

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN
INGEGNERIA E SCIENZA INFORMATICHE

SEGMENTAZIONE DI EMISSIONI RADIO DIFFUSE E SORGENTI ESTESE IN IMMAGINI RADIOASTRONOMICHE MEDIANTE TRANSUNET: ANALISI DELLE PRESTAZIONI E DELLA SCALABILITÀ

Tesi di laurea in High Performance Computing

Relatore

Prof. Moreno Marzolla

Presentata da

Alice Beccati

Correlatore

Dott.ssa Nicoletta Sanvitale

Dott. Claudio Gheller

Sessione di ottobre 2026

Anno Accademico 2025/2026

Sommario

La tesi riguarda lo studio di TUNA (*TransUNet Network for Radio Astronomy*), un modello di deep learning basato sull'architettura TransUNet, originariamente sviluppata per la segmentazione di immagini mediche e adattata alla segmentazione di emissioni radio diffuse ed estese nelle immagini radioastronomiche. Le immagini ottenute dalle osservazioni radioastronomiche possono presentare un'elevata presenza di rumore e diversi artefatti, rendendo difficile distinguere le emissioni radio di interesse. TUNA viene utilizzato per individuare automaticamente queste regioni nelle immagini, restituendo una maschera di confidenza che indica, per ogni pixel, la probabilità che appartenga a un'emissione. L'obiettivo del lavoro è studiare il comportamento del modello durante l'addestramento su più GPU e valutare come diverse precisioni numeriche possano influire sui tempi di esecuzione.

Dopo una descrizione del contesto radioastronomico e delle principali caratteristiche dell'architettura utilizzata, viene presentato il dataset impiegato per l'addestramento e vengono descritte le procedure di preprocessing e le metriche utilizzate per valutare il modello. Attraverso il confronto tra diversi addestramenti, effettuati variando il learning rate e mantenendo invariati gli altri principali parametri, viene quindi individuata una configurazione di riferimento sulla base delle metriche di segmentazione ottenute. Il modello selezionato viene successivamente applicato anche ad alcune immagini osservative reali, così da valutarne il comportamento al di fuori delle immagini simulate utilizzate nel dataset.

A partire da questa configurazione, viene valutato il comportamento del modello al variare delle risorse computazionali utilizzate. Sono stati analizzati i casi di strong scaling e weak scaling, considerando i tempi di esecuzione, lo speedup e l'efficienza al variare del numero di GPU. È stato inoltre valutato l'effetto della parallelizzazione sulla qualità delle predizioni attraverso il confronto delle metriche di segmentazione.

Infine, viene studiato l'utilizzo della mixed precision durante l'addestramento, confrontando le rappresentazioni FP32, FP16 e BF16. Anche per queste configurazioni vengono considerate sia le metriche di segmentazione sia le prestazioni computazionali, con l'obiettivo di valutare il rapporto tra qualità dei risultati e costo dell'addestramento.

Indice

Sommario **2**

1 Introduzione **5**
1.1 Contesto scientifico e problema affrontato 5
1.2 Obiettivi della tesi 6

2 Contesto scientifico e tecnologico **8**
2.1 Elementi di radioastronomia 8
2.2 Machine learning e segmentazione delle immagini 10
2.3 Il modello TUNA 11
2.4 High Performance Computing e calcolo parallelo 12
2.5 Mixed precision 14

3 Dataset e ambiente computazionale **16**
3.1 Ambiente computazionale e strumenti utilizzati 16
3.2 Dataset: Release 1.4 e Release 1.5 17
3.3 Preparazione dei dati 18
3.4 Configurazione del modello 20
3.5 Metriche di valutazione 20

4 Selezione e validazione del modello **22**
4.1 Addestramento su Release 1.4 22
4.2 Confronto tra i learning rate 22
4.2.1 Analisi delle loss 22
4.2.2 Metriche sul validation set 24
4.3 Valutazione sul test set 26
4.4 Cross-validation 27
4.5 Applicazione a immagini osservative 28

5 Parallelizzazione e analisi delle prestazioni **32**
5.1 Configurazioni per l'addestramento distribuito 32
5.2 Analisi delle loss 33

5.3 Effetto della parallelizzazione sulle metriche 35
5.3.1 Strong scaling 35
5.3.2 Weak scaling 36
5.4 Analisi delle prestazioni computazionali 38
5.4.1 Strong scaling 38
5.4.2 Weak scaling 40
5.5 Valutazione complessiva 42

6 Mixed precision **43**
6.1 Configurazioni sperimentali 43
6.2 Confronto delle precisioni 43
6.2.1 Metriche di segmentazione 43
6.2.2 Tempi di esecuzione 45
6.3 Valutazione complessiva 45

Conclusioni **47**

A Script utilizzati **50**
A.1 Addestramento con diversi learning rate 50
A.2 Inferenza dei modelli multi-GPU 51
A.3 Calcolo delle metriche dei modelli multi-GPU 53

Bibliografia **56**

Capitolo 1

Introduzione

1.1 Contesto scientifico e problema affrontato

L'astronomia radio studia l'emissione di onde radio provenienti da sorgenti astronomiche come galassie, ammassi di galassie e altri oggetti celesti. L'analisi di tali fenomeni richiede l'elaborazione di immagini radioastronomiche ottenute mediante osservazioni effettuate con radiotelescopi.

Le moderne osservazioni radio producono quantità sempre maggiori di dati, rendendo necessario lo sviluppo di strumenti automatici per la loro analisi. I metodi tradizionali basati su procedure manuali risultano infatti sempre meno adatti a gestire dataset di grandi dimensioni.

Per questo motivo negli ultimi anni si è diffuso l'utilizzo di tecniche di machine learning e deep learning per l'elaborazione di immagini astronomiche, con l'obiettivo di automatizzare attività di riconoscimento, classificazione e segmentazione delle strutture presenti nelle osservazioni. Il machine learning serve a costruire, a partire dai dati, un modello capace di associare un input a un output, mentre il deep learning ne rappresenta un sottoinsieme basato su reti neurali composte da più livelli. La segmentazione consiste invece nell'assegnare una classe alle diverse regioni di un'immagine.

Questa tesi si inserisce in questo contesto e riguarda lo studio di TUNA (*TransUNet Network for Radio Astronomy*), un modello di deep learning sviluppato per la segmentazione automatica di emissioni diffuse ed estese in immagini radioastronomiche [1].

Una **sorgente radio estesa** è un oggetto celeste la cui emissione occupa una regione del cielo più ampia della risoluzione angolare del radiotelescopio. A differenza di una sorgente puntiforme, che appare come un singolo punto perché lo strumento non riesce a distinguerne la struttura, presenta quindi una struttura spazialmente risolta, come getti e lobi associati a galassie attive.

Una **sorgente radio diffusa** è un'emissione radio molto debole (a bassa brillanza) e distribuita su grandi regioni, senza bordi netti o una forma compatta, spesso associata a gas e campi magnetici negli ammassi di galassie. La segmentazione automatica di emissioni da

sorgenti radio rappresenta un problema particolarmente complesso. Le strutture di interesse sono infatti caratterizzate da una debole intensità del segnale e da forme spesso irregolari, che possono estendersi su ampie regioni dell'immagine. Esempi di sorgenti radio diffuse e di radiogalassie sono riportati in Figura 1.1.

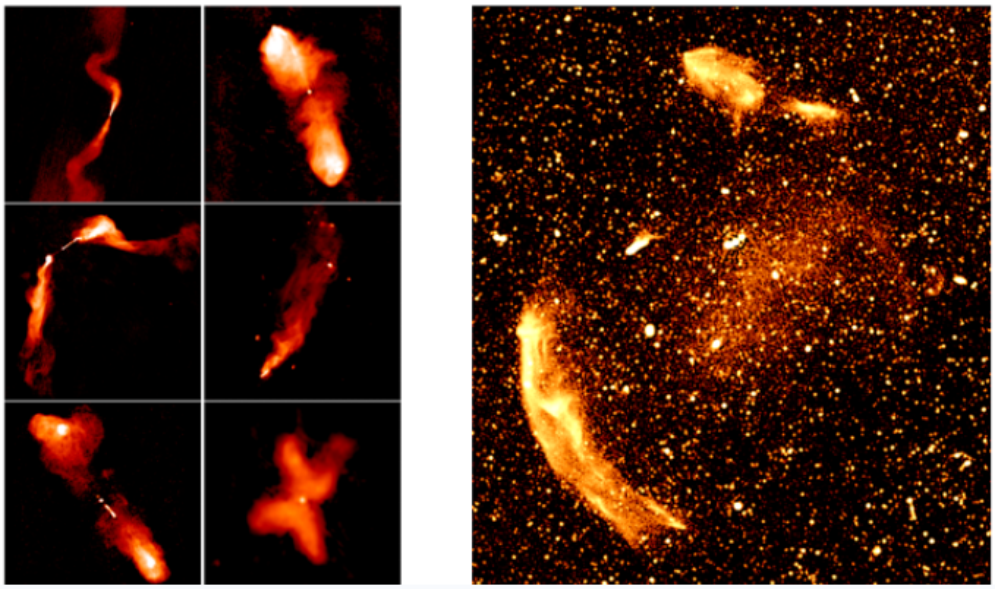


Figura 1.1: Esempi di sorgenti radioastronomiche: a sinistra, radiogalassie caratterizzate da differenti morfologie di getti e lobi; a destra, un'emissione radio diffusa costituita da un alone radio centrale e da due relitti radio periferici. Immagini a sinistra adattate da Hardcastle e Croston [2]; immagine a destra adattata da Knowles et al. [3].

La difficoltà del problema è ulteriormente aumentata dalla presenza di rumore e di artefatti introdotti durante il processo di osservazione e di elaborazione dei dati. Inoltre, nelle immagini radioastronomiche sono frequentemente presenti sorgenti puntiformi molto luminose e radiogalassie, che possono contribuire a rendere più difficile la distinzione tra segnale di interesse e sorgenti indesiderate.

Queste caratteristiche possono limitare l'accuratezza del modello, con regioni classificate erroneamente come emissioni diffuse o estese, oppure con la mancata individuazione di sorgenti di interesse. La riduzione di tali errori rappresenta uno degli aspetti più importanti nella valutazione delle prestazioni di una rete di segmentazione, poiché influisce direttamente sulla qualità dei risultati ottenuti.

1.2 Obiettivi della tesi

L'obiettivo della tesi è lo studio e l'ottimizzazione delle prestazioni di TUNA, estendendo l'applicazione originaria, rivolta alla segmentazione delle emissioni radio diffuse, anche alle sorgenti estese, come le radiogalassie.

Il modello combina un'architettura U-Net con componenti basate sui Transformer [1], descritti più nel dettaglio nel Capitolo 2. A causa della sua complessità e delle dimensioni dei dataset utilizzati, il processo di addestramento richiede significative risorse computazionali ed è stato eseguito sul supercomputer Leonardo di CINECA.

Una prima fase del lavoro ha previsto lo studio della letteratura relativa alla radioastronomia, alla segmentazione semantica, alle architetture U-Net, ai Transformer e a TransUNet, con particolare attenzione all'adattamento del modello al contesto radioastronomico.

La parte sperimentale si è concentrata sulla preparazione dei dati e sull'esecuzione di diversi addestramenti, variando alcuni dei principali iperparametri del modello. In particolare, sono stati confrontati differenti valori del learning rate e diversi checkpoint del processo di training, analizzando sia l'andamento delle funzioni di loss sia le metriche calcolate sulle maschere di segmentazione. L'obiettivo è stato individuare la configurazione in grado di offrire il miglior equilibrio tra le prestazioni considerate.

Una seconda parte dell'analisi è stata dedicata alle prestazioni computazionali e alla scalabilità del processo di addestramento. Il modello è stato eseguito utilizzando un numero crescente di GPU, verificando la consistenza delle metriche rispetto alla configurazione di riferimento su una singola GPU e quantificando l'incremento delle prestazioni ottenuto mediante la parallelizzazione. Sono stati quindi analizzati lo strong scaling e il weak scaling e sono stati calcolati speedup ed efficienza parallela, al fine di individuare il miglior compromesso tra tempo di esecuzione, utilizzo delle risorse computazionali e stabilità dei risultati.

Infine, è stato analizzato l'impiego della mixed precision, confrontando i formati FP16 (*16-bit Floating Point*) e BF16 (*Brain Floating Point a 16 bit*) con la configurazione di riferimento, eseguita in FP32 (*32-bit Floating Point*) su una singola GPU. Sono state esaminate sia la qualità delle predizioni sia i tempi di addestramento, così da valutare i vantaggi computazionali offerti dalla precisione ridotta e il loro eventuale impatto sulle prestazioni del modello.

Capitolo 2

Contesto scientifico e tecnologico

2.1 Elementi di radioastronomia

La radioastronomia è il ramo dell'astronomia che studia gli oggetti celesti attraverso la radiazione emessa alle frequenze radio. Nello spettro elettromagnetico, questa regione si estende indicativamente da circa 20 kHz a 300 GHz e corrisponde a lunghezze d'onda molto maggiori rispetto a quelle della luce visibile (si veda Figura 2.1). L'astronomia ottica osserva invece principalmente la regione visibile dello spettro, caratterizzata da frequenze dell'ordine delle centinaia di terahertz, insieme alle bande vicine dell'infrarosso e dell'ultravioletto. L'osservazione alle frequenze radio permette quindi di studiare fenomeni che possono risultare invisibili o difficili da rilevare in altre bande dello spettro elettromagnetico. Il segnale radio può avere origini fisiche differenti, poiché la radiazione osservata può essere prodotta da diversi meccanismi di emissione.

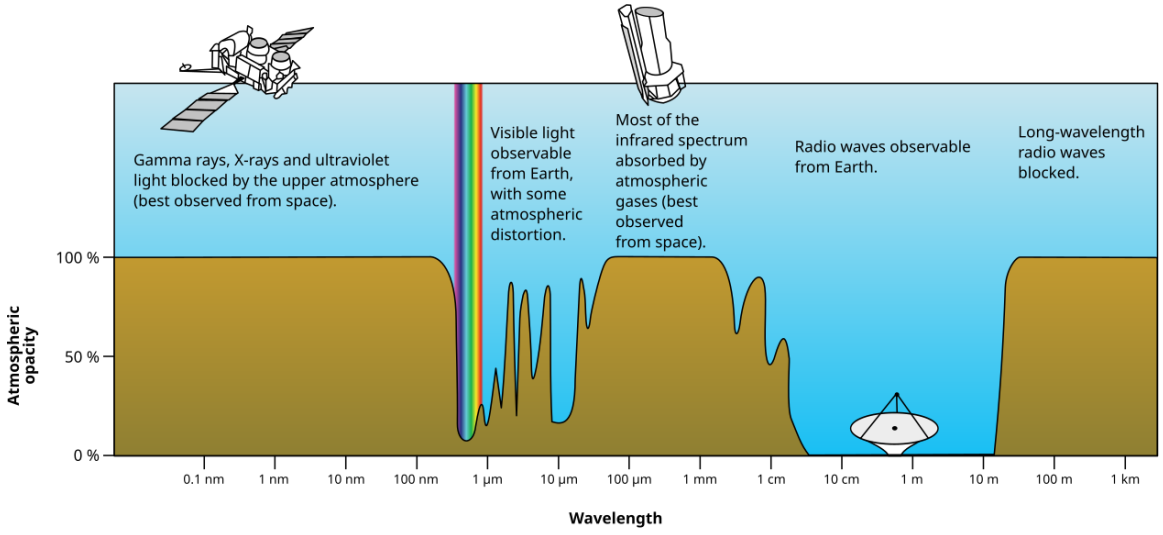


Figura 2.1: Opacità dell'atmosfera terrestre alle diverse lunghezze d'onda della radiazione elettromagnetica. Fonte: NASA [4].

Le osservazioni radio vengono effettuate mediante radiotelescopi. Un esempio di radiotelescopio a singola parabola è riportato in Figura 2.2 a sinistra. Tuttavia, a causa delle grandi lunghezze d'onda osservate, ottenere immagini ad alta risoluzione con una singola antenna richiederebbe strumenti di dimensioni estremamente elevate. Per questo motivo vengono spesso utilizzati sistemi interferometrici, composti da più antenne separate tra loro, che consentono di ottenere una risoluzione angolare equivalente a quella di un telescopio avente dimensioni pari alla massima distanza tra le antenne:

$$\theta \approx \frac{\lambda}{B_{\text{max}}}$$

dove θ è la risoluzione angolare (più è piccola, più l'immagine è nitida e ricca di dettagli), λ è la lunghezza d'onda della radiazione osservata e B_{max} la distanza massima tra le antenne più lontane dell'array.

Un esempio di questo tipo di sistema è LOFAR (*LOw-Frequency ARray*), mostrato nella parte destra della Figura 2.2. Questo approccio prende il nome di interferometria o *aperture synthesis*.



Figura 2.2: Due tipologie di strumentazione radioastronomica. A sinistra, l'antenna parabolica del Sardinia Radio Telescope; a destra, il Superterp, nucleo centrale dell'interferometro LOFAR. Immagine a sinistra adattata da caprowsky [5], CC BY 3.0; immagine a destra adattata da van Haarlem et al. [6].

L'interferometro non acquisisce direttamente un'immagine del cielo, ma misura la correlazione tra i segnali ricevuti da diverse coppie di antenne. Tali misure, chiamate visibilità complesse, rappresentano campioni della trasformata di Fourier della distribuzione di luminosità della sorgente nel dominio delle frequenze spaziali. La ricostruzione dell'immagine nel dominio spaziale (x, y) avviene idealmente applicando la trasformata di Fourier inversa alle visibilità misurate. Poiché non è possibile disporre di un numero infinito di antenne, la ricostruzione interferometrica delle immagini radioastronomiche è fondamentalmente limitata dal campionamento incompleto del piano di Fourier (il cosiddetto piano uv). Tale incompletezza genera significativi artefatti nelle immagini, errori nella stima del flusso e un generale degrado nel recupero dell'emissione diffusa [7].

