Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Deep learning per la risonanza magnetica

Studente/i
Simone Gasparetto

Relatore
Loris Cannelli

Correlatore
-

Committente
Loris Cannelli

Corso di laurea
Ingegneria informatica TP

Codice progetto
C10499

Anno
2021/2022

02.09.2022

Sommario

Indice delle figure

Abstract

L'obiettivo principale di questo progetto, oltre al lato didattico, è l'implementazione di un modello di deep learning per la ricostruzione e il denoising di immagini derivanti da risonanza magnetica.
I dataset a disposizione concernano immagini di cervelli e di cuori.
La parte iniziale si indirizza verso la modellizzazione e la standardizzazione dei dati, per poi proseguire con l'inserimento manuale degli errori.
Gli errori sono dati dalla variazione dei singoli pixel come conseguenza di una distribuzione normale e come risultato di rimozione di righe e singoli pixels dell'immagine.
La metodologia applicata è inerente alle reti neurali, più precisamente agli autoencoder. Ne sono stati usati diversi, contenenti layer densi o convoluzionali, da più semplici a più complicati; all'utilizzo di reti convoluzionali i risultati sono stati più promettenti e portano a lavorare in quella direzione.
Sono stati infatti trovati valori di MSE molto inferiori, 4 volte minori partendo da situazioni di rumore maggiore.
Analizzando i risultati dei test si può affermare che l'MSE è proporzionale al valore del rumore, di conseguenza quando i disturbi aumentano si necessita di reti più potenti.
Un dataset contenente immagini da 128x128 pixels varia da 10 righe rimosse con un MSE di 0.0045 fino a 20 righe rimosse con un MSE di 0.0093.
Come proseguimento di questo progetto si valuterà la possibilità di continuare a lavorare con reti neurali con l'aggiunta di qualche regolarizzazione per diminuire ulteriormente il rumore.

The main objective of this project, apart from the educational side, is the implementation of a deep learning model for the reconstruction and denoising of MRI-derived images.
The available datasets concern images of brains and hearts.
The initial part addresses the modelling and standardisation of the data, followed by the manual input of errors.
Errors are given by the variation of individual pixels because of a normal distribution and as a result of removing rows and individual pixels of the image.
The methodology applied is inherent to neural networks, more specifically autoencoders.
Different ones were used, containing dense or convolutional layers, from the simplest to the most complicated; when using convolutional networks, the results were more promising and lead to work in that direction.
In fact, much lower MSE values were found, 4 times lower starting from situations of higher noise.
Analysing the test results, it can be stated that the MSE is proportional to the value of the noise, so that when the noise increases, more powerful networks are needed.
A dataset containing images of 128x128 pixels ranges from 10 rows removed with an MSE of 0.0045 to 20 rows removed with an MSE of 0.0093.
As a continuation of this project, we will consider continuing to work with neural networks with the addition of some regularisation to further reduce noise.

Progetto assegnato

Descrizione

La risonanza magnetica è un procedimento medico attraverso il quale una grande quantità di dati viene immagazzinata ogni secondo che un paziente rimane in uno scanner.
Questo avviene sotto forma di intensità di un campo magnetico opportunamente generato e controllato.
Questa considerevole quantità di dati deve poi essere processata in maniera efficiente, al fine di ricostruire immagini e/o video a fini diagnostici.
È solo negli ultimi anni che si è passati dall'utilizzo di algoritmi iterativi per la ricostruzione delle immagini a partire dai dati grezzi, all'implementazione di tecniche di machine learning e deep learning.
Lo scopo di questo progetto è l'implementazione di un modello di deep learning per ricostruzione e denoising di immagini a partire da dati acquisiti da scanner per risonanza magnetica.
In particolare, il progetto si concentrerà sull'implementazione Python e sul test di reti convoluzionali standard, supportate, se necessario, da autoencoder per ottenere risultati più raffinati.
Un preprocessing dei dati di partenza sarà parte integrante del progetto di tesi.
Alcuni dei dataset che saranno utilizzati sono disponibili in forma pubblica, altri sono forniti da collaboratori esterni.

Compiti

- Review stato dell'arte
- Preprocessing dei dati in un formato consono per il modello di deep learning che si vuole addestrare
- Scelta architettura
- Implementazione
- Test

Obiettivi

Obiettivi del progetto:

- Analizzare la problematica di data analysis ed i tools disponibili
- Sviluppare un modello di deep learning capace di fare sia preprocessing che, in particolare, analisi dei dataset a disposizione, con lo scopo di ricostruire e fare denoising di immagini provenienti da macchinari per la risonanza magnetica.
- Redigere la documentazione di progetto

Obiettivi didattici:

- Imparare a gestire un progetto in maniera autonoma ed efficiente, col supporto del relatore
- Sviluppare la capacità di definire un problema ed i requisiti necessari alla sua soluzione
- Estendere le proprie competenze nella capacità di sperimentare e di creare soluzioni, applicando modelli di sviluppo moderni

Tecnologie

- Python (sfruttando le classiche librerie per il deep learning, il machine learning e la visualizzazione risultati: Scikit-learn, TensorFlow, PyTorch, Keras, Matplotlib, etc.)
- MATLAB

