample = min(Config.DESIRED_PATCHES_PER_IMAGE,
len(surviving_points_yx))
sampled_indices = np.random.choice(len(surviving_points_yx),
size=num_to_sample, replace=False)
for i, idx in enumerate(sampled_indices):
    y, x = surviving_points_yx[idx]

```

```
# ... (logica per ritagliare la patch 128x128 centrata su  
(y,x)) ...
```

Appendice D

Listato D.1 - Autocalibrazione

```
# --- Snippet 1: Logica Chiave dell'Auto-Calibrazione ---  
import cv2  
import numpy as np  
  
DIAMETRO_REALE_MM = 6.0  
AREA_NOMINALE_MM2 = np.pi * (DIAMETRO_REALE_MM / 2)**2  
  
# All'interno della funzione extract_features_self_calibrated(mask_path):  
# 1. Isola la maschera del foro (pixel con valore 1)  
hole_mask = (mask == 1).astype(np.uint8)  
area_foro_px = np.count_nonzero(hole_mask)  
  
# Se non c'è foro, non è possibile calibrare.  
if area_foro_px < 100:  
    return None  
  
# 2. Calcola la scala (pixel2/mm2) basandosi sull'area del foro segmentato.  
# Questo è il cuore dell'auto-calibrazione.  
px_per_mm_squared = area_foro_px / AREA_NOMINALE_MM2  
px_per_mm = np.sqrt(px_per_mm_squared)  
  
# 3. Usa la scala calcolata per convertire le misure del danno.  
damage_mask = (mask == 2).astype(np.uint8)  
area_danno_px = np.count_nonzero(damage_mask)  
area_danno_mm2 = area_danno_px / px_per_mm_squared  
# ... e così via per Dmax e le altre feature metriche.
```

Listato D.2 - keras modello sequenziale

```
# Funzione per La Quantile Loss  
def quantile_loss(q, y_true, y_pred):  
    e = y_true - y_pred  
    return tf.keras.backend.mean(tf.keras.backend.maximum(q * e, (q  
    - 1) * e), axis=-1)
```

```

for q in QUANTILES:
    print(f"\n--- Addestramento modello per quantile q={q} ---")

    # Costruiamo un nuovo modello per ogni quantile
    model = Sequential([
        LSTM(units=100, activation='relu',
            input_shape=(X_train.shape[1], X_train.shape[2]),
            return_sequences=True),
        Dropout(0.2),
        LSTM(units=50, activation='relu'),
        Dropout(0.2),
        Dense(units=y_train.shape[1])
    ])

    # Usiamo la nostra loss customizzata
    model.compile(optimizer='adam', loss=lambda y_true, y_pred:
        quantile_loss(q, y_true, y_pred))

    early_stopping =
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
        restore_best_weights=True)

    history = model.fit(
        X_train, y_train,
        epochs=50, # 50 epoche sono sufficienti per convergere
        batch_size=32,
        validation_data=(X_test, y_test),
        callbacks=[early_stopping],
        verbose=0 # Output pulito
    )

```

Appendice E: Repository del Codice Sorgente

Il codice completo sviluppato per questa tesi, comprensivo di tutti i notebook di analisi, script di utility e pesi dei modelli addestrati, è stato archiviato e reso permanentemente accessibile tramite la piattaforma Zenodo per garantire la massima riproducibilità. L'archivio è navigabile seguendo la struttura di cartelle descritta di seguito.

Venturi, R. (2024). *Codice sorgente per lo studio comparativo di pipeline di deep learning per la valutazione automatica del danno da foratura in compositi CFRP* (Versione v1.0.0) [Software]. Zenodo. <https://doi.org/10.5281/zenodo.17250427>

Guida alla Struttura del Repository

E.1 – Pipeline di Acquisizione Dati (Rif. Capitolo 2)

Percorso nel Repository: 01_data_preprocessing/

- **YoloRilevazioneForiVersione1_3.ipynb:** Notebook principale per il rilevamento fori con YOLOv8, l'ordinamento bustrofedico e il cropping normalizzato delle patch.
- **cuciturascan_separate_orizzontale.py:** Script per l'unione di scansioni radiografiche divise.

E.2 – Framework Cross-Modale (RPOS) (Rif. Capitolo 3)

Percorso nel Repository: 01_data_preprocessing/

- **Homography1.ipynb:** Notebook esplorativo relativo alla *Pipeline A* (approccio cross-modale poi scartato), che implementa la registrazione di immagini tramite omografia locale adattiva.

E.3 – Pipeline di Segmentazione del Danno (ROIA) (Rif. Capitolo 4)

Percorso nel Repository: 02_segmentation_unet++/

- **MAsk_Factory.ipynb:** Implementa la "Mask Factory" algoritmica per la creazione del dataset di pre-training.
- **Unet++V3.ipynb:** Contiene la logica per il fine-tuning incrementale e l'addestramento ibrido "foveale".

E.4 – Modulo di Estrazione Feature (Rif. Capitolo 5)

Percorso nel Repository: 03_feature_extraction/

- **EstrattoreFeatures.ipynb:** Script che implementa la strategia di calibrazione a due livelli (a monte e a valle) e l'algoritmo di **auto-calibrazione per-immagine** per l'estrazione di feature metricamente robuste.

E.5 – Modulo di Modellazione Predittiva (Rif. Capitolo 6)

Percorso nel Repository: 04_predictive_modeling_lstm/

- **sequential_finding_... .ipynb:** Notebook finale che include l'analisi diagnostica per l'imputazione dei dati di forza e l'implementazione del modello predittivo **LSTM** con Quantile Loss.

Bibliografia Essenziale

- [1] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Netw.*, vol. 22, no. 5–6, pp. 544–557, 2009.
- [2] C. R. Harris and others, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
- [3] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 03, pp. 90–95, 2007.
- [4] G. Jocher, A. Chaurasia, and J. Qiu, "YOLOv8: Open-source object detection architecture." 2023. [Online]. Available: <https://docs.ultralytics.com>
- [5] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, Springer, Cham, 2018, pp. 3–11.
- [6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [7] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sept. 11, 2020, *arXiv*: arXiv:1905.11946. doi: 10.48550/arXiv.1905.11946.
- [8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Springer, Cham, 2015, pp. 234–241.
- [9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [10] C. Olah, "Understanding LSTMs." 2015. [Online]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [11] G. Bradski, "The OpenCV Library," *Dr Dobbs J. Softw. Tools*, 2000.
- [12] F. Pedregosa and others, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [13] M. Melis, "Processo di foratura: influenza dell'usura sul danneggiamento di laminati compositi," Università degli Studi di Cagliari, 2018.
- [14] G. Ke and others, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.
- [15] M. Abadi and others, "TensorFlow: A System for Large-Scale Machine Learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 2016.
- [16] RiccardoVenturi, "CFRP-ROIA-weights", doi: 10.57967/HF/6729.