2.2 Machine learning e segmentazione delle immagini

Il machine learning comprende metodi che permettono di costruire modelli in grado di apprendere una relazione tra dati di input e output a partire da esempi osservati. Nel caso dell'apprendimento supervisionato, il dataset è costituito da coppie (x_i, y_i), dove x_i rappresenta un dato di input e y_i il corrispondente valore o classe corretta, detta anche *target* o *label*. Durante l'addestramento, il modello produce una predizione a partire da x_i e confronta il risultato ottenuto con y_i mediante una funzione di loss, che quantifica l'errore commesso. I parametri del modello vengono quindi aggiornati iterativamente con l'obiettivo di minimizzare tale funzione e apprendere una relazione che sia in grado di generalizzare anche a dati non utilizzati durante l'addestramento [8].

Il deep learning rappresenta una particolare classe di tecniche di machine learning basata sull'utilizzo di reti neurali composte da più livelli. Una rete neurale è un modello composto da unità di elaborazione interconnesse, i cui parametri vengono appresi durante l'addestramento. Queste reti sono particolarmente adatte all'elaborazione delle immagini, poiché permettono di apprendere automaticamente caratteristiche a diversi livelli di complessità direttamente dai dati. Un'immagine digitale può essere rappresentata come una matrice di pixel, ciascuno associato a uno o più valori numerici che ne descrivono l'intensità. L'elaborazione automatica di queste informazioni costituisce uno dei principali ambiti della visione artificiale.

A seconda dell'obiettivo, un'immagine può essere analizzata a diversi livelli. Nei problemi di classificazione, il modello associa generalmente un'unica classe all'intera immagine. La segmentazione, invece, ha lo scopo di individuare anche la posizione delle diverse regioni di interesse. In particolare, nella segmentazione semantica viene assegnata una classe a ciascun pixel dell'immagine, ottenendo come risultato una maschera in cui ogni pixel indica la classe assegnata alla corrispondente posizione dell'immagine di input.

Nel caso delle immagini radioastronomiche considerate, il problema può essere formulato come una segmentazione semantica binaria: ciascun pixel deve essere classificato come appartenente a una sorgente radio oppure al background. Durante l'addestramento, le immagini vengono quindi fornite alla rete insieme alle corrispondenti maschere di *ground truth*, che rappresentano la classificazione corretta dei pixel e costituiscono il riferimento rispetto al quale viene calcolato l'errore della predizione.

Per valutare la capacità del modello di generalizzare a dati non utilizzati direttamente durante l'addestramento, i dati vengono normalmente suddivisi in *training set*, *validation set* e *test set*. Il training set viene utilizzato per aggiornare i parametri della rete, mentre il validation set permette di controllarne il comportamento durante lo sviluppo e di confrontare differenti configurazioni. Il test set viene infine utilizzato per valutare le prestazioni del modello su dati mantenuti separati dall'addestramento. Questa distinzione è importante anche per individuare fenomeni di *overfitting*, che si verificano quando il modello si adatta eccessivamente ai dati di training e perde capacità di generalizzazione [8].

2.3 Il modello TUNA

TransUNet è un'architettura di deep learning originariamente sviluppata per la segmentazione di immagini mediche e basata sulla combinazione di Convolutional Neural Networks (CNN), reti neurali particolarmente adatte all'elaborazione delle immagini, e Transformer [9]. L'obiettivo di questa architettura è combinare la capacità delle CNN di estrarre caratteristiche locali, tra cui dettagli spaziali e morfologici ad alta risoluzione, con quella dei Transformer di modellare relazioni globali tra le diverse regioni dell'immagine.

La struttura di TransUNet deriva da U-Net, una delle architetture più diffuse per la segmentazione di immagini. U-Net è composta da una fase di encoding e una fase di decoding. Durante l'encoding, l'immagine viene progressivamente trasformata in rappresentazioni sempre più astratte mediante operazioni convoluzionali. Nella fase di decoding, tali informazioni vengono utilizzate per ricostruire una mappa di segmentazione alla risoluzione originale. Un elemento caratteristico dell'architettura è l'utilizzo delle skip connections, che collegano direttamente encoder e decoder consentendo di preservare informazioni spaziali importanti durante la ricostruzione dell'immagine [10].

Per migliorare la capacità del modello di comprendere la struttura complessiva dell'immagine, TransUNet integra un modulo Transformer all'interno dell'encoder. I Transformer sono stati inizialmente sviluppati nell'ambito dell'elaborazione del linguaggio naturale e si basano sul meccanismo di self-attention, che permette di valutare contemporaneamente le relazioni tra tutti gli elementi dell'input [11]. Uno dei loro principali vantaggi è la possibilità di elaborare l'intero input in parallelo.

Nel caso delle immagini, le caratteristiche estratte dalla componente convoluzionale vengono suddivise in una sequenza di token, cioè elementi che rappresentano diverse regioni dell'immagine, e convertite in embedding numerici, ossia rappresentazioni vettoriali che costituiscono l'ingresso del Transformer. Poiché il meccanismo di self-attention non contiene informazioni implicite sulla posizione degli elementi, vengono aggiunti opportuni positional encoding che permettono al modello di mantenere la disposizione spaziale originale dei dati [12]. Attraverso il meccanismo di multi-head self-attention, il Transformer è in grado di apprendere simultaneamente diversi tipi di relazione tra le varie regioni dell'immagine.

Le informazioni elaborate dal Transformer vengono infine inviate al decoder di U-Net, che genera la maschera finale di segmentazione combinando le rappresentazioni apprese durante l'encoding con le informazioni provenienti dalle skip connections.

Durante l'addestramento, la maschera prodotta viene confrontata con la corrispondente *ground truth*. La Cross-Entropy Loss misura l'errore tra le probabilità predette dal modello e la classe corretta, mentre la Dice Loss valuta il grado di sovrapposizione tra le due maschere. La loss totale utilizzata da TUNA combina le due componenti con lo stesso peso:

$$L = 0{,}5L_{\text{CE}} + 0{,}5L_{\text{Dice}}. \tag{2.1}$$

La struttura generale dell'architettura TransUNet utilizzata nel progetto TUNA è riportata in Figura 2.3.

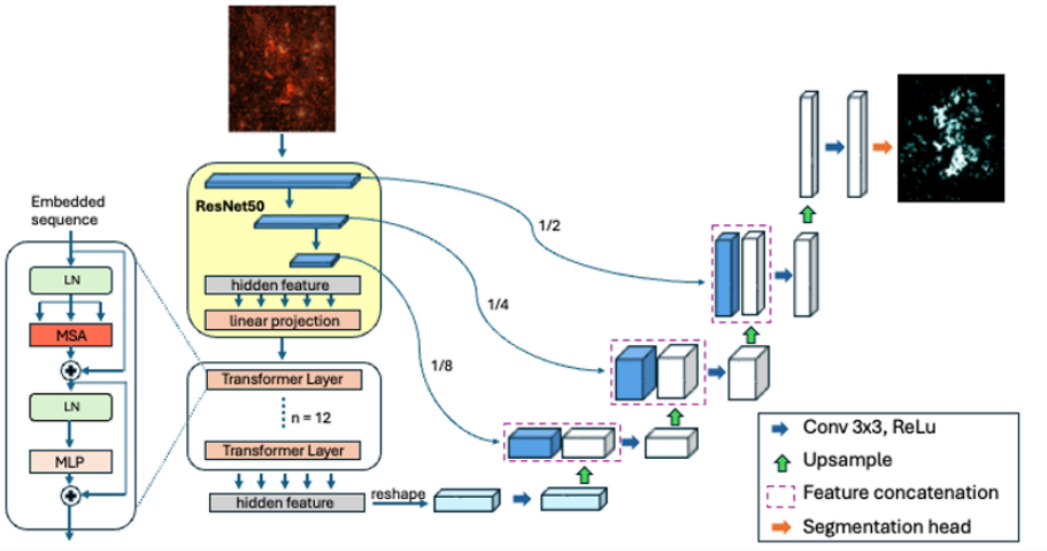


Figura 2.3: Rappresentazione schematica della deep neural network TUNA: estrazione delle caratteristiche mediante CNN e Transformer e successiva ricostruzione della maschera tramite decoder e skip connections. Riprodotta da Sanvitale et al. [1].

2.4 High Performance Computing e calcolo parallelo

L'High Performance Computing (HPC) comprende l'insieme di architetture e tecniche utilizzate per affrontare problemi caratterizzati da un elevato costo computazionale, sfruttando contemporaneamente più unità di elaborazione. Il calcolo parallelo permette di distribuire il lavoro tra più unità di calcolo, che possono operare contemporaneamente, riducendo i tempi di esecuzione e rendendo possibile l'elaborazione di problemi di dimensioni maggiori [13].

Le prestazioni di un programma parallelo dipendono da quanto efficacemente il lavoro può essere distribuito tra le diverse unità di calcolo. L'esecuzione parallela introduce un costo aggiuntivo, detto *overhead*, dovuto alla gestione del parallelismo. Nel caso di un'esecuzione distribuita su più GPU, questo costo è legato soprattutto alle operazioni di comunicazione e sincronizzazione tra i dispositivi.

Nel deep learning le GPU (*Graphics Processing Unit*) sono ampiamente utilizzate per svolgere le operazioni computazionalmente più intensive. A differenza delle CPU, progettate per eseguire un numero relativamente ridotto di operazioni complesse, le GPU dispongono di un elevato numero di unità di calcolo in grado di svolgere contemporaneamente operazioni simili su grandi quantità di dati. Questa caratteristica le rende particolarmente adatte alle operazioni matriciali e convoluzionali tipiche dell'addestramento delle reti neurali.

Quando sono disponibili più GPU, una delle strategie utilizzate per distribuire l'addestramento è il *data parallelism*. Il modello viene replicato su ciascuna unità di calcolo, mentre

il dataset viene suddiviso e distribuito in parallelo sotto forma di *mini-batch* distinti. Ogni GPU esegue quindi il calcolo della predizione e della *backpropagation* sulla propria porzione di dati. La *backpropagation* è il processo con cui vengono calcolati i gradienti della funzione di loss rispetto ai parametri della rete, utilizzati poi per aggiornare i pesi del modello. Prima dell'aggiornamento dei parametri, i gradienti calcolati dalle diverse repliche devono essere sincronizzati, in modo che tutte le GPU continuino a utilizzare gli stessi pesi del modello.

TUNA adotta questa modalità attraverso una strategia di Distributed Data Parallel (DDP), nella quale ogni processo gestisce una GPU e una replica completa della rete. La sincronizzazione dei gradienti richiede operazioni di comunicazione tra le GPU.

Per analizzare il comportamento dell'applicazione all'aumentare delle risorse computazionali si considerano due principali modalità di valutazione della scalabilità: strong scaling e weak scaling.

Nell'analisi di strong scaling la dimensione complessiva del problema viene mantenuta costante, mentre aumenta il numero di unità di calcolo. Idealmente, aumentando il numero di GPU, il tempo necessario per completare lo stesso lavoro dovrebbe diminuire in modo proporzionale.

Per valutare l'efficacia della parallelizzazione nello strong scaling possono essere considerate diverse misure di prestazione. Lo speedup indica quanto l'esecuzione parallela risulta più veloce rispetto a quella effettuata utilizzando una singola unità di calcolo. Indicando con T_1 il tempo di esecuzione con una GPU e con T_p il tempo ottenuto utilizzando p GPU, lo speedup è definito come:

$$S_p = \frac{T_1}{T_p}. \tag{2.2}$$

Nel caso ideale, utilizzando p unità di calcolo il tempo di esecuzione dovrebbe ridursi di un fattore p, ottenendo quindi uno speedup pari a p. Nelle applicazioni reali questo comportamento è difficile da raggiungere a causa dell'overhead introdotto dalla parallelizzazione.

L'efficienza parallela permette di quantificare quanto lo speedup ottenuto si avvicini al comportamento ideale ed è definita come:

$$E_p = \frac{S_p}{p}. \tag{2.3}$$

Un'efficienza pari a 1 corrisponde al caso ideale, mentre valori inferiori indicano una perdita di efficienza nell'utilizzo delle risorse aggiuntive.

Nel weak scaling, invece, il carico di lavoro assegnato a ciascuna unità di calcolo viene mantenuto costante, mentre la dimensione complessiva del problema cresce insieme al numero di risorse. In condizioni ideali, il tempo di esecuzione dovrebbe quindi rimanere costante, corrispondendo a un'efficienza di weak scaling pari a 1 [13]. Tale efficienza può essere espressa come:

$$E_p^{\text{weak}} = \frac{T_1}{T_p}, \tag{2.4}$$

Nel Capitolo 5 questi criteri vengono utilizzati per valutare la parallelizzazione dell'addestramento di TUNA al variare del numero di GPU.

2.5 Mixed precision

L'addestramento delle reti neurali viene normalmente eseguito utilizzando numeri in virgola mobile a 32 bit (FP32). Questa rappresentazione offre un buon compromesso tra precisione numerica e intervallo dei valori rappresentabili, ma richiede tempi di calcolo e un impiego di memoria maggiori.

La mixed precision è una tecnica di ottimizzazione che prevede invece l'impiego combinato di rappresentazioni numeriche a diversa precisione, eseguendo alcune operazioni in FP32 e altre mediante formati a 16 bit, principalmente FP16 (*Floating Point* a 16 bit) e BF16 (*Brain Floating Point* a 16 bit) [14, 15]. Entrambi occupano metà della memoria rispetto a FP32, ma distribuiscono diversamente i bit disponibili:

- **FP16:** utilizza 1 bit per il segno, 5 bit per l'esponente e 10 bit per la mantissa. La mantissa relativamente ampia consente di mantenere una buona precisione nella rappresentazione di valori vicini tra loro, mentre il numero ridotto di bit dedicati all'esponente limita l'intervallo dei valori rappresentabili, rendendo questo formato maggiormente soggetto a fenomeni di overflow e underflow.

- **BF16:** utilizza 1 bit per il segno, 8 bit per l'esponente e 7 bit per la mantissa. Avendo lo stesso numero di bit dedicati all'esponente di FP32, mantiene un intervallo di valori rappresentabili molto simile ed è quindi meno soggetto a problemi di overflow e underflow. La mantissa più corta comporta però una minore precisione nella rappresentazione dei singoli valori rispetto a FP16.

Durante l'addestramento, convoluzioni e moltiplicazioni tra matrici, generalmente meno sensibili alla riduzione di precisione, sono tra le operazioni che si prestano meglio a essere eseguite in FP16 o BF16, mentre le operazioni numericamente più delicate vengono mantenute in FP32. Questa scelta può essere gestita automaticamente attraverso meccanismi di *automatic mixed precision* (AMP) forniti da PyTorch, che selezionano la precisione più appropriata per le diverse operazioni [14]. L'impiego di formati a precisione ridotta permette inoltre di diminuire il volume dei dati trasferiti e di sfruttare unità hardware specializzate, come i Tensor Cores presenti nelle GPU NVIDIA, particolarmente efficienti nell'esecuzione di operazioni matriciali a precisione ridotta [15].

Nel caso di FP16 può inoltre essere necessario utilizzare tecniche di *gradient scaling* per ridurre il rischio di underflow durante il calcolo dei gradienti. Il *gradient scaling* consiste

nel moltiplicare temporaneamente la funzione di loss per un fattore di scala prima della fase di *backpropagation*, in modo da rendere i gradienti numericamente più grandi e quindi rappresentabili in FP16. Prima dell'aggiornamento dei parametri, i gradienti vengono riportati alla loro scala originale [14]. BF16, grazie al maggiore intervallo numerico, è generalmente meno soggetto a questo problema.

Esistono anche rappresentazioni a precisione inferiore, come formati floating point a 8 bit o rappresentazioni basate su interi a 8 o 4 bit. La loro adozione comporta tuttavia una maggiore complessità nella gestione della scala dei valori e un rischio più elevato di perdita di informazione e instabilità numerica; inoltre, alcune di queste modalità richiedono hardware e procedure di addestramento maggiormente specializzate.

L'analisi è stata quindi limitata ai formati FP16 e BF16, che rappresentano le principali alternative a 16 bit supportate dalle GPU di Leonardo. Il confronto con FP32 permette di valutare i benefici della mixed precision in termini di prestazioni e costo computazionale, verificando allo stesso tempo che la riduzione della precisione non comprometta la stabilità numerica e la qualità delle predizioni.

Capitolo 3

Dataset e ambiente computazionale

3.1 Ambiente computazionale e strumenti utilizzati

Le attività di preparazione dei dati, addestramento, inferenza (ossia l'utilizzo del modello addestrato per produrre predizioni su nuovi dati) e valutazione del modello sono state svolte utilizzando le risorse di High Performance Computing della partizione Booster del supercomputer Leonardo di CINECA. Il sistema comprende 3456 nodi di calcolo, ciascuno dotato di un processore Intel Xeon Platinum 8358 a 32 core e di quattro GPU NVIDIA A100 basate su architettura Ampere. Ogni nodo dispone inoltre di 512 GB di memoria DDR4, mentre ciascuna GPU è equipaggiata con 64 GB di memoria HBM2e [16].

L'utilizzo delle GPU è particolarmente importante per l'addestramento di reti neurali profonde, poiché consente di eseguire in parallelo un elevato numero di operazioni e di ridurre i tempi di elaborazione. Il codice è stato sviluppato principalmente in Python utilizzando il framework PyTorch, che ha consentito di sfruttare queste risorse computazionali.

Sono state inoltre impiegate librerie come NumPy per l'elaborazione numerica, Astropy per la lettura e la scrittura dei file FITS e Matplotlib per la produzione di grafici e immagini comparative.

A questi strumenti si aggiungono diversi script, in parte forniti e successivamente modificati e in parte sviluppati da zero, per la preparazione dei dati, l'esecuzione delle procedure di inferenza e la generazione e la visualizzazione di grafici, tabelle e immagini utili al confronto dei risultati.

L'esecuzione dei job sul cluster è stata gestita tramite il workload manager SLURM, un software che gestisce l'allocazione delle risorse computazionali e la coda di esecuzione dei job. Durante le fasi di sviluppo e test sono stati utilizzati comandi interattivi mediante `srun`, mentre per gli esperimenti di maggiore durata è stato impiegato `sbatch`. Quest'ultimo consente di sottomettere script batch alla coda di esecuzione del sistema e di avviare automaticamente gli addestramenti e le procedure di inferenza, senza richiedere una connessione continua da parte dell'utente.