1. Strumenti ai fini della ricerca

1.1 Risonanza Magnetica

La risonanza magnetica, MRI, acronimo di Magnetic Resonance Imaging, è una tecnologia di imaging non invasiva che produce immagini anatomiche dettagliate.
Spesso viene utilizzata per il rilevamento, la diagnosi e il monitoraggio delle malattie.
Il suo funzionamento si basa sul principio che in buona parte il corpo umano è costituito da molecole d'acqua, che consistono in atomi di idrogeno e ossigeno.
Gli atomi di idrogeno contengono una particella chiamata protone, la quale è molto sensibile ai campi magnetici.
La risonanza magnetica si basa appunto su una sofisticata tecnologia che impiega potenti magneti.
Quest'ultimi producono un forte campo magnetico che costringe i protoni nel corpo ad allinearsi con quel campo.
Quando l'energia aggiuntiva (sottoforma di onde radio) viene aggiunta al campo magnetico, devia i protoni dall'allineamento.
Una volta che la sorgente di radiofrequenza viene spenta i protoni ritornano al loro stato di riposo.
Questo provoca l'emissione di segnali di tipo radio.
Il valore delle intensità dei a segnali a radiofrequenza forniscono informazioni sulla posizione esatta dei protoni nel corpo e vengono salvate al fine di processarle per ottenere immagini visibili.
Queste intensità delle onde radio vengono organizzate in matrici; la visualizzazione di questi segnali crea immagini comprensibili all'occhio umano.
Infatti, questi segnali sono nel cosiddetto dominio della frequenza.
Vi sono infatti due differenti domini: delle frequenze e delle immagini.
Tramite un procedimento matematico opportuno è però possibile creare immagini visibili.

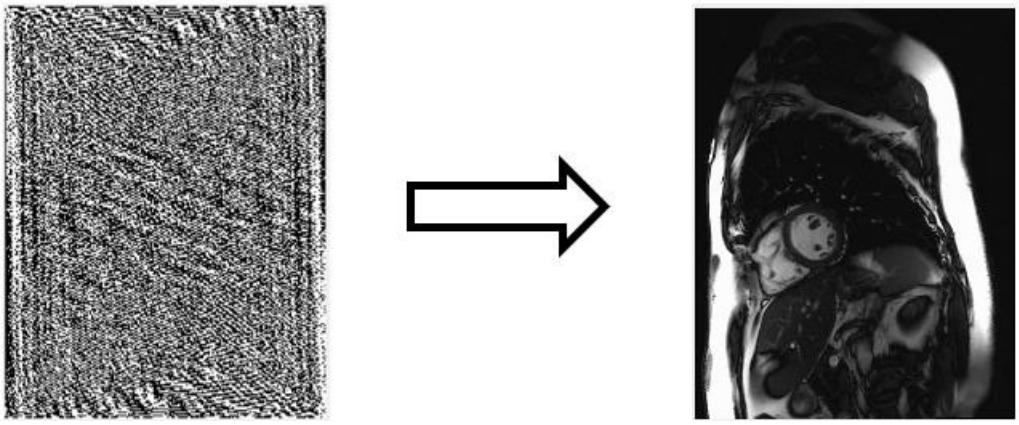


Fig. 1 Passaggio da dominio delle frequenze a dominio delle immagini

Nella figura 1 mostra entrambe le situazioni, sulla sinistra un segnalerappresentato nel cosiddetto dominio delle frequenze, mentre a destra nel dominio delle immagini.
Per ottenere le immagini visibili dell'organo interessato ricostruite a partire dal segnale nel dominio della frequenza esiste un procedimento che si basa su algoritmi iterativi. Questi algoritmi fanno uso della trasformata di Fourier e il passaggio da un dominio all'altro è reversibile.
Un problema noto dell'MRI è che le immagini ricostruite possono essere soggette a diversi tipi di disturbi: avere delle distorsioni, non essere nitide, essere sfocate, ecc. Questo può provocare diversi problemi nell'ambito, come la lettura delle immagini da parte dei dottori.
Le cause possono essere molteplici, sia in fase di acquisizione dati, sia in fase di ricostruzione dell'immagine.

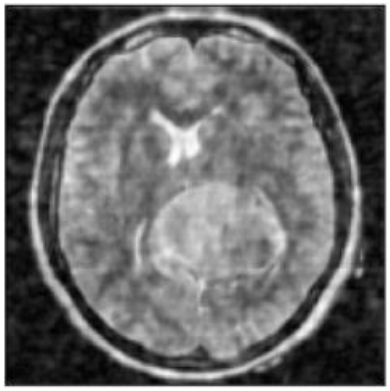


Fig. 3 Risonanza magnetica di un cervello contenente disturbi

Fig. 2 Risonanza magnetica di un cuore contenente disturbi

Le immagini 2 e 3 mostrano esempi di alcuni problemi tipici nel MRI.
Questi comportano perdita di dettagli dell'immagine e lo scopo di questo lavoro è di trovare una soluzione per alleviarli.

1.2 Reti neurali ed Autoencoder

Le reti neurali sono ispirate alla biologia umana, infatti la loro struttura composta da neuroni tende a imitare il funzionamento del cervello umano.
Le reti sono composte da diversi nodi, chiamati neuroni, che prendono un input di dati lo fa passare per i neuroni, i quali sono collegati tra loro e possono svolgere diverse operazioni sui dati, e infine restituiscono un output.
Gli autoencoder sono una particolare architettura di reti neurali, sviluppata per alcuni obiettivi specifici.
Lo scopo degli autoencoder è di imparare a codificare qualcosa automaticamente e di essere in grado di imparare a scomporre i dati e in seguito ricostruirli più fedelmente possibile all'originale.
L'Autoencoder è formato da due parti: Encoder e Decoder.