Per la visualizzazione delle immagini FITS e delle predizioni è stato infine utilizzato SAOImage DS9, che ha permesso di analizzare le mappe prodotte dal modello e di confrontarle con le immagini radioastronomiche originali [17].

3.2 Dataset: Release 1.4 e Release 1.5

Dataset originale

Il dataset utilizzato per l'addestramento di TUNA è stato costruito a partire da simulazioni cosmologiche numeriche, che forniscono la distribuzione teorica dell'emissione radio priva di distorsioni, chiamata *sky model*. Per simulare un'osservazione reale, lo *sky model* è stato elaborato mediante WSClean, un software per la ricostruzione di immagini radiointerferometriche [18]. L'immagine risultante riproduce gli effetti di un'osservazione effettuata con il radiotelescopio interferometrico LOw-Frequency ARray (LOFAR) e include rumore termico e artefatti dovuti al campionamento incompleto del piano uv e al processo di ricostruzione [1].

L'immagine così ottenuta costituisce il dato di ingresso della rete. La *ground truth* utilizzata durante l'addestramento è invece una maschera binaria generata a partire dallo *sky model*, applicando una soglia alla brillanza superficiale dell'emissione radio. In particolare, i pixel con brillanza uguale o superiore a 10^{-8} Jy/pixel vengono assegnati alla classe emissione e assumono valore 1, mentre quelli al di sotto della soglia vengono assegnati alla classe background e assumono valore 0. Un esempio di *sky model*, della maschera di *ground truth* ricavata da esso e della corrispondente immagine di input utilizzata da TUNA è riportato in Figura 3.1.

Il dataset risulta fortemente sbilanciato, poiché circa il 3% dei pixel appartiene alla classe emissione, mentre il restante 97% appartiene alla classe background, corrispondente all'assenza di emissione [1].

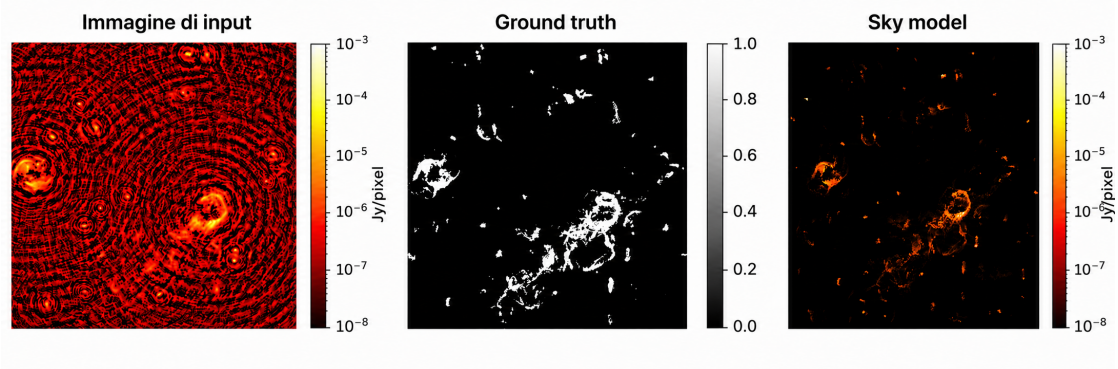


Figura 3.1: Esempio delle tre rappresentazioni associate alla stessa simulazione: *sky model*, maschera binaria di *ground truth* e immagine di input utilizzata da TUNA, ottenuta simulando gli effetti di un'osservazione interferometrica con LOFAR.

Modifiche introdotte

Rispetto a quello utilizzato nello studio di Sanvitale et al. [1], il dataset di riferimento è stato modificato con l'obiettivo di individuare non soltanto le emissioni diffuse, ma anche le radiogalassie. Sono stati individuati due dataset, la Release 1.5 e la Release 1.4.

La **Release 1.5** comprende, oltre alle emissioni diffuse e alle radiogalassie, anche sorgenti puntiformi molto luminose. Queste introducono nelle immagini un forte contributo di rumore e artefatti, spesso caratterizzati da strutture radiali attorno alle sorgenti più intense. La presenza di tali componenti rende il dataset particolarmente complesso e permette di valutare il comportamento della rete in condizioni difficili, nelle quali il segnale di interesse può essere facilmente confuso.

La **Release 1.4** mantiene le emissioni diffuse e le radiogalassie, ma non include le sorgenti puntiformi. Le immagini risultano quindi meno contaminate dagli artefatti associati alle componenti compatte, consentendo di valutare il modello in uno scenario più controllato.

In una prima fase sperimentale, svolta nell'ambito del tirocinio, è stata utilizzata la Release 1.5. Anche su questo dataset sono stati eseguiti addestramenti utilizzando diversi valori di learning rate, confrontando le relative loss e le metriche ottenute in fase di inferenza. È stato inoltre analizzato l'effetto della soglia di confidenza, confrontando differenti valori per valutare il compromesso tra falsi positivi e falsi negativi.

L'analisi ha tuttavia evidenziato che la presenza delle sorgenti puntiformi luminose introduce un livello elevato di rumore e numerosi artefatti, rendendo più difficile valutare in modo affidabile il comportamento della rete sulle emissioni diffuse ed estese. Per questo motivo si è scelto di proseguire la sperimentazione utilizzando la Release 1.4, che è quindi il dataset su cui si concentrano le analisi presentate nella tesi. I risultati ottenuti sulla Release 1.5 sono stati invece considerati solo come una fase preliminare del lavoro.

3.3 Preparazione dei dati

I dati utilizzati per l'addestramento sono costituiti da 300 immagini radioastronomiche in formato FITS, uno standard impiegato in astronomia per la memorizzazione di immagini e delle relative informazioni scientifiche, senza perdita di qualità.

Poiché le immagini originali presentano dimensioni elevate di 2000×2000 pixel, sono state suddivise, insieme alle rispettive maschere di *ground truth*, in tile di 512×512 pixel, sia per ottimizzare il consumo di memoria GPU sia per rendere l'input corrispondente alla dimensione richiesta dalla rete. Poiché le dimensioni delle immagini non sono divisibili per 512, è stata applicata la tecnica dello zero padding, ossia sono stati aggiunti pixel neri ai margini dell'immagine per consentire una suddivisione regolare in tile [1].

Le tile ottenute sono state successivamente salvate in formato `.npz`, un formato compresso basato su NumPy. Ogni file contiene l'immagine di input e la corrispondente maschera di

segmentazione, memorizzate rispettivamente mediante le chiavi `image` e `label`.

Per analizzare il contenuto di questi file è stato sviluppato uno script dedicato alla lettura e alla visualizzazione delle tile in formato `.npz`. Lo script ha permesso di osservare affiancate l'immagine di input e la relativa *ground truth*, facilitando la verifica visiva dei dati generati. Un esempio di visualizzazione di una tile e della corrispondente maschera è riportato in Figura 3.2.

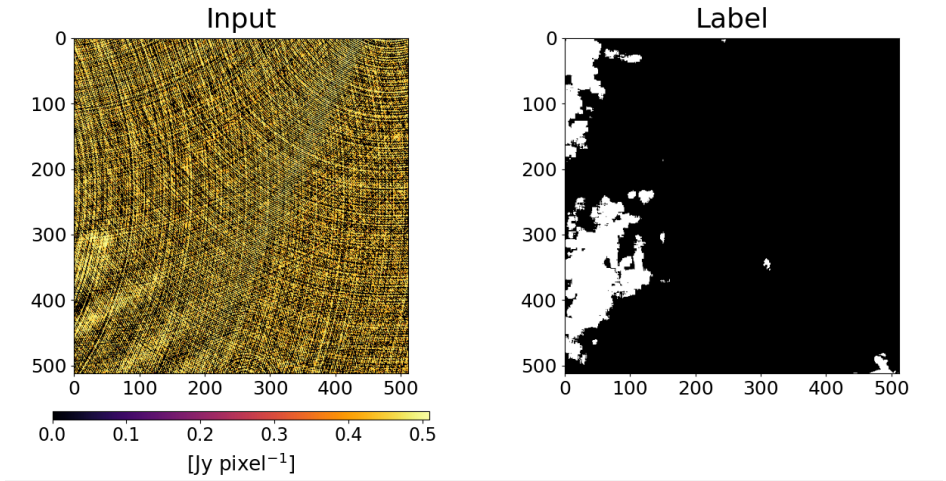


Figura 3.2: Esempio di tile in formato `.npz`: a sinistra l'immagine radioastronomica fornita in input al modello; a destra la corrispondente maschera binaria, con le sorgenti evidenziate in bianco e il background in nero.

Le 300 immagini complessive sono state suddivise in 250 immagini destinate alle fasi di training e validazione e 50 immagini mantenute separate come test set.

Le 250 immagini sono state divise in tile non sovrapposte, quindi ogni immagine ha generato 16 tile, per un totale di 4000 tile per l'intero dataset. Quest'ultimo è stato suddiviso in cinque split disgiunti, ciascuno contenente il 20% dei dati, pari a 800 tile. La suddivisione è stata effettuata mantenendo in ogni split una proporzione simile di pixel appartenenti alle classi emissione e background, così da evitare la formazione di sottoinsiemi eccessivamente sbilanciati o costituiti quasi esclusivamente da immagini prive di emissione. In ciascuna delle cinque esecuzioni effettuate per la cross-validation (vedi Sezione 4.4) a rotazione quattro split sono stati utilizzati per il training, mentre lo split rimanente è stato impiegato per la validazione.

A differenza dei dataset di training e validation, le immagini di test sono state suddivise in tile con una sovrapposizione del 75%, ottenendo 169 tile per singola immagine. Ciascuna tile è stata segmentata autonomamente dal modello. Infine, da ogni predizione è stata estratta e assemblata la sola regione centrale da 128×128 pixel per ricomporre la mappa di predizione finale; questo approccio consente di scartare le aree periferiche delle singole tile, mitigando gli artefatti dovuti agli effetti di bordo, riducendo tuttavia l'area di predizione finale alla sola regione centrale dell'immagine (1664×1664 pixel).

Le metriche di valutazione sono state successivamente calcolate su queste immagini ricomposte, previa binarizzazione della maschera tramite l'applicazione di una soglia di confidenza.

3.4 Configurazione del modello

Prima di avviare gli addestramenti è stato necessario configurare il codice in base alla struttura delle cartelle utilizzate su Leonardo. In particolare, nel file `TU_parameters.py` sono stati impostati i percorsi relativi alle tile, alle liste contenenti i dati di training e validation, alle cartelle di output e alla directory destinata al salvataggio dei checkpoint, utilizzati per memorizzare lo stato della rete durante l'addestramento.

Questa configurazione ha permesso al modello di individuare correttamente i dati di input e i file necessari durante le diverse fasi del training.

Le tile preparate secondo le modalità descritte nella Sezione 3.3 sono state elaborate con un batch size pari a 24. Quest'ultimo indica il numero di tile elaborate dal modello prima dell'aggiornamento dei pesi della rete, mentre il learning rate determina l'ampiezza degli aggiornamenti applicati ai pesi durante l'addestramento. Per gli esperimenti descritti nel capitolo successivo sono stati considerati quattro valori del learning rate: 0,01, 0,005, 0,001 e 0,0005. Il batch size e gli altri principali parametri di configurazione sono stati invece mantenuti invariati, così da rendere confrontabili i diversi addestramenti.

3.5 Metriche di valutazione

Poiché il dataset è fortemente sbilanciato, basarsi soltanto sull'Accuracy non sarebbe sufficiente per determinare in maniera realistica le prestazioni della rete. Infatti, la prevalenza dei pixel di sfondo potrebbe portare a valori elevati di Accuracy anche in presenza di errori nella segmentazione delle sorgenti.

Per ciascun pixel, il modello assegna un punteggio di appartenenza alla classe delle sorgenti da segmentare. Applicando una soglia di confidenza, tali valori vengono convertiti in una maschera binaria, successivamente confrontata con la *ground truth*.

Il confronto tra predizione e *ground truth* permette di distinguere quattro possibili casi per ciascuna classe. Considerando la classe emissione, che rappresenta la classe di interesse, si distinguono:

- **veri positivi (TP):** pixel correttamente riconosciuti come appartenenti alla sorgente;
- **falsi positivi (FP):** pixel classificati come sorgente, ma appartenenti al background;
- **falsi negativi (FN):** pixel appartenenti alla sorgente che il modello non ha individuato;
- **veri negativi (TN):** pixel del background correttamente classificati.

Le prestazioni del modello sono state valutate mediante cinque metriche: Accuracy, Dice, Intersection over Union, Precision e Recall.

L'**Accuracy** misura la percentuale complessiva di pixel classificati correttamente:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Il **coefficiente Dice** misura il rapporto tra il doppio dell'intersezione e la somma totale dei pixel delle due maschere (quella predetta e la *ground truth*):

$$\text{Dice} = \frac{2TP}{2TP + FP + FN}$$

Un valore prossimo a 1 indica una forte sovrapposizione tra le due maschere, mentre un valore prossimo a 0 indica una segmentazione poco accurata.

L'**Intersection over Union**, o IoU, misura il rapporto tra l'area dell'intersezione e l'area dell'unione delle due maschere (quella predetta e la *ground truth*):

$$\text{IoU} = \frac{TP}{TP + FP + FN}$$

La **Precision** misura la percentuale di pixel classificati come positivi che appartengono effettivamente alla sorgente.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Una Precision elevata indica quindi una presenza ridotta di falsi positivi.

Il **Recall** misura invece la percentuale di pixel appartenenti alla sorgente che sono stati correttamente individuati:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Un Recall elevato indica che il modello riesce a rilevare gran parte dell'emissione presente nella *ground truth*, producendo pochi falsi negativi.

Capitolo 4

Selezione e validazione del modello

4.1 Addestramento su Release 1.4

Dopo la fase sperimentale preliminare condotta sulla Release 1.5, gli addestramenti sono stati ripetuti utilizzando il dataset Release 1.4, già descritto nella Sezione 3.2. Il passaggio a questa versione ha avuto lo scopo di valutare il comportamento della rete in assenza delle sorgenti puntiformi che introducono nel dataset più artefatti, mantenendo comunque la presenza delle radiogalassie, ossia sorgenti estese.

Sono stati quindi eseguiti quattro addestramenti distinti, utilizzando gli stessi valori del learning rate definiti nella Sezione 3.4, mantenendo invariati gli altri principali parametri di configurazione. Per ciascun addestramento sono stati salvati i checkpoint ogni 50 epoche, successivamente utilizzati per le procedure di inferenza e per il calcolo delle metriche.

In tutte le analisi svolte sulla Release 1.4 è stata adottata una soglia di confidenza pari a 0,5.

4.2 Confronto tra i learning rate

Gli addestramenti sono stati configurati per un massimo di 400 epoche. Il limite di 24 ore previsto per i job su Leonardo non ha tuttavia consentito di completare l'intero training: le esecuzioni si sono interrotte intorno all'epoca 238. Per il confronto sono stati quindi utilizzati i checkpoint comuni alle quattro configurazioni, corrispondenti alle epoche 49, 99, 149 e 199.

L'analisi delle stesse epoche ha garantito condizioni omogenee, evitando che differenze nella durata del training o nell'evoluzione del learning rate influenzassero i risultati.

4.2.1 Analisi delle loss

Anche per gli addestramenti eseguiti sulla Release 1.4 sono stati generati i grafici delle loss sul training e sul validation set. Poiché le curve di training mostrano andamenti

molto simili tra le diverse configurazioni, in Figura 4.1 viene riportato un solo esempio rappresentativo.

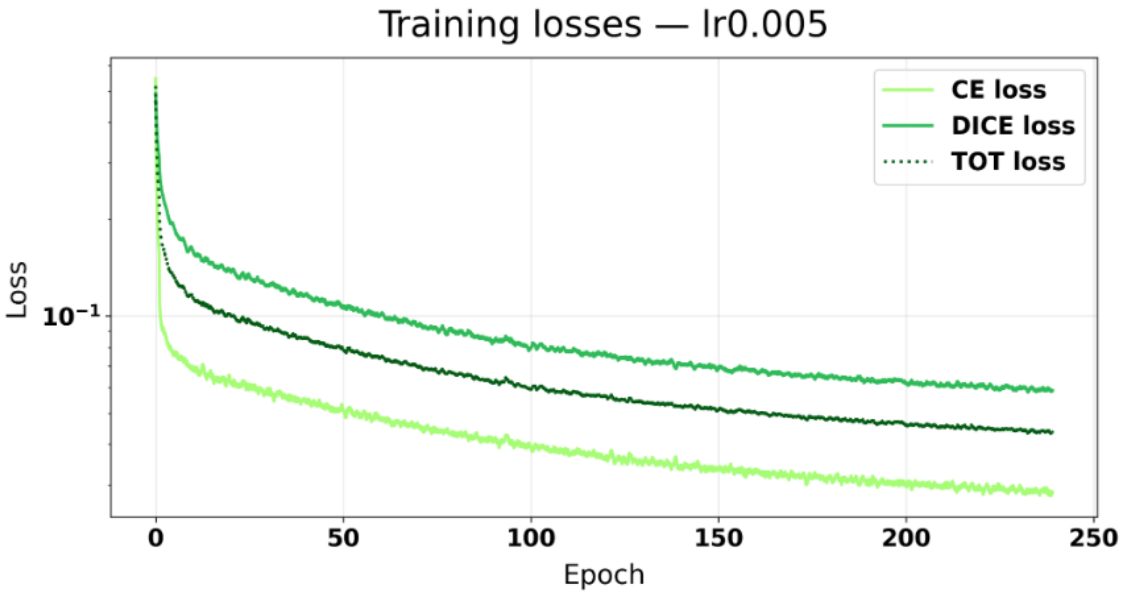


Figura 4.1: Andamento della Cross-Entropy Loss, della Dice Loss e della loss totale sul training set della Release 1.4, con learning rate pari a 0,005, in funzione delle epoche.

Analogamente, in Figura 4.2 è riportato un esempio rappresentativo dell'andamento delle loss sul validation set per la stessa configurazione. Oltre alla loss totale, definita nell'Equazione 2.1, il grafico mostra separatamente i contributi della Cross-Entropy Loss e della Dice Loss associati alle classi background ed emissione.

Per analizzare separatamente il contributo delle due classi, la Cross-Entropy Loss è stata scomposta tenendo conto della frazione di pixel appartenenti rispettivamente al background e all'emissione. Indicando con f_{bg} e f_{em} tali frazioni, i contributi ottenuti sono:

$$L^{w}_{\text{CE,bg}} = 0{,}5 f_{\text{bg}} L_{\text{CE,bg}}, \tag{4.1}$$

$$L^{w}_{\text{CE,em}} = 0{,}5 f_{\text{em}} L_{\text{CE,em}}. \tag{4.2}$$

La componente Dice, che nella loss complessiva ha peso pari a 0,5, viene invece suddivisa equamente tra le due classi, ottenendo:

$$L^{w}_{\text{Dice,bg}} = 0{,}25 L_{\text{Dice,bg}}, \tag{4.3}$$

$$L^{w}_{\text{Dice,em}} = 0{,}25 L_{\text{Dice,em}}. \tag{4.4}$$

La validation loss complessiva corrisponde quindi alla somma dei quattro contributi pesati:

$$L_{\text{val}} = L^{w}_{\text{CE,bg}} + L^{w}_{\text{CE,em}} + L^{w}_{\text{Dice,bg}} + L^{w}_{\text{Dice,em}}. \tag{4.5}$$

Questa scomposizione consente di osservare separatamente quanto gli errori relativi al background e all'emissione contribuiscano alla loss complessiva durante l'addestramento.

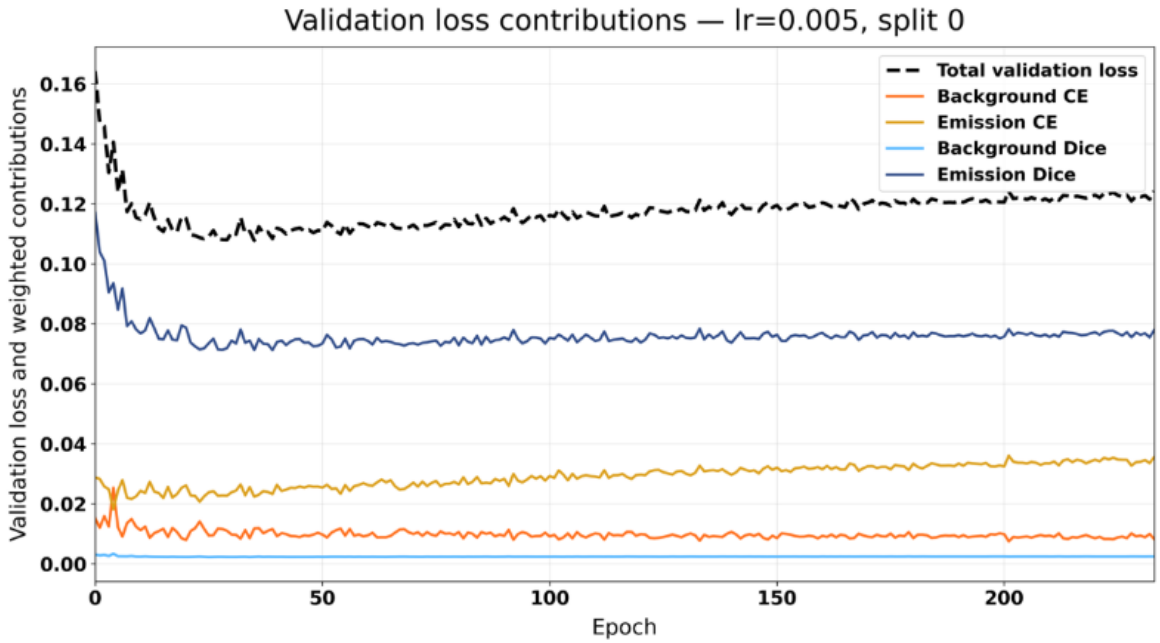


Figura 4.2: Andamento della loss totale di validazione e dei contributi pesati della Cross-Entropy e della Dice Loss per le classi background ed emissione sulla Release 1.4, con learning rate pari a 0,005.

La Figura 4.2 mostra che la loss totale è diminuita rapidamente nelle prime epoche, raggiungendo un minimo intorno alle epoche 20–30, per poi aumentare lievemente e stabilizzarsi. L'andamento è stato influenzato principalmente dalla Dice Loss associata alla classe emissione, mentre i contributi relativi al background sono rimasti molto bassi e pressoché costanti. Anche la Cross-Entropy della classe emissione ha presentato una crescita graduale dopo la fase iniziale.

Questo andamento indica che il modello ha appreso rapidamente nelle prime fasi, ma che successivamente il miglioramento sul validation set si è interrotto. La crescita della loss dopo il minimo raggiunto è compatibile con l'inizio di una fase di *overfitting*. Le loss sono state quindi affiancate al confronto delle metriche e delle maschere prodotte, necessario per valutare complessivamente le prestazioni del modello.

4.2.2 Metriche sul validation set

Per confrontare le quattro configurazioni sono state calcolate le metriche sul validation set, considerando la classe 1 (emissione) come classe positiva. Data la marcata sproporzione delle classi (la classe 1 copre circa il 3% del totale), questa scelta evita che le metriche vengano

sovrastimate dalla dominanza del background. Tali metriche sono state valutate globalmente a livello di pixel, accumulando i risultati su tutte le tile del dataset di validazione. I risultati sono riportati nella Tabella 4.1 e rappresentati graficamente in Figura 4.3.

EMISSION metrics Release 1.4 - Conf 0.5

Validation set

lr	epoch	Dice	IoU	Accuracy	Precision	Recall
0.01	49	**0.6769**	**0.5116**	0.9778	0.7515	0.6158
0.01	99	0.6583	0.4906	**0.978**	0.7924	0.563
0.01	149	0.6591	0.4916	0.9776	0.7734	0.5743
0.01	199	0.659	0.4914	0.9774	0.7626	0.5802
0.005	49	0.6742	0.5085	0.9777	0.7503	0.6121
0.005	99	0.6695	0.5031	0.9774	0.7465	0.6068
0.005	149	0.662	0.4948	0.9774	0.7585	0.5873
0.005	199	0.6661	0.4994	0.9772	0.7445	0.6027
0.001	49	0.6455	0.4765	**0.978**	**0.8193**	0.5325
0.001	99	0.6666	0.4999	0.9777	0.7638	0.5913
0.001	149	0.6661	0.4993	0.9777	0.7655	0.5895
0.001	199	0.6632	0.4961	0.9776	0.764	0.586
0.0005	49	0.6591	0.4915	0.9775	0.768	0.5772
0.0005	99	0.6726	0.5067	0.9768	0.7194	**0.6315**
0.0005	149	0.6636	0.4965	0.9774	0.7565	0.591
0.0005	199	0.6589	0.4913	0.9772	0.7543	0.5849

Tabella 4.1: Metriche calcolate sul validation set della Release 1.4 per i diversi learning rate e checkpoint, con soglia di confidenza pari a 0,5.

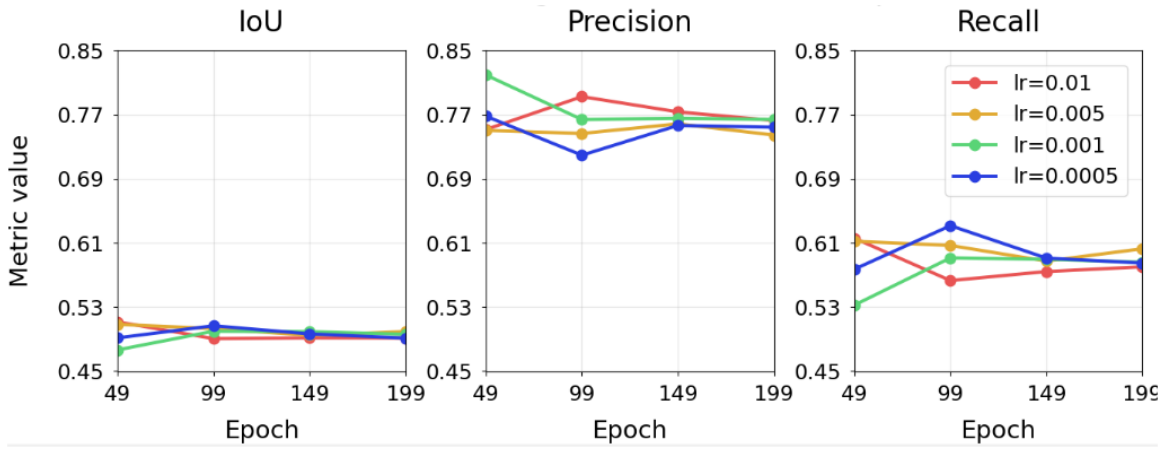


Figura 4.3: Andamento di IoU, Precision e Recall sul validation set della Release 1.4 per i diversi learning rate, ai checkpoint delle epoche 49, 99, 149 e 199, con soglia di confidenza pari a 0,5.

I valori di Accuracy sono risultati molto simili per tutte le configurazioni, principalmente a causa della forte prevalenza dei pixel appartenenti al background. L'analisi si è quindi concentrata soprattutto su IoU, Precision e Recall.

La configurazione con learning rate pari a 0,01 e checkpoint all'epoca 49 ha ottenuto i valori più elevati di Dice e IoU. Il learning rate 0,001 all'epoca 49 ha raggiunto invece

i valori massimi di Precision e Accuracy, mentre il Recall più elevato è stato ottenuto con learning rate 0,0005 all'epoca 99. Nessuna configurazione è risultata quindi migliore per tutte le metriche considerate.

Alla luce degli andamenti osservati, il learning rate 0,005 è risultato il più stabile nel garantire valori elevati di Recall e IoU attraverso le epoche, ed è stato quindi scelto come valore di riferimento.

Nell'interpretazione di questi risultati è tuttavia necessario considerare la procedura utilizzata per il calcolo delle metriche. Sul validation set le metriche sono state valutate globalmente a livello di pixel, accumulando i risultati su tutte le tile del dataset di validazione. Questa scelta è coerente con la modalità di training, nella quale la loss viene valutata su tutti i pixel delle tile appartenenti a un batch.

Come descritto nella Sezione 3.3, per il test set è stata invece adottata una procedura per immagine: ciascuna mappa di predizione (1664×1664 pixel) è stata ricostruita riassemblando le regioni centrali (128×128 pixel) di 169 tile sovrapposte al 75% per mitigare gli artefatti di bordo. Le metriche sono state quindi calcolate sulle singole immagini binarizzate e successivamente mediate sull'intero set. Questa scelta permette di stimare le prestazioni reali del modello su scala di immagine completa, simulando l'output tipico dell'osservazione di un telescopio su un singolo puntamento.

Per questo motivo, il checkpoint finale da utilizzare è stato scelto in base alle metriche così ottenute.

Un possibile miglioramento futuro consisterebbe nel costruire gli split a partire dalle immagini complete e nell'applicare anche in fase di validazione la stessa procedura di inferenza e ricostruzione utilizzata nella fase di test. In questo modo, la selezione del checkpoint potrebbe essere effettuata interamente sul validation set, mantenendo il test set indipendente e riservato esclusivamente alla valutazione finale del modello.

4.3 Valutazione sul test set

Una volta selezionato il learning rate pari a 0,005, sono stati confrontati sul test set i checkpoint alle epoche 49 e 199. Le metriche sono state calcolate per immagine e successivamente mediate sull'intero test set. I risultati sono riportati nella Tabella 4.2.

Checkpoint	**Dice**	**IoU**	**Precision**	**Recall**
Epoca 49	0,7069	0,5504	**0,8281**	0,6199
Epoca 199	**0,7198**	**0,5674**	0,7734	**0,6781**

Tabella 4.2: Metriche calcolate sul test set per il modello con learning rate pari a 0,005 ai checkpoint delle epoche 49 e 199, con soglia di confidenza pari a 0,5.

Il checkpoint all'epoca 49 ha presentato un comportamento più conservativo, con una maggiore Precision ma una minore capacità di individuare tutte le regioni di emissione. Proseguendo l'addestramento fino all'epoca 199, il Recall è aumentato a fronte di una moderata riduzione della Precision, mentre Dice e IoU hanno mostrato un miglioramento complessivo della qualità della segmentazione.

Sulla base di questo compromesso tra Precision e Recall e del miglioramento delle metriche di sovrapposizione, è stato selezionato come configurazione finale il modello con learning rate pari a 0,005 e checkpoint all'epoca 199.

4.4 Cross-validation

L'insieme di 4000 tile destinato alle fasi di training e validazione è stato suddiviso in cinque split da 800 tile ciascuno. A ogni esecuzione è stato utilizzato un diverso split come validation set, mentre i restanti split sono stati impiegati per l'addestramento. Il modello ottenuto per ciascuna suddivisione è stato quindi valutato sul test dataset comune a tutti, così da confrontare le prestazioni al variare della composizione dei dati di training. Le metriche ottenute nei cinque split, insieme ai relativi valori medi e alle deviazioni standard, sono riportate nella Tabella 4.3.

5-fold cross-validation – test set
LR=0.005, epoch=199, confidence=0.5

Split	Dice	IoU	Accuracy	Precision	Recall
0	**0.7198**	**0.5674**	0.9745	0.7734	**0.6781**
1	0.7171	0.5638	0.9748	0.7923	0.6595
2	0.7188	0.5657	0.9748	0.7794	0.6718
3	0.7179	0.5643	0.9746	0.7801	0.6684
4	0.7137	0.5596	**0.9748**	**0.8044**	0.6449
Mean	0.7175	0.5641	0.9747	0.7859	0.6645
Std	0.0023	0.0029	0.0002	0.0124	0.0129

Tabella 4.3: Metriche ottenute sul test set nei cinque esperimenti di cross-validation, con media e deviazione standard, utilizzando learning rate pari a 0,005, checkpoint all'epoca 199 e confidence threshold pari a 0,5.

La 5-fold cross-validation ha mostrato che la configurazione selezionata ha mantenuto un comportamento consistente anche al variare della suddivisione dei dati. Le metriche sono infatti variate molto poco tra gli split, indicando che il modello non dipende da una particolare composizione del training set.

Le oscillazioni maggiori hanno riguardato Precision e Recall e hanno riflesso soprattutto un diverso equilibrio tra falsi positivi e falsi negativi, senza compromettere la qualità comples-

siva della segmentazione. Nel complesso, i risultati hanno confermato una buona capacità di generalizzazione del modello e hanno mostrato la robustezza della scelta finale effettuata.

4.5 Applicazione a immagini osservative

Il modello selezionato è stato infine applicato a tre immagini osservative di ammassi di galassie considerate particolarmente significative per la presenza di emissioni diffuse ed estese: PSZ2G023.17+86.71, PSZ2G057.61+34.93 e PSZ2G149.22+54.18. Le osservazioni provengono dalla survey LoTSS-DR2, realizzata mediante LOFAR, e riguardano ammassi appartenenti al catalogo Planck PSZ2, inclusi nel campione osservativo utilizzato per lo studio di TUNA [1].

A differenza dei dati sintetici, per le immagini reali non è disponibile una *ground truth* binaria. Le predizioni sono state quindi confrontate con immagini ottenute a differenti risoluzioni, nelle quali l'emissione diffusa risulta maggiormente visibile.

Ogni figura è composta da quattro pannelli:

1. **Robust**: rappresenta l'immagine LoTSS ad alta risoluzione utilizzata come input del modello. Questo tipo di immagine conserva maggiormente le strutture compatte, ma rende più difficile distinguere le emissioni diffuse deboli.

2. **Prediction**: mostra la mappa di confidenza prodotta da TUNA, con valori compresi tra 0 e 1. Il contorno verde delimita le regioni con confidenza superiore alla soglia scelta, pari a 0,5.

3. **Taper = 25 kpc**: mostra l'osservazione ottenuta applicando un taper, cioè una tecnica che riduce la risoluzione dell'immagine, corrispondente a una scala fisica di 25 kpc. La risoluzione inferiore aumenta la sensibilità verso strutture più estese. Il contorno azzurro indica le regioni con intensità superiore a tre volte il rumore dell'immagine (rms), mentre quello verde riporta la predizione di TUNA.

4. **Taper = 100 kpc SUB**: rappresenta l'immagine a risoluzione ancora più bassa nella quale le sorgenti compatte sono state sottratte. Questa elaborazione rende più evidente l'emissione diffusa e costituisce il principale riferimento osservativo per valutare qualitativamente la predizione di tali sorgenti. Anche in questo pannello il contorno azzurro corrisponde al livello di $3 \times$ rms, mentre quello verde identifica le regioni selezionate dal modello.

I confronti ottenuti per i tre ammassi selezionati sono riportati nelle Figure 4.4, 4.5 e 4.6.

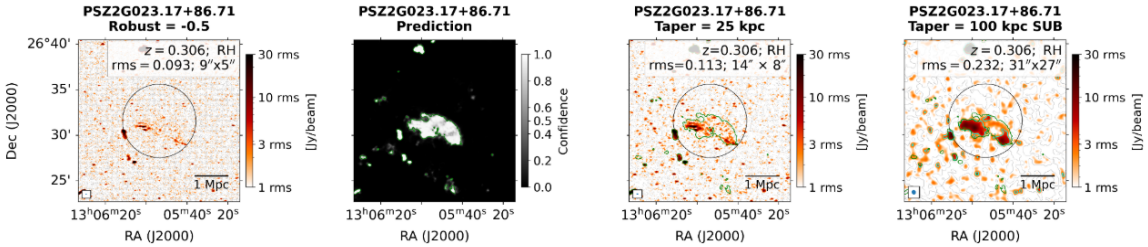


Figura 4.4: Confronto, per l'ammasso PSZ2G023.17+86.71, tra l'immagine LoTSS Robust utilizzata come input, la mappa di confidenza predetta da TUNA, l'immagine sottoposta a tapering a 25 kpc e quella sottoposta a tapering a 100 kpc con le sorgenti compatte sottratte. I contorni verdi delimitano le regioni individuate da TUNA con una soglia di confidenza pari a 0,5, mentre i contorni azzurri nelle immagini sottoposte a tapering indicano le regioni con intensità superiore a $3 \times$ rms.