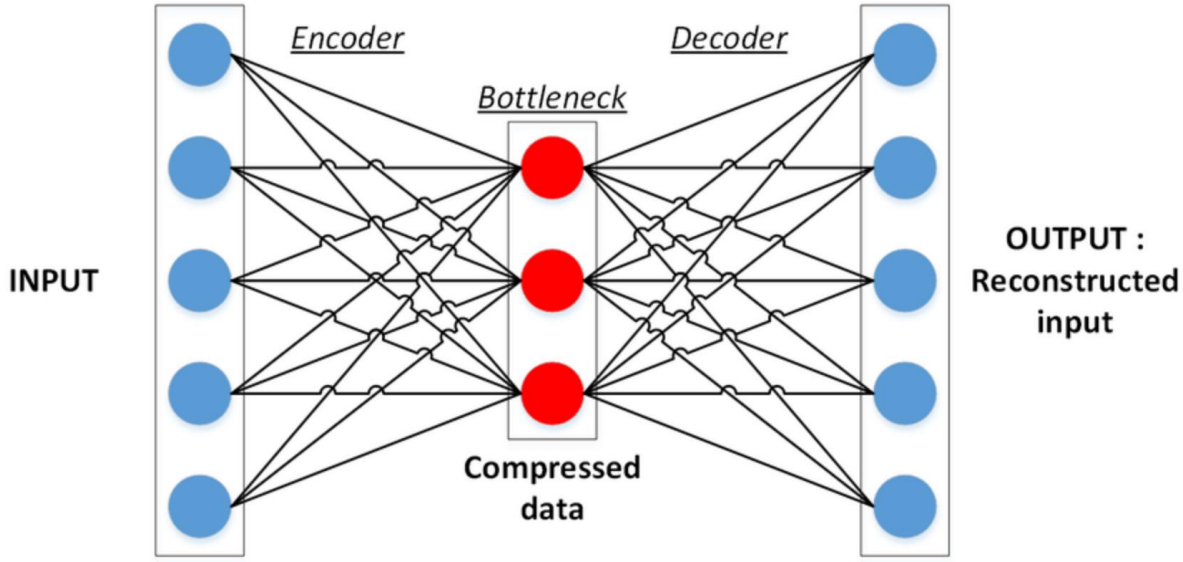


Fig. 4 Pipeline dell'autoencoder

L'encoder, o codificatore, ha il compito di trovare la più piccola rappresentazione possibile dei dati che può memorizzare, estraendo le caratteristiche più importanti dei dati originali.
Il codificatore prende i dati in input e ne genera una versione codificata: i dati compressi.
I quali verranno in seguito passati al decoder.
Il decoder, o decodificatore, funziona in modo simile all'encoder, ma inversamente.
Impara a leggere queste rappresentazioni di codice compresso e generare immagini basate su tali informazioni.
Durante la ricostruzione punterà a minimizzare la perdita.
L'output, ossia l'immagine ricostruita, viene confrontata con quella originale utilizzando un errore (MSE).

In base alla differenza delle due immagini sia il codificatore che il decodificatore vengono valutati e aggiornano i loro parametri per migliorarli.

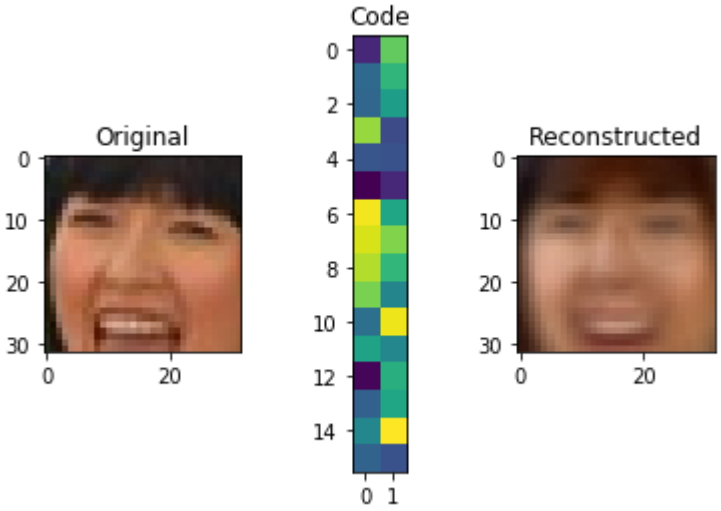


Fig. 5 Autoencoder con poche informazioni di codifica

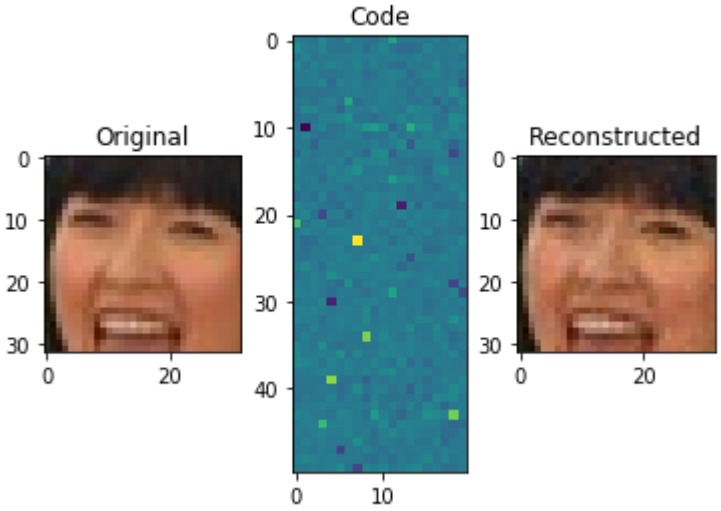


Fig. 6 Autoencoder con molte informazioni di codifica

Sono qua mostrate due situazioni differenti dell'Autoencoder nelle figure 5 e 6.
La parte centrale rappresenta la parte codificata, ossia la quantità di informazioni che vengono ritenute importanti da salvare.
Nel primo caso, essendo le informazioni poche, l'immagine viene ricostruita con pochi dettagli e risulta molto differente da quella originale.
La seconda invece, aumentando le informazioni in uscita dalla codifica, si può osservare un'immagine ricostruita molto più simile a quella originale.