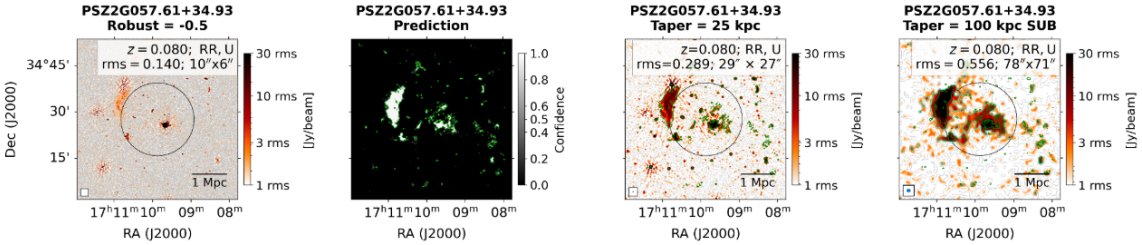


Figura 4.5: Confronto, per l'ammasso PSZ2G057.61+34.93, tra l'immagine LoTSS Robust utilizzata come input, la mappa di confidenza predetta da TUNA, l'immagine sottoposta a tapering a 25 kpc e quella sottoposta a tapering a 100 kpc con le sorgenti compatte sottratte. I contorni verdi delimitano le regioni individuate da TUNA con una soglia di confidenza pari a 0,5, mentre i contorni azzurri nelle immagini sottoposte a tapering indicano le regioni con intensità superiore a $3 \times$ rms.

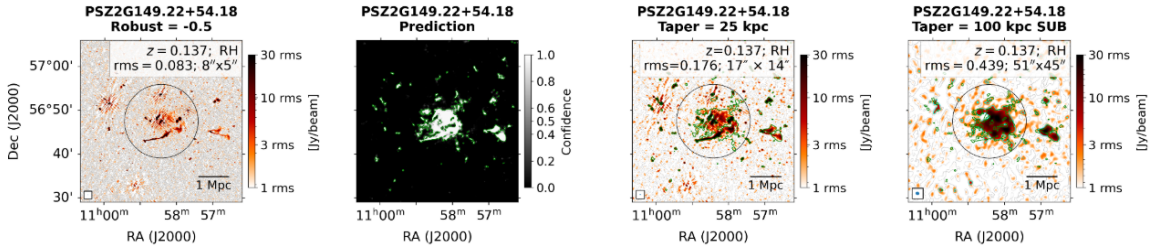


Figura 4.6: Confronto, per l'ammasso PSZ2G149.22+54.18, tra l'immagine LoTSS Robust utilizzata come input, la mappa di confidenza predetta da TUNA, l'immagine sottoposta a tapering a 25 kpc e quella sottoposta a tapering a 100 kpc con le sorgenti compatte sottratte. I contorni verdi delimitano le regioni individuate da TUNA con una soglia di confidenza pari a 0,5, mentre i contorni azzurri nelle immagini sottoposte a tapering indicano le regioni con intensità superiore a $3 \times$ rms.

Ciascuno dei tre ammassi selezionati presenta caratteristiche geometriche differenti, ideali per testare il comportamento del modello su diversi tipi di segnale:

- **PSZ2G023.17+86.71** (Figura 4.4) ospita una sorgente diffusa classificata come alone radio (*radio halo*), cioè un'emissione radio estesa associata alla regione centrale dell'ammasso. L'emissione presenta una morfologia marcatamente allungata in direzione est-ovest, anziché una forma approssimativamente circolare.

- **PSZ2G057.61+34.93** (Figura 4.5) mostra una struttura più complessa. Oltre a una debole emissione centrale, presenta una sorgente allungata nella regione periferica dell'ammasso, identificata come *radio relic*, una forma di emissione diffusa tipicamente osservata nelle regioni esterne degli ammassi. Sono inoltre presenti sorgenti luminose, tra cui un AGN (*Active Galactic Nucleus*) centrale particolarmente intenso, associato all'attività di un buco nero supermassiccio. La presenza di queste componenti rende più difficile distinguere l'emissione diffusa dagli artefatti.

- **PSZ2G149.22+54.18** (Figura 4.6) è caratterizzato da un alone radio esteso e diffuso, con una bassa brillanza superficiale. La debolezza del segnale rende questo caso particolarmente utile per valutare la capacità del modello di distinguere l'emissione reale dal rumore di fondo.

Questo sotto-campione consente di valutare il comportamento del modello in presenza di morfologie asimmetriche, sorgenti luminose e strutture a bassa brillanza superficiale.

Il confronto qualitativo tra le predizioni ottenute dalle immagini ad alta risoluzione e le emissioni visibili nelle immagini sottoposte a tapering ha mostrato, nei tre casi analizzati, una corrispondenza complessiva tra le regioni individuate da TUNA e le strutture diffuse ed estese osservate. L'assenza di una *ground truth* binaria non permette tuttavia di quantificare le prestazioni del modello su queste immagini.

Per superare almeno in parte questo limite, è stata analizzata anche un'ulteriore osservazione della survey LoTSS DR2. A differenza dei tre casi precedenti, riferiti a regioni centrate sui singoli ammassi selezionati, in questo caso è stato considerato un intero puntamento osservativo, il campo P42Hetdex07. Le regioni individuate da TUNA sono state quindi confrontate con le sorgenti già note e presenti nei seguenti cataloghi: il catalogo Clews [19], che raccoglie radiogalassie e ne descrive le principali caratteristiche morfologiche, il catalogo GRG / Mostert [20], dedicato alle radiogalassie caratterizzate da una grande estensione, e il catalogo Planck–LoTSS DR2 [21], relativo alle emissioni radio diffuse associate agli ammassi di galassie.

Nel pannello di destra è riportata nuovamente la maschera di TUNA, questa volta mostrando tutte le regioni individuate automaticamente nel campo. Tra queste compaiono anche strutture che non risultano associate alle sorgenti presenti nei cataloghi considerati. Questo non significa necessariamente che si tratti di nuove sorgenti, perché alcune rilevazioni potrebbero essere dovute ad artefatti o a strutture non comprese nei cataloghi utilizzati. Allo

stesso tempo, potrebbero corrispondere a emissioni reali non ancora catalogate e meritano quindi di essere analizzate più nel dettaglio.

Rispetto alle tre immagini osservative analizzate in precedenza, per questo caso è stato utilizzato anche uno script esterno per mettere in relazione la predizione di TUNA con le sorgenti presenti nei cataloghi astronomici.

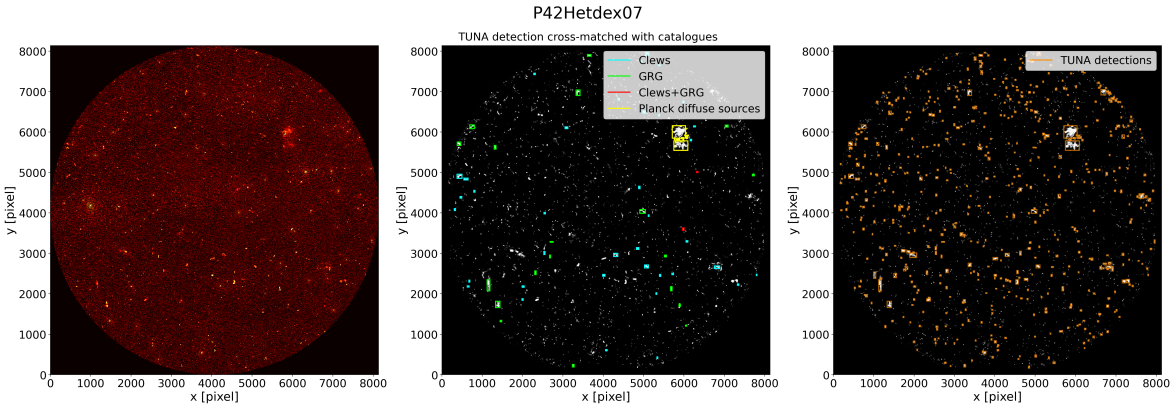


Figura 4.7: Applicazione di TUNA al campo osservativo P42Hetdex07 della survey LoTSS DR2. Sono mostrati l'immagine radio originale, il confronto tra la maschera prodotta dal modello e le sorgenti presenti nei cataloghi astronomici considerati e l'insieme delle regioni rilevate dal modello.

Capitolo 5

Parallelizzazione e analisi delle prestazioni

5.1 Configurazioni per l'addestramento distribuito

Come configurazione di riferimento è stato utilizzato il modello addestrato sulla Release 1.4, con learning rate pari a 0,005 e batch size pari a 24.

L'addestramento è stato esteso a configurazioni multi-GPU con l'obiettivo di sfruttare i vantaggi offerti dal calcolo parallelo, in particolare la riduzione dei tempi di esecuzione. Sono state quindi considerate configurazioni con 1, 2, 4 e 8 GPU. La valutazione della parallelizzazione non può tuttavia limitarsi alle sole prestazioni computazionali: un'eventuale riduzione dei tempi di addestramento è utile solo se i modelli ottenuti mantengono una qualità delle predizioni confrontabile con quelle della configurazione a singola GPU. Per questo motivo, oltre ai tempi di esecuzione, sono state analizzate anche le metriche di segmentazione, così da verificare che l'aumento delle risorse non compromettesse la qualità dei risultati.

La scelta del batch size è strettamente vincolata alle dinamiche di convergenza dell'architettura e alla memoria disponibile sui nodi di calcolo. Nello studio originale di TUNA, la ricerca empirica condotta sui valori 8, 16 e 24 ha individuato nel batch size a 24 la configurazione ottimale per le prestazioni di segmentazione, coincidente inoltre con il valore di riferimento dell'implementazione originale di TransUNet [1]. Un tentativo di incrementare il batch size a 32 ha infatti saturato la memoria della GPU su Leonardo, generando un errore di *Out-Of-Memory* (OOM) e rendendo impossibile il completamento dell'addestramento.

Per gli esperimenti di parallelizzazione è stato utilizzato lo stesso insieme di dati considerato nella configurazione di riferimento, mantenendo lo split 0 come suddivisione tra training e validation. In questo modo, il confronto tra le diverse esecuzioni non è stato influenzato da cambiamenti nella composizione dei dati.

L'addestramento distribuito è stato gestito mediante la libreria Hugging Face Accelerate. Le esecuzioni sono state effettuate sulla partizione Booster di Leonardo, già descritta nella

Sezione 3.1. Poiché ciascun nodo di calcolo dispone di quattro GPU, le configurazioni con 1, 2 e 4 GPU sono state eseguite utilizzando un singolo nodo, mentre per la configurazione con 8 GPU sono stati utilizzati due nodi.

A partire da queste impostazioni comuni sono state quindi condotte le analisi di strong scaling e weak scaling. Negli esperimenti di strong scaling il carico complessivo è stato mantenuto costante, aumentando il numero di GPU e riducendo di conseguenza il batch size assegnato a ciascun dispositivo. Negli esperimenti di weak scaling è stato invece mantenuto costante il batch size per GPU, aumentando quindi quello globale all'aumentare del numero di dispositivi. Le definizioni generali di strong e weak scaling e le relative metriche di prestazione sono state introdotte nella Sezione 2.4. Le sezioni successive analizzano l'effetto della parallelizzazione sull'andamento delle loss, sulle metriche del modello e sulle prestazioni computazionali.

5.2 Analisi delle loss

Per analizzare più nel dettaglio il comportamento degli addestramenti eseguiti con un numero crescente di GPU, sono state confrontate le curve della loss di training e di validation per i quattro casi considerati. I risultati relativi allo strong scaling sono riportati in Figura 5.1, mentre quelli relativi al weak scaling sono mostrati in Figura 5.2.

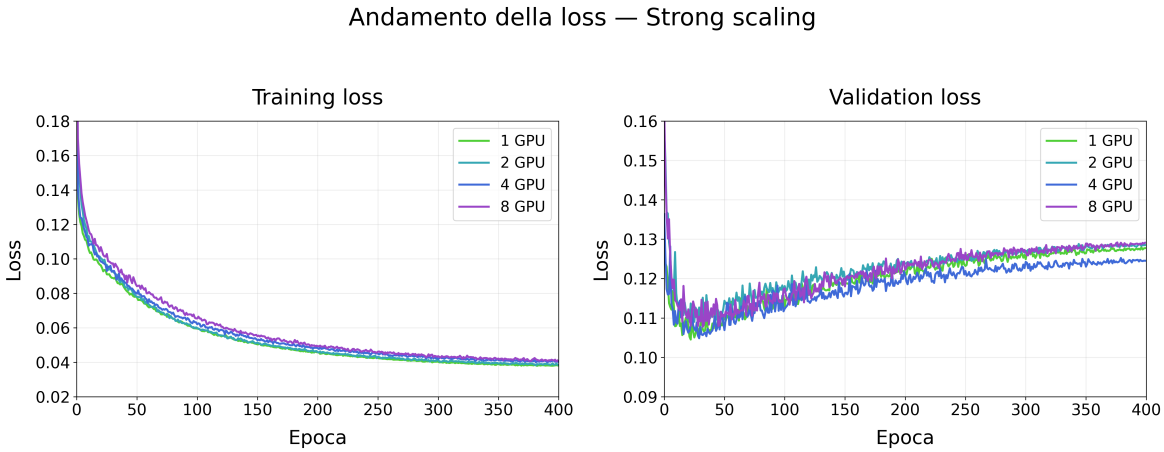


Figura 5.1: Andamento della loss di training e di validation nelle configurazioni di strong scaling.

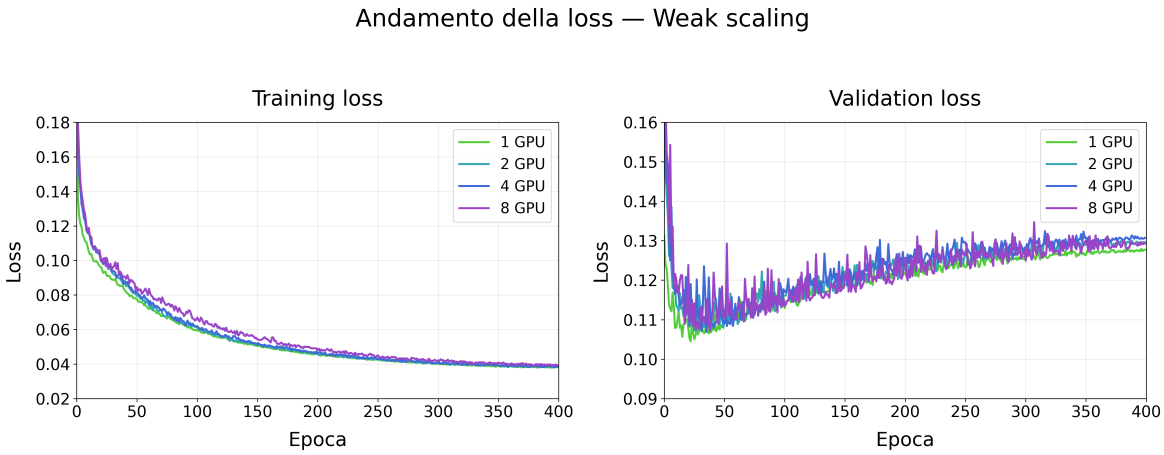


Figura 5.2: Andamento della loss di training e di validation nelle configurazioni di weak scaling.

Per quanto riguarda il training, le curve hanno un andamento molto simile in tutte le configurazioni. La loss cala rapidamente nelle prime epoche e poi continua a diminuire più lentamente, fino a stabilizzarsi nella parte finale dell'addestramento. All'aumentare del numero di GPU si osserva una lieve riduzione della velocità con cui la loss diminuisce, ma si tratta di una differenza molto contenuta, dato che le curve rimangono quasi sovrapposte per gran parte dell'addestramento. Nel complesso, quindi, non si osservano variazioni rilevanti né nello strong scaling né nel weak scaling.

La validation loss mostra invece un andamento diverso. Dopo una diminuzione iniziale, raggiunge un minimo nelle prime fasi dell'addestramento e successivamente tende ad aumentare in modo graduale. Questo comportamento è comune a tutte le configurazioni e risulta più chiaro osservando separatamente le diverse componenti della loss. A titolo di esempio, la Figura 5.3 mostra l'andamento delle singole componenti per il caso con 4 GPU nell'esperimento di weak scaling. Il comportamento osservato è analogo anche negli altri casi. Il calo iniziale è dovuto soprattutto alla Dice loss relativa all'emissione, che diminuisce rapidamente nelle prime epoche. In seguito questa componente si mantiene su valori abbastanza stabili, ma che tendono leggermente all'aumento, mentre la cross-entropy associata all'emissione cresce progressivamente. Al contrario, la cross-entropy del background tende a diminuire, mentre la Dice loss del background rimane molto bassa e quasi costante. L'aumento della validation loss nelle fasi successive dell'addestramento sembra quindi essere legato principalmente alle componenti associate all'emissione.

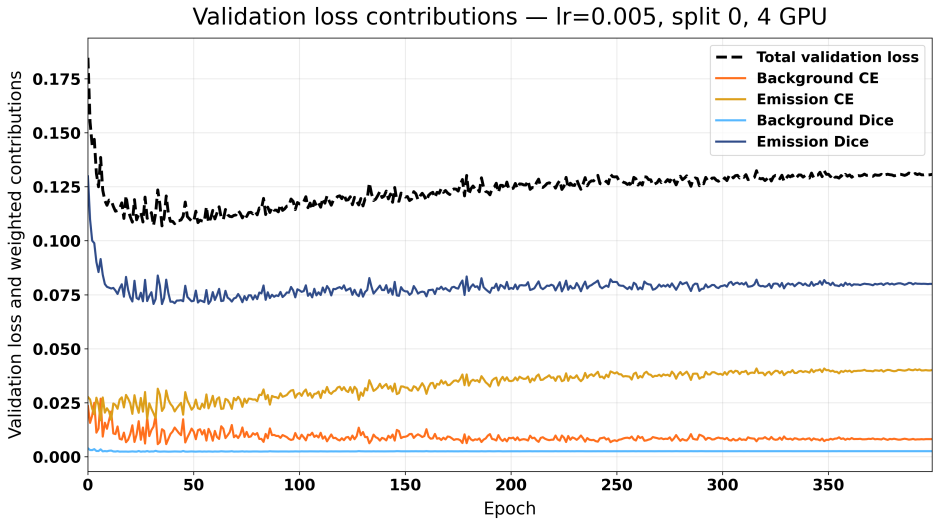


Figura 5.3: Contributi delle diverse componenti alla validation loss per la configurazione con 4 GPU nell'esperimento di weak scaling.

Nel complesso, le curve restano molto simili anche cambiando il numero di GPU. Le differenze tra le varie configurazioni sono limitate e riguardano soprattutto i valori della loss raggiunti durante l'addestramento, mentre il modo in cui le curve evolvono nel tempo rimane praticamente lo stesso.

5.3 Effetto della parallelizzazione sulle metriche

L'effetto della parallelizzazione sui risultati del modello è stato valutato confrontando le metriche ottenute nelle diverse configurazioni di addestramento. Le metriche sono state calcolate utilizzando una soglia di confidenza pari a 0,5 e confrontate con quelle ottenute nella configurazione a singola GPU. Poiché l'Accuracy è poco discriminante, principalmente a causa della forte prevalenza dei pixel appartenenti al background, l'analisi si è concentrata, anche in questo caso, soprattutto su IoU, Precision e Recall. I risultati sono stati analizzati separatamente per gli esperimenti di strong scaling e weak scaling.

5.3.1 Strong scaling

I risultati ottenuti nelle configurazioni di strong scaling sono riportati nella Tabella 5.1 e rappresentati graficamente in Figura 5.4.

Metrics across GPUs — Strong scaling (Release 1.4, conf. 0.5)

gpu	epoch	Dice	IoU	Accuracy	Precision	Recall
1	49	0.7147	0.5604	0.9748	0.7972	0.6513
1	99	0.7208	0.5683	0.9750	0.7968	0.6615
1	199	0.7177	0.5645	0.9746	0.7873	0.6629
1	299	0.7138	0.5598	0.9745	0.7895	0.6552
2	49	**0.7232**	**0.5712**	0.9750	0.7728	0.6851
2	99	0.7212	0.5693	0.9748	0.7863	0.6704
2	199	0.7151	0.5620	0.9747	0.7953	0.6542
2	299	0.7143	0.5608	0.9745	0.7841	0.6608
4	49	0.7226	0.5701	0.9753	0.7892	0.6704
4	99	0.7222	0.5697	0.9745	0.7570	**0.6959**
4	199	0.7129	0.5595	0.9748	0.7988	0.6503
4	299	0.7137	0.5599	0.9746	0.7891	0.6563
8	49	0.7214	0.5689	**0.9758**	**0.8070**	0.6565
8	99	0.7172	0.5643	0.9752	0.8055	0.6513
8	199	0.7162	0.5630	0.9751	0.8016	0.6526
8	299	0.7133	0.5593	0.9747	0.7901	0.6553

Tabella 5.1: Metriche ottenute negli esperimenti di strong scaling per le diverse configurazioni e i checkpoint considerati sul test set, con soglia di confidenza pari a 0,5.

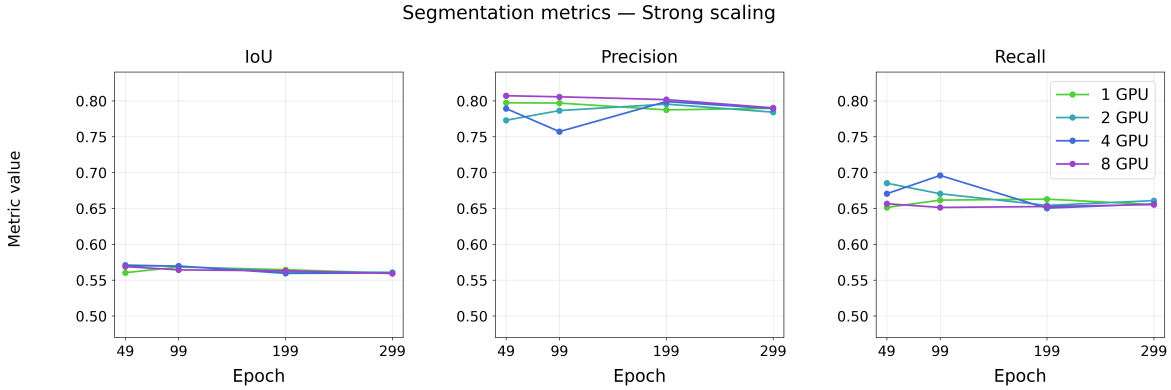


Figura 5.4: Andamento di IoU, Precision e Recall negli esperimenti di strong scaling per le diverse configurazioni e i checkpoint considerati, con soglia di confidenza pari a 0,5.

Nel complesso, le metriche sono rimaste su valori simili al variare del numero di GPU. In particolare, l'IoU ha mostrato variazioni contenute tra le diverse configurazioni e non è emerso un deterioramento della qualità della segmentazione all'aumentare delle risorse utilizzate.

Le differenze più evidenti hanno riguardato il bilanciamento tra Precision e Recall, che ha mostrato alcune oscillazioni tra i diversi checkpoint. Le variazioni osservate non mostrano però un andamento regolare all'aumentare delle risorse.

5.3.2 Weak scaling

Le metriche ottenute nelle configurazioni di weak scaling sono riportate nella Tabella 5.2 e in Figura 5.5.

Metrics across GPUs — Weak scaling (Release 1.4, conf. 0.5)

gpu	epoch	Dice	IoU	Accuracy	Precision	Recall
1	49	0.7226	**0.5700**	0.9749	0.7708	0.6859
1	99	0.7209	0.5679	0.9749	0.7832	0.6720
1	149	0.7189	0.5663	0.9748	0.7885	0.6658
1	199	0.7207	0.5682	0.9747	0.7708	0.6821
2	49	0.7187	0.5651	0.9747	0.7876	0.6645
2	99	0.7152	0.5617	0.9747	0.7890	0.6585
2	199	0.7177	0.5650	0.9749	0.7915	0.6612
2	299	0.7153	0.5617	0.9750	**0.8032**	0.6489
4	49	**0.7228**	0.5695	0.9745	0.7714	0.6851
4	99	0.7165	0.5627	0.9748	0.7898	0.6598
4	199	0.7183	0.5653	0.9745	0.7706	0.6778
4	299	0.7136	0.5596	0.9745	0.7897	0.6553
8	49	0.7174	0.5645	**0.9751**	0.7983	0.6567
8	99	0.7202	0.5677	0.9740	0.7513	**0.6974**
8	199	0.7110	0.5568	0.9742	0.7865	0.6535
8	299	0.7098	0.5558	0.9744	0.7942	0.6478

Tabella 5.2: Metriche ottenute negli esperimenti di weak scaling per le diverse configurazioni e i checkpoint considerati, con soglia di confidenza pari a 0,5.

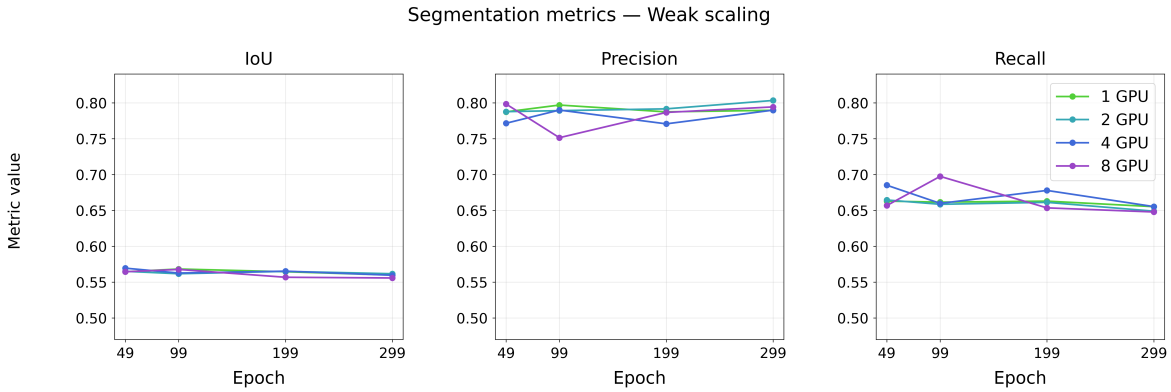


Figura 5.5: Andamento di IoU, Precision e Recall negli esperimenti di weak scaling per le diverse configurazioni e i checkpoint considerati, con soglia di confidenza pari a 0,5.

Anche nel weak scaling l'IoU ha mostrato variazioni contenute, con una lieve riduzione nel caso con il maggior numero di GPU.

Precision e Recall hanno invece presentato un andamento più variabile. La configurazione con 8 GPU ha mostrato le oscillazioni più marcate: in particolare, a un aumento del Recall è corrisposta una diminuzione della Precision, seguita nei checkpoint successivi da un'inversione di questo comportamento. Le altre configurazioni hanno presentato variazioni più contenute, pur senza mostrare un andamento completamente uniforme.

Nel complesso, anche nel weak scaling non è emerso un deterioramento della qualità della segmentazione all'aumentare del numero di GPU.

5.4 Analisi delle prestazioni computazionali

Dopo aver osservato che la parallelizzazione non ha determinato variazioni sostanziali né nelle metriche ottenute né nell'andamento delle curve di loss, sono state analizzate le prestazioni computazionali delle diverse configurazioni. Per ciascun addestramento sono stati considerati il tempo medio di esecuzione per iterazione e il tempo medio per epoca. Questi valori sono stati calcolati effettuando la media rispettivamente su tutte le iterazioni e su tutte le epoche disponibili per ciascuna esecuzione. Ogni addestramento è configurato per un massimo di 400 epoche: nello strong scaling queste corrispondono a 53.600 iterazioni per tutte le configurazioni, mentre nel weak scaling a 53.600, 26.800, 13.600 e 6.800 iterazioni rispettivamente per 1, 2, 4 e 8 GPU. Le medie e le deviazioni standard sono state quindi calcolate considerando tutte queste misurazioni. A partire dai tempi misurati sono stati inoltre calcolati lo speedup e l'efficienza, utilizzati per valutare il comportamento delle diverse configurazioni rispetto al caso a singola GPU. I risultati sono stati analizzati separatamente per gli esperimenti di strong scaling e weak scaling.

5.4.1 Strong scaling

I risultati relativi alle prestazioni computazionali dello strong scaling sono riportati nelle Figure 5.6 e 5.7. In questa configurazione il batch size globale è stato mantenuto costante a 24, mentre il batch assegnato a ciascuna GPU è stato progressivamente ridotto all'aumentare delle risorse utilizzate. Il numero di iterazioni per epoca è rimasto quindi invariato.

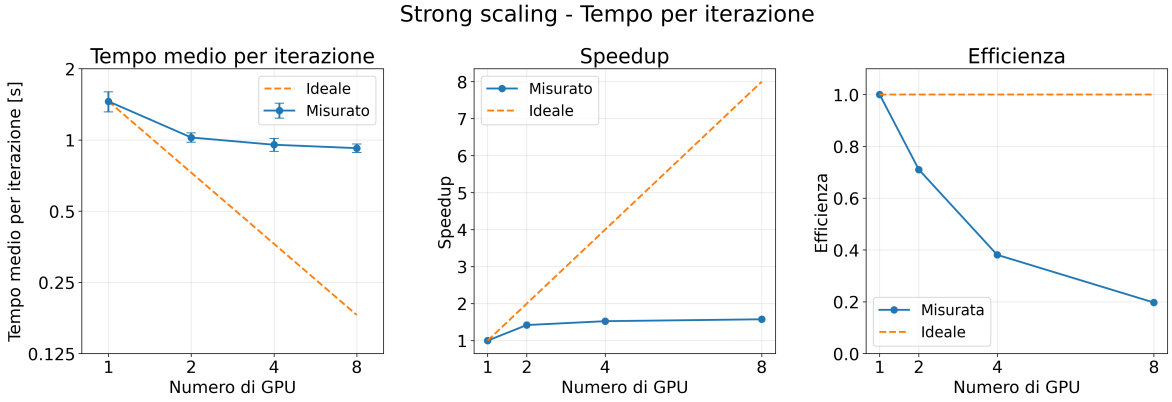


Figura 5.6: Tempo medio per iterazione, speedup ed efficienza negli esperimenti di strong scaling al variare del numero di GPU. Le barre di errore indicano la deviazione standard del tempo per iterazione.

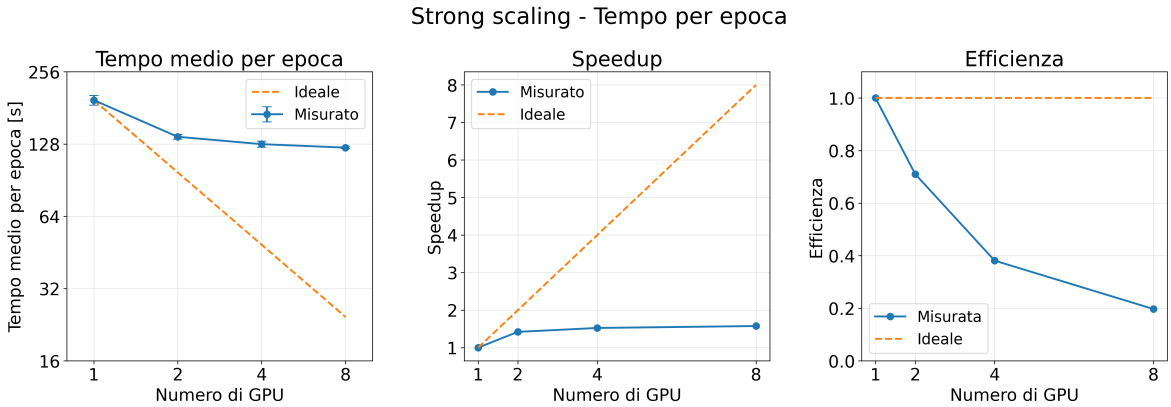


Figura 5.7: Tempo medio per epoca, speedup ed efficienza negli esperimenti di strong scaling al variare del numero di GPU. Le barre di errore indicano la deviazione standard del tempo per epoca.

I tempi di esecuzione sono diminuiti all'aumentare del numero di GPU, sia considerando la singola iterazione sia l'intera epoca. La riduzione maggiore è stata osservata nel passaggio da 1 a 2 GPU. Con 4 GPU, invece, il tempo medio per epoca è sceso solo da circa 137 a 128 secondi, mostrando un beneficio molto più ridotto rispetto al passaggio precedente. Con 8 GPU il miglioramento è stato ancora più limitato, con un tempo di circa 124 secondi nonostante il raddoppio delle risorse. L'andamento del tempo per epoca rispecchia quello osservato per la singola iterazione, poiché nello strong scaling il numero di iterazioni necessarie per completare un'epoca rimane invariato.

Questo comportamento risulta ancora più evidente osservando lo speedup e l'efficienza. Lo speedup è cresciuto all'aumentare del numero di GPU, ma si è allontanato rapidamente dal comportamento ideale e ha mostrato una tendenza alla saturazione già a partire dalla configurazione con 4 GPU. Allo stesso tempo, l'efficienza è diminuita soprattutto con 4 e 8 GPU, perché l'aumento delle risorse non ha portato a una riduzione altrettanto significativa dei tempi di esecuzione.

Una delle cause è legata alla suddivisione del batch tra le GPU. Poiché nello strong scaling il batch size globale è mantenuto costante a 24, quello locale elaborato da ciascuna GPU passa rispettivamente a 24, 12, 6 e 3 campioni nelle configurazioni con 1, 2, 4 e 8 GPU. All'aumentare delle risorse, ciascuna GPU dispone quindi di una quantità sempre minore di lavoro per iterazione, riducendo la possibilità di sfruttarne pienamente la capacità di calcolo.

A questo si aggiunge l'overhead dovuto alla comunicazione tra i diversi processi, necessaria per la riduzione e la sincronizzazione dei gradienti, cioè per ottenere un unico gradiente globale a partire da quelli calcolati dalle diverse GPU. Poiché il lavoro computazionale svolto da ciascuna GPU diminuisce, il peso relativo di queste operazioni diventa progressivamente maggiore e limita ulteriormente lo speedup. Nella configurazione con 8 GPU, inoltre, l'addestramento coinvolge due nodi di calcolo e richiede quindi anche comunicazioni tra nodi, che possono contribuire ad appesantire lo scambio di informazioni tra processi. Queste co-

municazioni risultano infatti più onerose rispetto a quelle tra GPU dello stesso nodo, poiché la rete di interconnessione presenta una latenza maggiore e una banda inferiore rispetto ai collegamenti locali.

5.4.2 Weak scaling

L'esperimento riproduce le condizioni di weak scaling a livello della singola iterazione, mantenendo costante il carico computazionale per GPU, ma non costituisce un weak scaling classico a livello dell'intero dataset, poiché la dimensione complessiva del dataset rimane fissa. In particolare, ogni GPU ha elaborato un batch size di 24 campioni; aumentando il numero di GPU, è quindi aumentato anche il batch size globale e il numero di iterazioni necessarie per completare un'epoca è diminuito. Il tempo medio per iterazione costituisce quindi la misura principale dell'analisi, mentre le prestazioni per epoca permettono comunque di valutare quanto si riduce il tempo complessivo di addestramento sul dataset disponibile. L'analisi è riportata nelle Figure 5.8 e 5.9.