1.3 Revisione dei lavori precedenti

Le tecniche per la ricostruzione di immagini da risonanza magnetica esistono da decenni.
La maggior parte degli approcci standard utilizzano modelli di machine learning come least squares regolarizzati, PCA, kernel methods o manifolds (si può far riferimento a [1] per maggiori dettagli sugli approci appena citati).
Questi modelli vengono prevalentemente gestiti con algoritmi iterativi piuttosto complicati.
In questo progetto di tesi invece, è stato deciso di utilizzare tecniche di deep learning (ossia con reti neurali) e in particolare è stato sperimentato il funzionamento del denoising tramite autoencoders.
Il deep learning nel campo del MRI è un qualcosa di recente, ma si molti lavori ed approcci di vario tipo sono stati proposti negli ultimi anni.
Il deep learning può venire implementato in diversi step della pipeline che parte dall'acquisizione dati fino alla ricostruzione di immagini con una certa qualità; una buona overview a riguardo è presente in [2].
Questo progetto di tesi si concentra su denoising per MRI con tecniche di deep learning, un ambito in cui non molto è stato svolto finora. Il lavoro più affine a quello contenuto in questa tesi è [3] nel quale, però, si è svolto denoising solo su due tipologie di rumore, non molto interessanti dal punto di vista pratico.

2. Implementazione

2.1 Dataset utilizzati

L'iniziale intento di questo progetto era quello di lavorare con un dataset di cuori, ossia una serie di immagini ottenute dalla risonanza magnetica.
Quest'ultime, infatti hanno una fase di elaborazione più lunga del previsto, si è inserito un ulteriore dataset composto solamente da immagini di cervelli.

I due dataset sono composti in modi differenti: per dimensioni e quantità.

Il dataset delle risonanze magnetiche dei cervelli si presenta da subito con immagini JPG, pronte per essere modellate.
Il dataset viene suddiviso a priori tra immagini di training e immagini di testing.
Entrambe contengono una suddivisione in cervelli sani e malati.
Più precisamente vi sono immagini con tre tipi diversi di tumore, tra cui: glioma, meningioma e pituitario.

I campioni sono un totale di 2870 per la parte di training e 425 per la parte di testing.

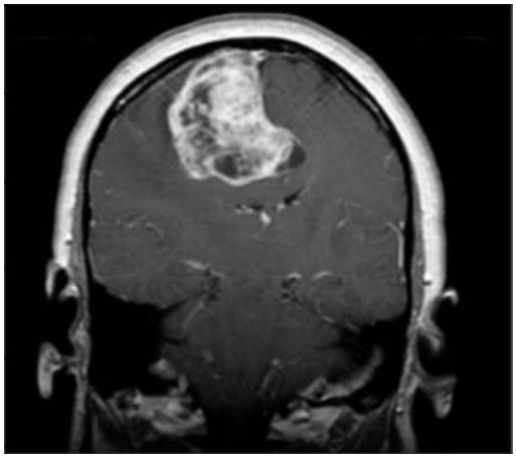


Fig. 8 Risonanza magnetica di un cervello con un tumore gliale visto posteriormente

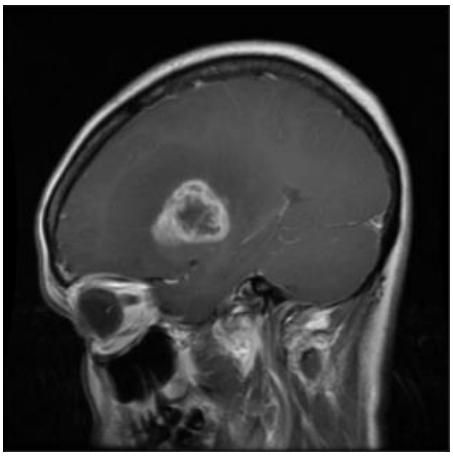


Fig. 7 Risonanza magnetica di un cervello con un tumore meningioma visto lateralmente

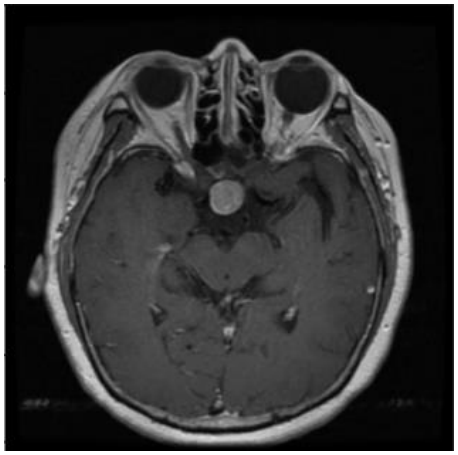


Fig. 10 Risonanza magnetica di un cervello con un tumore pituitario visto dall'alto

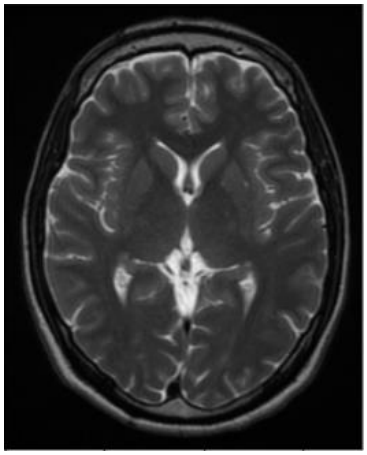


Fig. 9 Risonanza magnetica di un cervello senza tumori visto dall'alto

Un'altra caratteristica di questo dataset è la differenziazione tra immagini.
Sono presenti immagini da diverse angolazioni: lateralmente, posteriormente e al di sopra del cranio.
Come detto in precedenza, si può osservare dalla dimensione delle immagini che quest'ultime posseggono altezze e larghezze differenti tra loro.
Questo è un problema per la rete neurale, poiché dovranno essere modellate per risultare conformi.