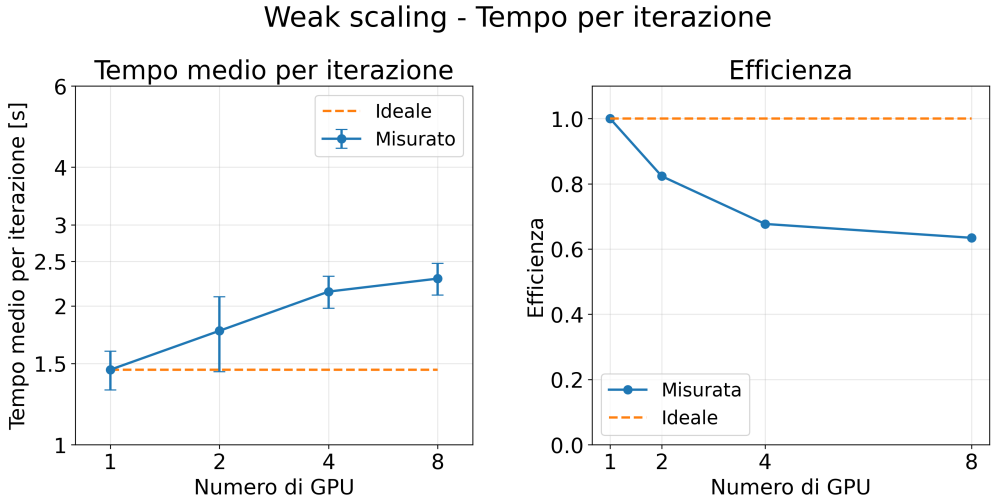


Figura 5.8: Tempo medio per iterazione ed efficienza negli esperimenti di weak scaling al variare del numero di GPU, mantenendo costante il batch size per GPU. Le barre di errore indicano la deviazione standard del tempo per iterazione.

Nel weak scaling ideale, mantenendo costante il batch size per GPU, il tempo necessario per completare un'iterazione dovrebbe rimanere costante all'aumentare del numero di GPU, con un'efficienza prossima a 1. Dal grafico ottenuto, il tempo medio per iterazione è rimasto dello stesso ordine di grandezza, ma è aumentato progressivamente rispetto al caso ideale. Questo andamento è dovuto in parte all'overhead introdotto dall'addestramento distribuito: durante la *backpropagation* i gradienti devono essere sincronizzati tra i diversi processi a ogni iterazione e il costo di queste operazioni aumenta al crescere del numero di GPU. Di conseguenza, pur mantenendo costante il batch size per GPU, l'aumento del tempo per iterazione ha comportato una progressiva riduzione dell'efficienza.

Per quanto riguarda il tempo medio per epoca, nel weak scaling ideale la dimensione complessiva del problema dovrebbe crescere insieme al numero di GPU, mantenendo costante il numero di iterazioni necessarie per completare un'epoca. In questo caso non è stato possibile aumentare ulteriormente la dimensione del dataset, poiché tutte le immagini disponibili erano già state utilizzate per costruire i set di training, validation e test. In ambito radioastronomico, infatti, ottenere o produrre nuovi dati adatti all'addestramento può richiedere tempi e risorse considerevoli.

L'aumento del batch size globale ha ridotto il numero di iterazioni per epoca da 134 con una GPU a 67, 34 e infine 17 con otto GPU. La riduzione del tempo per epoca è dipesa quindi principalmente dal minor numero di iterazioni necessarie. Il confronto del tempo per epoca non deve essere interpretato come una misura di weak scaling, ma come una misura pratica della riduzione del tempo necessario a completare un'epoca sul dataset disponibile.

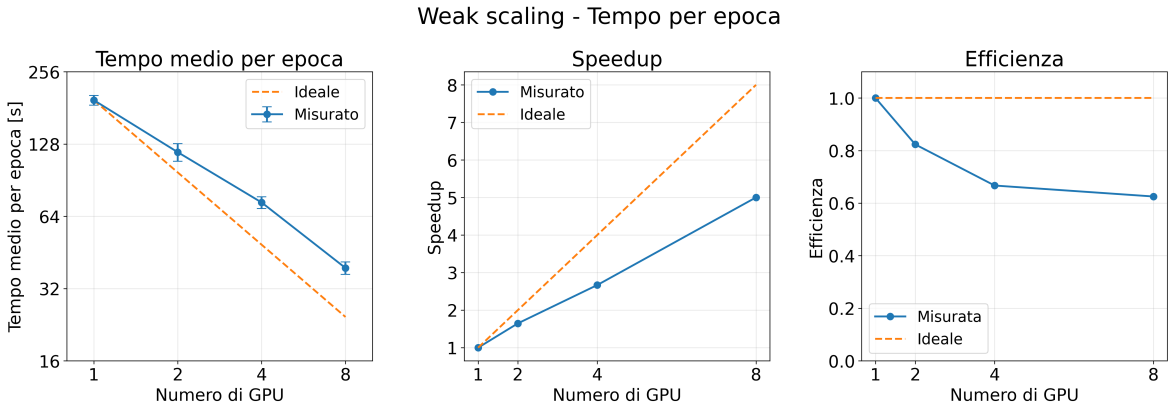


Figura 5.9: Tempo medio per epoca, speedup ed efficienza al variare del numero di GPU, mantenendo costante il batch size per GPU e utilizzando un dataset di dimensione fissa. Le barre di errore indicano la deviazione standard del tempo per epoca.

Per quantificare la riduzione del tempo per epoca all'aumentare del numero di GPU, è stato calcolato lo speedup per epoca come:

$$S_{\text{epoch}}(p) = \frac{T_{\text{epoch}}(1)}{T_{\text{epoch}}(p)}. \tag{5.1}$$

L'efficienza è stata invece calcolata come:

$$E_{\text{epoch}}(p) = \frac{T_{\text{epoch}}(1)}{p \cdot T_{\text{epoch}}(p)}. \tag{5.2}$$

Nel caso ideale, il tempo per epoca diminuirebbe proporzionalmente al numero di GPU, corrispondendo a uno speedup pari a p e a un'efficienza pari a 1. Nei risultati ottenuti, lo speedup è aumentato all'aumentare delle GPU, rimanendo tuttavia inferiore al comportamento ideale, mentre l'efficienza è progressivamente diminuita. Questo andamento deriva dall'aumento del tempo per iterazione osservato negli stessi esperimenti di weak scaling.

5.5 Valutazione complessiva

Nel complesso, gli esperimenti hanno mostrato che l'addestramento su più GPU permette di ridurre i tempi di esecuzione senza compromettere in modo evidente la qualità delle predizioni. Le metriche ottenute nelle diverse configurazioni sono rimaste infatti abbastanza simili, pur mostrando alcune oscillazioni tra i diversi checkpoint. Anche le curve di training e validation loss hanno mantenuto un andamento complessivamente simile tra le diverse configurazioni, con differenze generalmente contenute. Dal momento che per ciascuna configurazione è stata eseguita una sola prova, non è però possibile stabilire con certezza se queste variazioni siano effettivamente legate al diverso numero di GPU utilizzate, oppure siano dovute alla normale variabilità stocastica del processo di training o, nel caso dello strong scaling, alla variazione del batch size globale.

Considerando insieme qualità delle predizioni e prestazioni computazionali, la scelta del numero di GPU dipende quindi dall'obiettivo che si vuole raggiungere. Se si vuole privilegiare la stabilità e la qualità dei risultati, l'utilizzo di 1 GPU rimane una valida scelta. Se invece si vuole ridurre il tempo di esecuzione, dagli esperimenti di strong scaling emerge che il passaggio da 1 a 2 GPU rappresenta il compromesso migliore. In questa configurazione si osserva infatti la riduzione più significativa dei tempi, mentre con 4 e 8 GPU i benefici aggiuntivi risultano limitati rispetto all'aumento delle risorse impiegate, come indicato dalla progressiva diminuzione dell'efficienza parallela.

L'esecuzione distribuita offre inoltre la possibilità di utilizzare batch size globali più elevati, mantenendo costante il numero di campioni elaborati da ciascuna GPU. Questa caratteristica può risultare utile per modelli o configurazioni di addestramento che richiedono o beneficiano di batch size elevati, superando i limiti di memoria di una singola GPU. Nel caso di TUNA, tuttavia, gli esperimenti svolti non hanno evidenziato un vantaggio significativo associato all'aumento del batch size globale. Per questo motivo, oltre al tempo di esecuzione, è importante considerare anche l'efficienza nell'utilizzo delle risorse. L'impiego di un numero maggiore di GPU a fronte di una bassa efficienza parallela comporta infatti un maggiore utilizzo di risorse per ottenere benefici prestazionali limitati e può tradursi anche in un consumo energetico più elevato, sebbene questo aspetto non sia stato misurato direttamente negli esperimenti svolti.

La scelta del numero di GPU va quindi fatta considerando insieme qualità dei risultati, tempi di esecuzione ed efficienza nell'utilizzo delle risorse.

Capitolo 6

Mixed precision

6.1 Configurazioni sperimentali

L'analisi della mixed precision, introdotta nella Sezione 2.5, è stata svolta confrontando tre formati: FP32, utilizzato come riferimento, FP16 e BF16. Tutti gli esperimenti sono stati eseguiti su una singola GPU della partizione Booster di Leonardo, così da valutare l'effetto della precisione numerica senza introdurre anche quello dell'esecuzione distribuita.

Gli altri parametri sono rimasti invariati rispetto al modello di riferimento, così da poter confrontare direttamente i risultati. Per le esecuzioni in FP16 e BF16, Accelerate è stato configurato specificando il formato di mixed precision da utilizzare. Durante l'addestramento, Accelerate ha quindi gestito l'utilizzo della mixed precision attraverso i meccanismi di *automatic mixed precision* (AMP) di PyTorch. Nel caso di FP16 è stato inoltre utilizzato il *gradient scaling*, per ridurre il rischio di underflow durante il calcolo dei gradienti.

6.2 Confronto delle precisioni

Al termine degli addestramenti, le tre configurazioni sono state confrontate prima attraverso le metriche di segmentazione già utilizzate nei capitoli precedenti e poi considerando i tempi di esecuzione. In questo modo è stato possibile verificare se l'uso di una precisione ridotta ha permesso di velocizzare l'addestramento senza peggiorare troppo la qualità delle predizioni.

6.2.1 Metriche di segmentazione

Per valutare l'effetto della precisione numerica sulle metriche di segmentazione, queste sono state calcolate sul test set utilizzando una soglia di confidenza pari a 0,5. Il confronto tra FP32, FP16 e BF16 è stato effettuato sui checkpoint alle epoche 49, 99, 149, 199 e 299, così da confrontare le tre configurazioni nelle diverse fasi dell'addestramento. I risultati completi

sono riportati nella Tabella 6.1, mentre la Figura 6.1 mostra l'andamento di IoU, Precision e Recall.

Metrics across numerical precisions - Release 1.4, conf. 0.5

precision	epoch	Dice	IoU	Accuracy	Precision	Recall
FP32	49	0.7147	0.5604	0.9748	0.7972	0.6513
FP32	99	**0.7208**	**0.5683**	**0.9750**	0.7968	0.6615
FP32	149	0.7171	0.5636	0.9747	0.7924	0.6584
FP32	199	0.7177	0.5645	0.9746	0.7873	**0.6629**
FP32	299	0.7138	0.5598	0.9745	0.7895	0.6552
FP16	49	0.6608	0.4975	0.9729	**0.8100**	0.5630
FP16	99	0.6923	0.5346	0.9730	0.7542	0.6482
FP16	149	0.6839	0.5250	0.9728	0.7725	0.6198
FP16	199	0.6863	0.5276	0.9730	0.7763	0.6210
FP16	299	0.6850	0.5264	0.9729	0.7708	0.6235
BF16	49	0.6893	0.5306	0.9731	0.7585	0.6376
BF16	99	0.6853	0.5262	0.9725	0.7604	0.6298
BF16	149	0.6909	0.5325	0.9725	0.7395	0.6546
BF16	199	0.6909	0.5323	0.9725	0.7412	0.6524
BF16	299	0.6903	0.5318	0.9728	0.7468	0.6479

Tabella 6.1: Metriche di segmentazione ottenute sul test set con le configurazioni FP32, FP16 e BF16 ai diversi checkpoint, utilizzando una soglia di confidenza pari a 0,5.

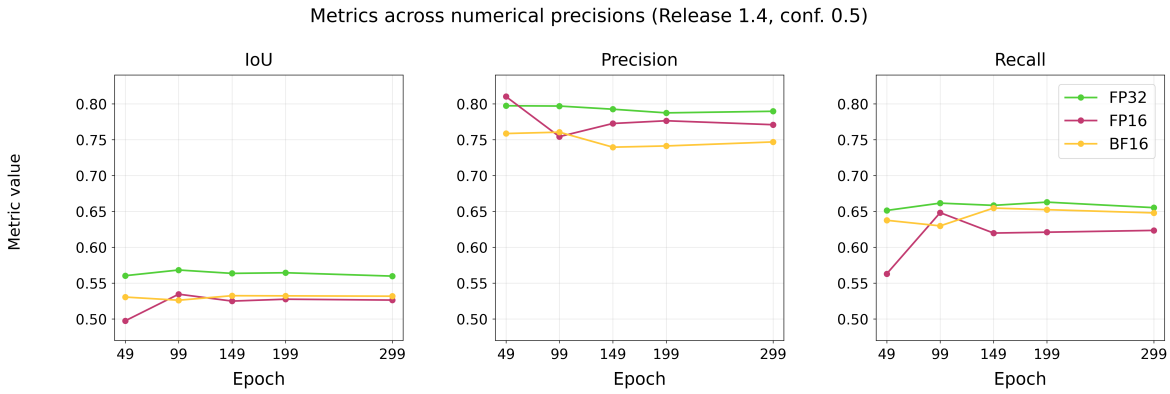


Figura 6.1: Andamento di IoU, Precision e Recall sul test set per le configurazioni FP32, FP16 e BF16 ai diversi checkpoint, con soglia di confidenza pari a 0,5.

Nel complesso, FP32 ha fornito le prestazioni migliori, ottenendo valori più elevati di Dice, IoU e Recall rispetto alle configurazioni a precisione ridotta. Le metriche sono inoltre rimaste sostanzialmente stabili tra i diversi checkpoint. FP16 ha invece mostrato una maggiore variabilità, soprattutto nelle prime fasi dell'addestramento: l'elevata Precision iniziale è stata accompagnata da un Recall più basso, indicando che il modello ha prodotto meno falsi positivi, ma non ha riconosciuto una parte dell'emissione presente. BF16 ha presentato un andamento più stabile rispetto a FP16, con un maggiore equilibrio tra Precision e Recall e valori generalmente superiori di Dice e IoU. Il proseguimento dell'addestramento fino all'epoca 299 non ha comunque prodotto miglioramenti evidenti per nessuna delle due configurazioni a precisione ridotta.

L'Accuracy è risultata molto simile in tutte le configurazioni. A causa del forte sbilanciamento del dataset verso la classe di background, questa metrica è però poco sensibile alle

differenze nella capacità di riconoscere l'emissione e deve quindi essere interpretata insieme a Dice, IoU, Precision e Recall. Nel complesso, FP16 e BF16 hanno raggiunto prestazioni simili, ma entrambe inferiori a quelle di FP32; tra le due, BF16 ha mostrato risultati più stabili nei diversi checkpoint.

Non è possibile attribuire con certezza le differenze osservate alla precisione numerica, poiché per ciascuna configurazione è stato eseguito un solo addestramento e i risultati potrebbero dipendere anche dalla variabilità del processo.

6.2.2 Tempi di esecuzione

Per confrontare i tempi di esecuzione delle tre configurazioni è stato considerato il tempo medio necessario per completare un'iterazione. La Figura 6.2 riporta sia i tempi misurati, con la relativa deviazione standard, sia il rapporto tra il tempo ottenuto con FP32 e quello delle altre configurazioni. Un rapporto superiore a 1 indica quindi una riduzione del tempo di esecuzione rispetto a FP32.

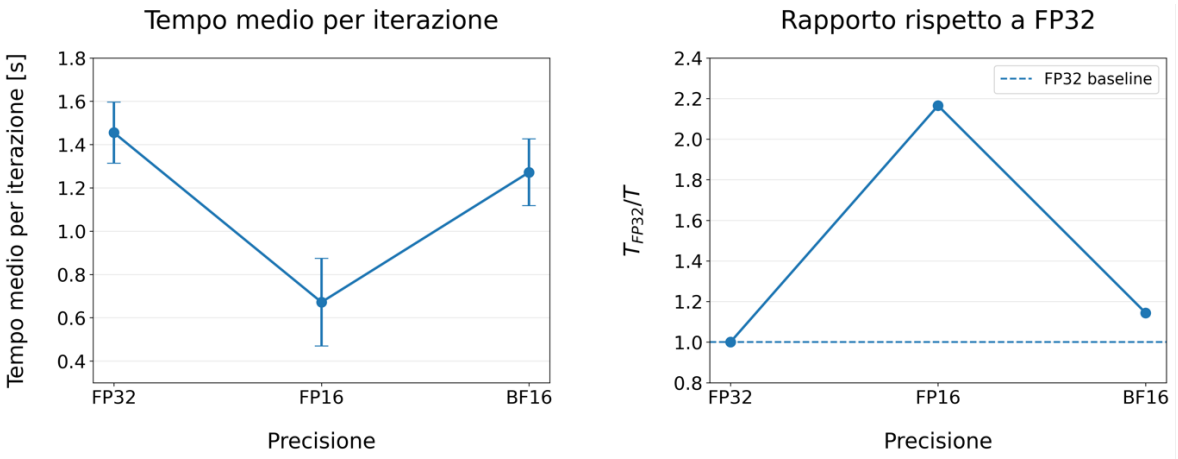


Figura 6.2: Tempo medio per iterazione ottenuto con FP32, FP16 e BF16 e relativo rapporto rispetto alla configurazione FP32. Le barre di errore indicano la deviazione standard del tempo per iterazione.

FP16 ha mostrato la maggiore riduzione del tempo per iterazione, risultando più di due volte più veloce rispetto a FP32. Anche BF16 ha ridotto il tempo di esecuzione, ma il miglioramento è stato minimo. Nelle condizioni sperimentali considerate, FP16 è quindi risultata la configurazione più vantaggiosa dal punto di vista dei tempi di addestramento.