Per il dataset 2 la situazione è stata differente. Non partiamo più direttamente con immagini JPG pronte per essere modellate, bensì con un tensore di dimensione 256 * 200 * 256, ossia in breve 256 immagini aventi un altezza di 256 pixels e una larghezza di

200 pixels. È servita quindi una procedura per trasformare il tensore in immagini visibili. I due vantaggi rispetto all'altro dataset sono: avere immagini della stessa dimensione e averle dalla stessa angolatura.

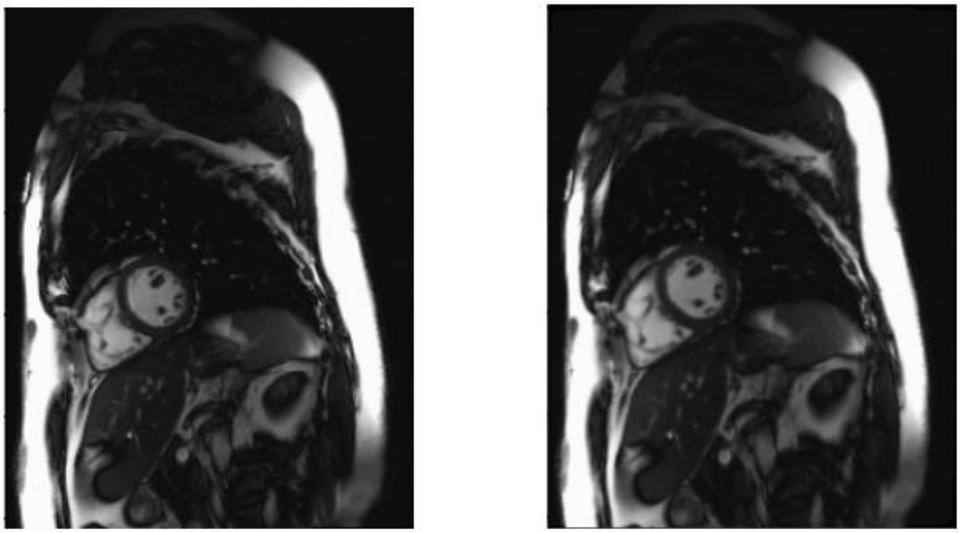


Fig. 11 Risonanza magnetica di un cuore

Le immagini del dataset 2 sono, contrariamente a quelle del dataset 1, più vaste.
Esse infatti raffigurano anche gli organi circostanti. Sono state effettuate sullo stesso individuo in momenti diversi.
Questo può portare le immagini ad assomigliarsi tra loro, ma contengono comunque momenti di compressione e decompressione differenti del cuore.

Per lavorare con le reti neurali, in generale con l'intelligenza artificiale, è importante la preparazione dei dati prima di poterli utilizzare.
Per questo motivo in entrambi i dataset viene svolto del lavoro di modellizzazione per omologarli alla rete.

2.2 Modellazione dati

In questo lavoro, dopo avere ben in chiaro quali sono i dataset a disposizione e quali sono le loro caratteristiche, si è proseguito con una fase di pre-processing.
Nella situazione attuale, avendo due dataset così differenti tra loro a livello di caratteristiche, si è deciso di trattarli in modo diverso.

Nel primo dei due casi, ossia il dataset dei cervelli (dataset 1), sono già presenti dall'inizio immagini nella forma corretta.
Le immagini, però, soffrono di un problema relativo a dimensioni diverse di altezza e larghezza.

In questi casi non esiste un modo di procedere che sia ottimale, bisogna valutare le varie possibilità in base a quella situazione.

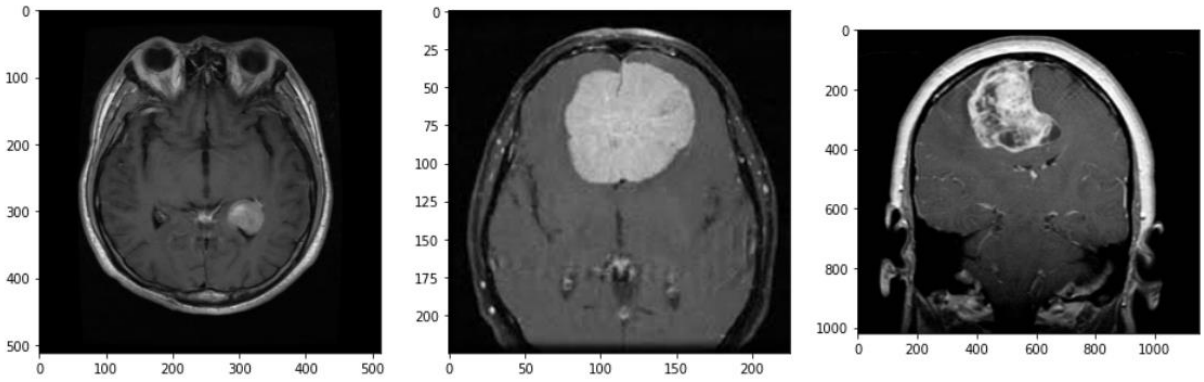


Fig. 12 Dataset di cervelli con risoluzioni differenti

La circostanza attuale non è favorevole siccome le immagini non solo non hanno le stesse altezze e larghezza, ma bensì nemmeno lo stesso rapporto tra esse.

Data la grande presenza di immagini 512 * 512 la prima soluzione presa in considerazione è stata quella di tenere unicamente quelle immagini, eliminando le altre, e di valutare se vi fossero abbastanza dati per testare in seguito.
Il risultato non è stato positivo per via della poca omogeneità tra i vari dati.
Il problema che è stato riscontrato è di aver conservato numerose immagini nei dati inerenti ai tumori, mentre una scarsità in quelle senza.

Scartata questa opzione si è proseguito valutando un ridimensionamento omogeneo del dataset.
Come detto in precedenza, vista la grande quantità di immagini 512 * 512, conviene ridimensionare le altre immagini con tale risoluzione.

Una volta ridimensionate tutte le immagini si ottiene un dataset uniforme, a questo si possono passare alla rete che sarà in grado di allenarsi.
Dato che i training preliminari, che analizzeremo meglio in seguito, si sono rivelati essere molto lunghi, si è deciso di procedere a un rescaling delle immagini dell'intero dataset per ridurre il numero di pixel in input al modello neurale.
Si possono fare diversi test per capire dopo quanto l'immagine è ancora ben riconoscibile.
In questo caso, diminuendo le dimensioni delle immagini a 128 * 128, che saranno poi quelle utilizzate, si ottengono delle immagini chiare con ¼ dei pixel totali.
Ciò comporta una riduzione del tempo, utilizzato dalla rete per allenarsi con i dati, non indifferente.

Il discorso è diverso nel momento in cui si parla del dataset dei cuori. In quest'ultimo i dati non sono presenti sottoforma di immagini, ma sottoforma di tensore. La prima cosa da fare è, dunque, portare i dati in una forma in cui la modellazione è facilitata. Per fare ciò si è utilizzato il software MATLAB. Il passaggio da tensore a immagini consiste nello scorrere quest'ultimo e ad ogni matrice di pixel 256 * 200 salvare l'immagine rispettiva.

Il vantaggio di questo dataset è che, una volta salvate, le immagini sono già omogenee tra loro avendo le stesse dimensione.
Di conseguenza esse risulteranno compatibili con la rete.

2.3 Divisione dataset

Nell'ambito delle reti neurali, arrivati al punto di avere un dataset uniforme, esiste una suddivisione che deve essere fatta ai fini del suo funzionamento.
L'allenamento di un modello di rete neurale (quale l'autoencoder che andremo ad utilizzare) è solitamente suddiviso in tre fasi: training, validation e test.
La parte dei dati dedicata al training, ovvero la più vasta e la principale, è quella che si occupa dell'allenamento vero e proprio della rete.
Prende in input i dati e li processa uno per uno, in questo caso immagine per immagine.
Questo procedimento solitamente viene eseguito più volte per permettere alla rete di imparare meglio a riconoscere le immagini.

La seconda parte dei dati, dedicata alla validation, si occupa di valutare che la rete si stia allenando correttamente.
Uno dei compiti principali di questa fase è quello di verificare che la rete non commetta overfitting.
Quando una rete si sta allenando il parametro principale di giudizio per verificare se sta imparando e sta migliorando è il valore della loss function, ossia l'errore compiuto nel riconoscere i dati di validazione.
Minore è il valore migliore è, in genere, il funzionamento della rete.
Tornando al termine overfitting, questa condizione avviene nel momento in cui la rete, vedendo sempre le stesse immagini, comincia ad impararle a memoria, ossia a creare delle associazioni.
Questo viene fatto con il solo scopo di diminuire l'errore ed è controproducente poiché, se la rete fosse sottoposta ad un'immagine nuova, non saprebbe come comportarsi.
Il modo principale di alleviare il fenomeno di overfitting consiste nello sfruttare il dataset di validazione.

Quest'ultimi non sono presenti tra le immagini che la rete usa per allenarsi. Queste figure inedite permettono, tramite una loss calcolata su quest'ultime, di verificare se sta avvenendo overfitting o meno.

L'ultima parte dei dati, dedicata al test, ha il compito di testare la rete.
È in questo momento dove viene valutato se la rete ha imparato correttamente a saper gestire le immagini.
Vengono mostrate alla rete diverse figure che non ha mai visto e con quello che ha appreso deve sapere come comportarsi nelle diverse situazioni.

Una parte importante della divisione del dataset è la quantità di dati da attribuire ad ogni settore.
Nella situazione presente ci sono due casi: uno dove abbiamo già una divisione tra training e test, dove bisognerà riservare la parte per il validation.
Il secondo caso avviene nel momento in cui abbiamo un dataset intero da suddividere in tre parti.

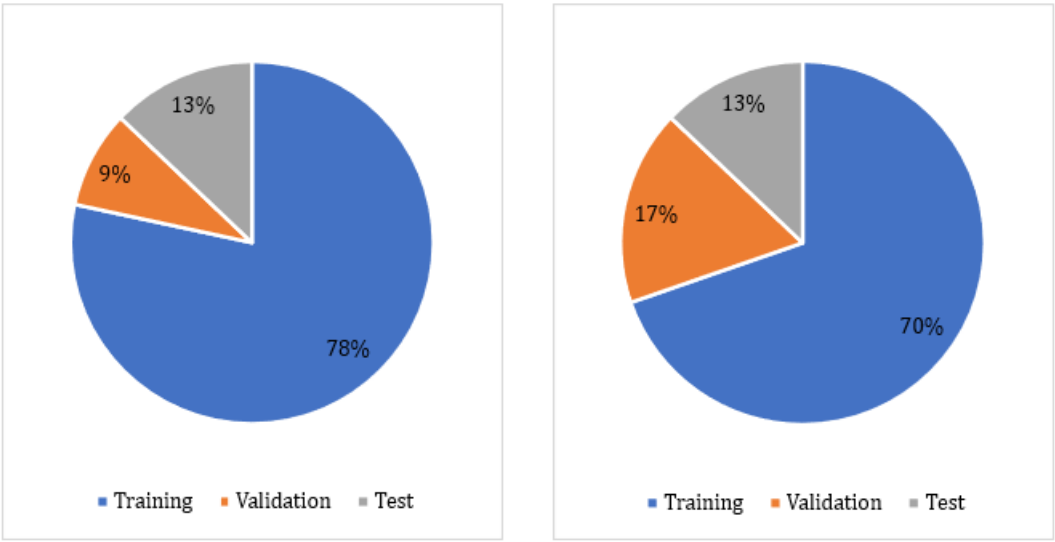


Fig. 13 Grafici rappresentanti la divisione del dataset dei cervelli

Per quanto riguarda il dataset dei cervelli è stato diviso in due modi diversi.
Il primo tentativo è stato usato il 10% di validation e il 90% di training, senza contare la parte di test, mentre nel secondo il 20% di validation e l'80% di training.