6.3 Valutazione complessiva

I risultati ottenuti mostrano che la precisione numerica influisce sia sulla qualità delle predizioni sia sui tempi di addestramento. FP32 ha ottenuto i risultati migliori nelle metriche di segmentazione, ma ha richiesto un tempo maggiore per completare un'iterazione. FP16

ha invece prodotto valori inferiori di Dice, IoU e Recall, con un calo comunque contenuto rispetto a FP32, ma ha permesso di ridurre di oltre la metà il tempo per iterazione. BF16 ha ottenuto metriche simili a quelle di FP16, ma ha ridotto solo leggermente il tempo di esecuzione rispetto a FP32 e non ha quindi offerto un vantaggio evidente rispetto alle altre due configurazioni.

La scelta tra FP32 e FP16 dipende quindi dalle esigenze: FP32 permette di ottenere predizioni più accurate, mentre FP16 consente di ridurre notevolmente i tempi e il costo computazionale dell'addestramento. Inoltre, l'uso di formati a 16 bit richiede meno memoria per rappresentare i dati, un aspetto utile soprattutto quando il modello diventa particolarmente oneroso.

Conclusioni

La tesi si è concentrata sull'analisi di TUNA (*TransUNet Network for Radio Astronomy*) sia dal punto di vista della qualità della segmentazione sia delle prestazioni computazionali. Il modello è stato addestrato utilizzando dati simulati, costruiti in modo da riprodurre il più fedelmente possibile le caratteristiche delle osservazioni radioastronomiche. Inizialmente sono stati confrontati diversi valori del learning rate, mantenendo invariati gli altri principali parametri di addestramento. Il confronto delle metriche ha portato alla scelta di una configurazione di riferimento, utilizzata poi per le analisi successive.

Il modello selezionato è stato applicato anche alle immagini osservative descritte nel Capitolo 4. Sono stati inizialmente considerati tre ammassi della survey LoTSS DR2, confrontando qualitativamente le predizioni di TUNA con immagini a diversa risoluzione, nelle quali l'emissione diffusa risulta più evidente. I tre casi hanno permesso di osservare il comportamento del modello in presenza di morfologie differenti, sorgenti luminose e strutture a bassa brillanza superficiale.

Per approfondire la valutazione su dati reali, è stato inoltre considerato un intero puntamento osservativo, il campo P42Hetdex07. Le regioni individuate da TUNA sono state confrontate con sorgenti già presenti nei cataloghi astronomici indipendenti considerati. Il confronto ha mostrato un'ottima corrispondenza con le sorgenti note, ma anche numerosi oggetti non compresi nei cataloghi osservativi considerati. Questi casi non possono essere interpretati direttamente come nuove sorgenti ed è necessaria un'ulteriore specifica analisi per chiarirne la natura.

La configurazione scelta è stata successivamente utilizzata per studiare il comportamento dell'addestramento all'aumentare del numero di GPU. Gli esperimenti di strong scaling e weak scaling hanno mostrato una riduzione dei tempi di esecuzione con l'aumento delle risorse, anche se il miglioramento non è risultato proporzionale al numero di GPU utilizzate. All'aumentare delle GPU diminuisce infatti il lavoro assegnato a ciascun acceleratore e aumenta il peso della comunicazione tra i processi. Nella configurazione a 8 GPU si aggiunge inoltre il costo della comunicazione tra i due nodi. Le metriche di segmentazione sono invece rimaste complessivamente confrontabili tra le diverse esecuzioni.

È stato infine analizzato l'effetto della precisione numerica confrontando FP32, FP16 e BF16 durante l'addestramento. I risultati hanno mostrato un compromesso tra qualità delle predizioni e tempi di esecuzione. FP32 ha ottenuto complessivamente i valori migliori nel-

le metriche di segmentazione, mentre FP16 ha permesso di ridurre maggiormente il tempo necessario per completare un'iterazione. BF16 ha mostrato metriche simili a quelle ottenute con FP16, ma con una riduzione dei tempi più contenuta. La precisione numerica rappresenta quindi un altro aspetto da considerare nell'ottimizzazione dell'addestramento, scegliendo la configurazione più adatta in base all'obiettivo che si vuole privilegiare: qualità delle predizioni oppure riduzione dei tempi di esecuzione.

Limiti del lavoro

I risultati ottenuti presentano alcuni limiti da tenere in considerazione. Per ciascuna configurazione di parallelizzazione e di precisione numerica è stato eseguito un solo addestramento, a causa delle risorse computazionali limitate a disposizione. Non è quindi possibile stabilire con certezza quanto le differenze osservate dipendano dalle configurazioni considerate e quanto dalla normale variabilità stocastica del training.

La valutazione sulle immagini osservative è rimasta principalmente qualitativa, poiché per questi dati non è disponibile una *ground truth* binaria con cui confrontare direttamente le predizioni. Il confronto con immagini a diversa risoluzione e, nel caso dell'analisi sull'intero puntamento osservativo, con sorgenti già presenti nei cataloghi astronomici ha comunque fornito indicazioni utili sul comportamento del modello, anche se non è stato possibile calcolare le stesse metriche utilizzate sui dati simulati.

Gli esperimenti sono stati inoltre eseguiti su una specifica architettura hardware e con una determinata configurazione del dataset e dell'addestramento. I risultati relativi alle prestazioni computazionali sono quindi legati all'ambiente utilizzato e potrebbero cambiare con hardware, dataset o configurazioni differenti.

Sviluppi futuri

In futuro, TUNA sarà applicato a un numero maggiore di immagini osservative reali, così da valutarne il comportamento su un insieme più ampio di casi. Questo permetterà anche di confrontare in modo più sistematico le regioni individuate dal modello con le sorgenti già presenti nei cataloghi e di approfondire i casi in cui non verrà trovata una corrispondenza.

Un ulteriore sviluppo sarà l'introduzione di un secondo modello dedicato alla classificazione delle sorgenti. TUNA continuerà a occuparsi dell'individuazione e della delimitazione delle regioni di emissione, mentre una rete successiva assegnerà a ciascuna sorgente una classe in base alle sue caratteristiche fisiche o morfologiche. In questo modo sarà possibile non solo individuare automaticamente le sorgenti, ma anche riconoscerle.

Dal punto di vista computazionale, gli esperimenti di parallelizzazione e mixed precision verranno ripetuti più volte per ciascuna configurazione, così da limitare l'effetto della variabilità stocastica del training e rendere più affidabile il confronto tra i risultati. Verranno inoltre analizzate combinazioni diverse di iperparametri, modificando ad esempio il batch si-

ze e altri parametri dell’addestramento, per individuare le configurazioni che permetteranno di sfruttare in modo più efficace le risorse disponibili.

Appendice A

Script utilizzati

In questa appendice sono riportati alcuni degli script SLURM utilizzati per eseguire le principali fasi sperimentali descritte nei capitoli precedenti. In particolare, sono inclusi lo script utilizzato per eseguire gli addestramenti con diversi valori di learning rate e trovare il modello migliore su cui lavorare, quello impiegato per eseguire l'inferenza dei modelli addestrati su più GPU e lo script utilizzato per il successivo calcolo delle metriche di segmentazione.

A.1 Addestramento con diversi learning rate

Lo script seguente è stato utilizzato per avviare in parallelo quattro addestramenti indipendenti sulla Release 1.4. Lo script specifica inoltre il percorso dei pesi pre-addestrati utilizzati per inizializzare il modello. In questo modo l'addestramento non parte da un'inizializzazione completamente casuale, ma sfrutta parametri già appresi in una fase precedente. Il parametro `deterministic=0` indica invece che non viene forzata l'esecuzione deterministica delle operazioni di PyTorch e CUDA. Questa scelta consente di utilizzare le implementazioni disponibili più efficienti, ma può introdurre piccole variazioni tra esecuzioni successive anche mantenendo invariati gli stessi parametri.

Listing A.1: Script SLURM utilizzato per gli addestramenti con i diversi valori di learning rate.

```
#!/bin/bash
#SBATCH -A INA24_C3T02
#SBATCH -p boost_usr_prod
#SBATCH -N 1
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4
#SBATCH -t 24:00:00
```

```
SCRIPT=/leonardo/home/userexternal/abeccati/tunet/train.py
PRETRAIN=/leonardo/home/userexternal/abeccati/model/vit_checkpoint/imagenet21k

LRS=(0.01 0.005 0.001 0.0005)

for LR in "${LRS[@]}"; do
  srun -N 1 -n 1 --gres=gpu:1 \
    python $SCRIPT \
    --dataset TUNA \
    --base_lr $LR \
    --max_epochs 400 \
    --batch_size 24 \
    --vit_name R50-ViT-B_16 \
    --split 0 \
    --img_size 512 \
    --deterministic 0 &
done

wait
```

A.2 Inferenza dei modelli multi-GPU

Lo script seguente è stato utilizzato per eseguire l'inferenza sui modelli ottenuti negli esperimenti multi-GPU. Per ogni addestramento vengono indicati il modello da utilizzare e i checkpoint da analizzare. Le predizioni ottenute per ciascun checkpoint vengono salvate in cartelle separate, così da mantenere distinti i risultati delle diverse configurazioni. Le diverse inferenze vengono eseguite contemporaneamente, ciascuna utilizzando una singola GPU, così da ridurre il tempo necessario per completare l'analisi di tutti i modelli considerati.

Listing A.2: Script SLURM utilizzato per eseguire l'inferenza dei modelli addestrati con Accelerate.

```
#!/bin/bash
#SBATCH -A INA24_C3T02
#SBATCH -p boost_usr_prod
#SBATCH -N 1
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4
#SBATCH -t 24:00:00
```

```
SCRIPT=/leonardo/home/userexternal/abeccati/tunet_first/merge_infer_segmentation.py
INPUT=/leonardo_work/INA24_C3T02/release1.4/LOFAR_HBA_6asec/clean

MODEL_DIRS=(
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs24_lr0.005
    _512_split0__1GPU_det0"
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs12_lr0.0025
    _512_split0__2GPU_200726"
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs6_lr0.00125
    _512_split0__4GPU_200726"
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs3_lr0.000625
    _512_split0__8GPU_200726"
)

EPOCHS=(49 99 149 199 299 399)

COUNT=0

for MODEL_DIR in "${MODEL_DIRS[@]}"
do
  for EPOCH in "${EPOCHS[@]}"
  do
    CHECKPOINT="${MODEL_DIR}/epoch_${EPOCH}_0.pth"
    OUTPUT="${MODEL_DIR}/inference_epoch${EPOCH}_allcheckpoints"

    srun --exclusive \
      --nodes=1 \
      --ntasks=1 \
      --gres=gpu:1 \
      --unbuffered \
      python "$SCRIPT" \
      --input_dir "$INPUT" \
      --output_dir "$OUTPUT" \
      --checkpoint "$CHECKPOINT" \
      --first_id 251 \
```

```
        --last_id 299 &

    COUNT=$((COUNT + 1))

    if (( COUNT % 4 == 0 )); then
      wait
    fi
  done
done

wait
```

A.3 Calcolo delle metriche dei modelli multi-GPU

Lo script seguente è stato utilizzato per calcolare le metriche di segmentazione a partire dalle predizioni generate nella fase di inferenza. Per ciascun modello e checkpoint viene individuata la relativa cartella contenente le predizioni e viene definito un percorso separato per il salvataggio dei risultati. Lo script specifica inoltre la soglia di confidenza utilizzata per la binarizzazione delle predizioni. Anche in questo caso più calcoli vengono avviati in parallelo, così da elaborare contemporaneamente diverse configurazioni.

Listing A.3: Script SLURM utilizzato per il calcolo delle metriche.

```
#!/bin/bash
#SBATCH -A INA24_C3T02
#SBATCH -p boost_usr_prod
#SBATCH -N 1
#SBATCH --ntasks-per-node=4
#SBATCH -t 24:00:00

SCRIPT=/leonardo/home/userexternal/abeccati/tunet_first/metrics_simulated_rev1.py

MODEL_DIRS=(
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs24_lr0.005
    _512_split0__1GPU_det0"
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs12_lr0.0025
    _512_split0__2GPU_200726"
```

```
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs6_lr0.00125
    _512_split0__4GPU_200726"
  "/leonardo_scratch/large/userexternal/abeccati/model_release1.4/model_accelerate/
    TUNA512_pretrain_R50-ViT-B_16_skip3_epo400_bs3_lr0.000625
    _512_split0__8GPU_200726"
)

LRS=(0.005 0.0025 0.00125 0.000625)

EPOCHS=(49 99 149 199 299)

CONF=0.5

COUNT=0

for i in "${!MODEL_DIRS[@]}"
do
  MODEL_DIR="${MODEL_DIRS[$i]}"
  LR="${LRS[$i]}"

  for EPOCH in "${EPOCHS[@]}"
  do
    PRED_DIR="${MODEL_DIR}/inference_epoch${EPOCH}_allcheckpoints"
    OUT_DIR="${MODEL_DIR}/metrics_epoch${EPOCH}_allcheckpoints_conf${
    CONF}"

    if [[ ! -d "$PRED_DIR" ]]; then
      echo "Directory non trovata: $PRED_DIR"
      exit 1
    fi

    srun --exclusive \
      --nodes=1 \
      --ntasks=1 \
      --unbuffered \
      python "$SCRIPT" \
      --pred_dir "$PRED_DIR" \
      --output_dir "$OUT_DIR" \
```

```
        −−lr ”$LR” \
        −−epoch ”$EPOCH” \
        −−conf ”$CONF” &

    COUNT=$((COUNT + 1))

    if (( COUNT % 4 == 0 )); then
      wait
    fi
  done
done

wait
```

Bibliografia

[1] N. Sanvitale, C. Gheller, F. Vazza, A. Bonafede, V. Cuciti, E. De Rubeis, F. Govoni, M. Murgia, and V. Vacca, "Mapping diffuse radio sources using TUNA: A transformer-based deep learning approach," *Monthly Notices of the Royal Astronomical Society*, vol. 541, no. 4, pp. 3479–3493, 2025.

[2] M. J. Hardcastle and J. H. Croston, "Radio galaxies and feedback from AGN jets," *New Astronomy Reviews*, vol. 88, p. 101539, 2020.

[3] K. Knowles *et al.*, "The MeerKAT galaxy cluster legacy survey. i. survey overview and highlights," *Astronomy & Astrophysics*, vol. 657, p. A56, 2022.

[4] NASA, "Atmospheric electromagnetic transmittance or opacity," Wikimedia Commons, versione vettoriale realizzata da Mysid, 2026, consultato il 4 agosto 2026. [Online]. Available: https://commons.wikimedia.org/wiki/File:Atmospheric_electromagnetic_opacity.svg

[5] caprowsky, "Radio telescopio – panoramio," Wikimedia Commons, fotografia distribuita con licenza Creative Commons Attribution 3.0 Unported, 2026, consultato il 4 agosto 2026. [Online]. Available: https://commons.wikimedia.org/wiki/File:Radio_telescopio_-_panoramio.jpg

[6] M. P. van Haarlem *et al.*, "LOFAR: The LOw-Frequency ARray," *Astronomy & Astrophysics*, vol. 556, p. A2, 2013.

[7] D. J. Wilner, "Radio astronomy and interferometry fundamentals," ANITA Lecture Series, 2015. [Online]. Available: https://www.youtube.com/watch?v=0TwnZhiEc3A

[8] G. Moro and R. Pasolini, "Applicazioni predittive con regressione lineare," Materiale didattico del corso Programmazione di Applicazioni Data Intensive, Università di Bologna.

[9] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUNet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[10] 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*, ser. Lecture Notes in Computer Science, vol. 9351. Springer, 2015, pp. 234–241.

[11] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

[12] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[13] M. Marzolla, "Performance evaluation of parallel programs," Materiale didattico del corso High Performance Computing, Università di Bologna, 2025.

[14] PyTorch, "Automatic mixed precision package – torch.amp," PyTorch Documentation, accessed: 2026-08-28. [Online]. Available: https://docs.pytorch.org/docs/stable/amp.html

[15] NVIDIA, "Train with mixed precision," NVIDIA Documentation, accessed: 2026-08-28. [Online]. Available: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html

[16] CINECA, "Leonardo," HPC CINECA, 2026, consultato il 5 agosto 2026. [Online]. Available: https://www.hpc.cineca.it/systems/hardware/leonardo/

[17] W. A. Joye, E. Mandel, A. Fruscione *et al.*, "SAOImageDS9: Applicazione per l'imaging astronomico e la visualizzazione dei dati," 2003, software disponibile su http://ds9.si.edu/. [Online]. Available: http://ds9.si.edu/

[18] A. R. Offringa, B. McKinley, N. Hurley-Walker *et al.*, "WSClean: An Implementation of a Fast, Generic Wide-Field Imager for Radio Astronomy," *Monthly Notices of the Royal Astronomical Society*, vol. 444, no. 1, pp. 606–619, 2014.

[19] L. Clews, J. H. Croston, H. Dickinson, B. Mingo, M. J. Hardcastle, B. Barkus, J. M. G. H. J. de Jong, and H. J. A. Röttgering, "Radio-loud AGN morphology and host-galaxy properties in the LOFAR two-metre sky survey data release 2," *Monthly Notices of the Royal Astronomical Society*, vol. 541, no. 4, pp. 3452–3467, 2025.

[20] R. I. J. Mostert, M. S. S. L. Oei, B. Barkus, L. Alegre, M. J. Hardcastle, K. J. Duncan, H. J. A. Röttgering, R. J. van Weeren, and M. Horton, "Constraining the giant radio galaxy population with machine learning and bayesian inference," *Astronomy & Astrophysics*, vol. 691, p. A185, 2024.

[21] A. Botteon, T. W. Shimwell, R. Cassano, V. Cuciti, X. Zhang, L. Bruno, L. Camillini, R. Natale, A. Jones, F. Gastaldello, A. Simionescu, M. Rossetti, H. Akamatsu, R. J. van Weeren, G. Brunetti, M. Brüggen, C. Groeneveld, D. N. Hoang, M. J. Hardcastle, A. Ignesti, G. Di Gennaro, A. Bonafede, A. Drabent, H. J. A. Röttgering, M. Hoeft, and F. de Gasperin, "The Planck clusters in the LOFAR sky. i. LoTSS-DR2: New detections and sample overview," *Astronomy & Astrophysics*, vol. 660, p. A78, 2022.