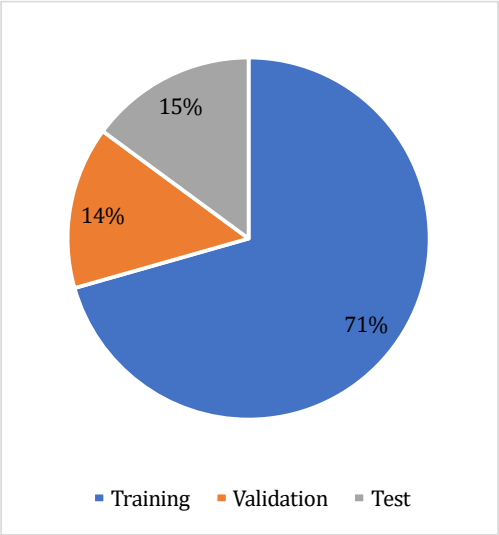


Fig. 14 Grafici rappresentanti la divisione del dataset dei cuori

Il dataset dei cuori invece viene suddiviso circa 70% di training, 15% di validation e 15% di test.

2.4 Generazione rumore

Il fulcro di questo lavoro è di riuscire a togliere da immagini rumorose la maggior parte dei disturbi ed ottenere un risultato più comprensibile possibile.
Dato che le immagini dei dataset originali sono tutte prive di rumore bisognerà inserire del rumore manualmente al fine di poter testare la rete.

Ci sono diversi tipi di rumore che si possono trovare comunemente in un'immagine proveniente da risonanza magnetica, di seguito verranno mostrati i tre principali utilizzati in questo progetto.

2.4.1 Rumore con distribuzione normale

La prima tipologia di rumore testata è data da una distribuzione normale (o gaussiana). Specificatamente, l'aggiunta di questo disturbo consiste nell'alterare di una quantità a scelta dal programmatore il valore di ogni singolo pixel.

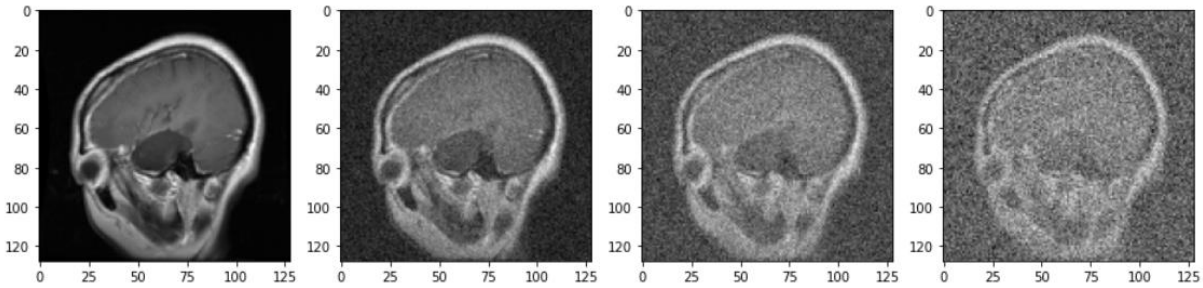


Fig. 15 Diversi livelli di rumore applicati ad un'immagine di un cervello ottenuta da risonanza magnetica

All'interno della funzione che genera questa distribuzione vi è un parametro che determina la deviazione standard, ossia la quantità random di alterazione di intensità che può subire ogni pixel. Maggiore è questo valore maggiore l'immagine verrà distorta.

Nei test effettuati con questo rumore il massimo valore raggiunto di deviazione standard è stato 0.2. Oltre a quel valore le immagini diventavano incomprensibili e una correzione dei disturbi diventava difficoltosa.

L'applicazione di questo rumore, differentemente dagli altri due che discuteremo in seguito, viene effettuata all'interno del dominio delle immagini. La funzione di distribuzione normale viene infatti applicata direttamente sull'immagine mostrata a figura 15.

Le sperimentazioni in questo caso sono state svolte unicamente sul dataset dei cervelli. Questa decisione è stata presa conseguentemente ai risultati trovati, che verranno esposti nei capitoli successivi.

2.4.2 Rumore con rimozione righe

Il rumore in questione punta ad essere il più simile alle situazioni presenti nella realtà, succede infatti che in una risonanza magnetica vengano perse delle righe nel dominio della frequenza a causa, ad esempio, di spostamenti compiuti dal paziente all'interno dello scanner.

Differentemente dalla situazione precedente, quindi, il disturbo che si viene a creare o che andremo a generare noi manualmente non viene effettuato direttamente sull'immagine presente nel dominio delle immagini.

Per applicare questo rumore dovremo quindi effettuare un passaggio intermedio, anche questa volta grazie all'utilizzo di MATLAB.

Esistono due funzioni per passare tra i due domini che sono rispettivamente: fft2 e ifft2. Il procedimento sarà quindi di spostarsi nel dominio delle frequenze tramite la funzione fft2 ed in seguito andare a eliminare il quantitativo di righe desiderato. In seguito tramite la funzione ifft2 si torna nel dominio delle immagini e si può osservare i disturbi applicati.

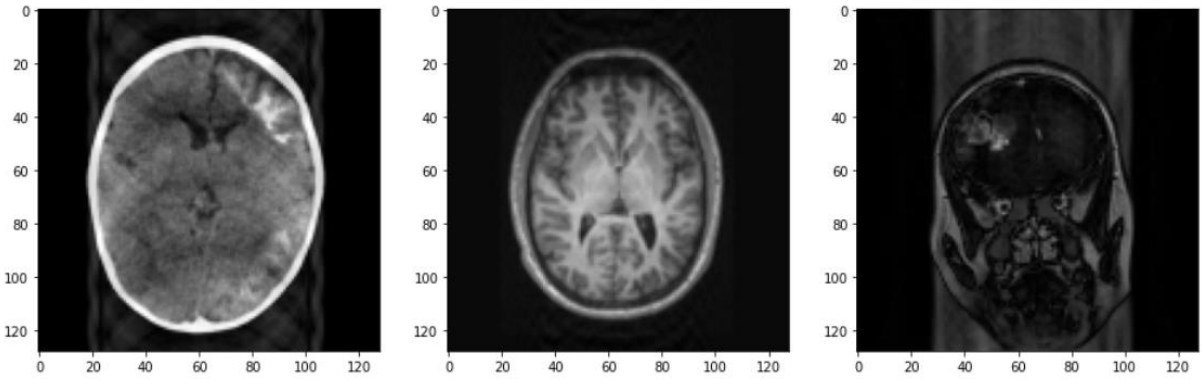


Fig. 16 10 righe rimosse da un'immagine di un cervello ottenuta da risonanza magnetica

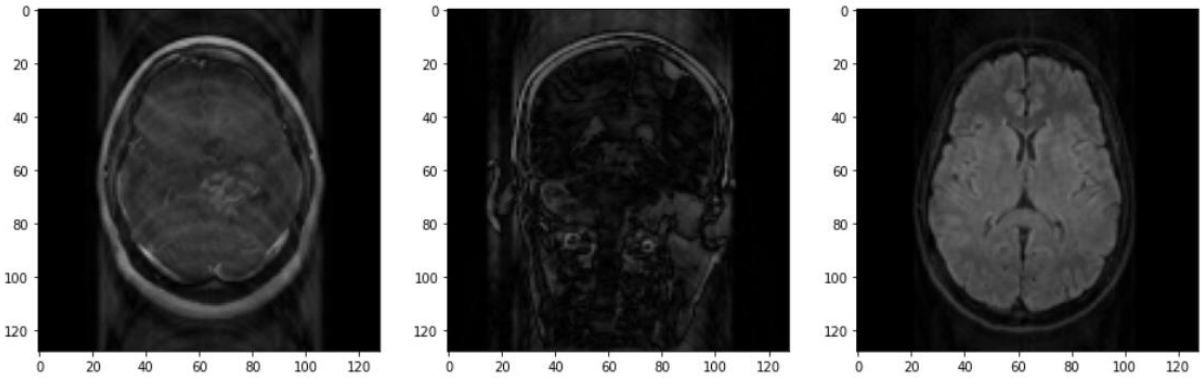


Fig. 17 20 righe rimosse da un'immagine di un cervello ottenuta da risonanza magnetica

La decisione per entrambi i dataset è stata di rimuovere 10 e 20 righe da quelle totali in modo randomico. Questo porta ad avere immagini eterogenee tra loro, poiché la rimozione di una riga piuttosto che un'altra cambia l'esito del rumore. Vi sono appunto immagini molto disturbate con solo 10 righe eliminate, mentre altre molto simili alle originali con 20 righe eliminate.
Effettuando alcuni test di prova sulla rimozione delle righe, svolta manualmente, si è potuto notare come le informazioni più importanti risiedono agli apici dell'immagine. Rimuovendo le prime e le ultime righe vi era molto più rumore rispetto a quelle al centro, che lasciavano l'immagine con la maggior parte dei dettagli.

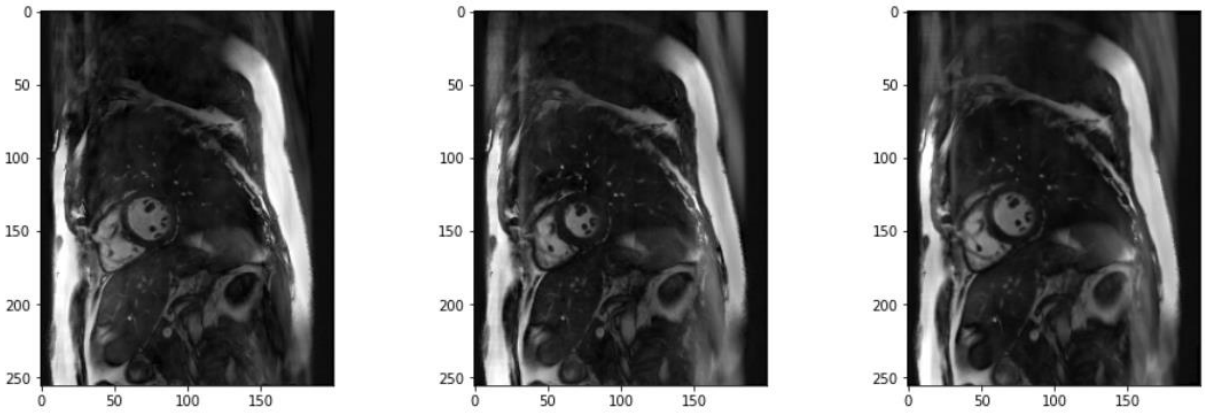


Fig. 18 20 righe rimosse da un'immagine di un cuore ottenuta da risonanza magnetica

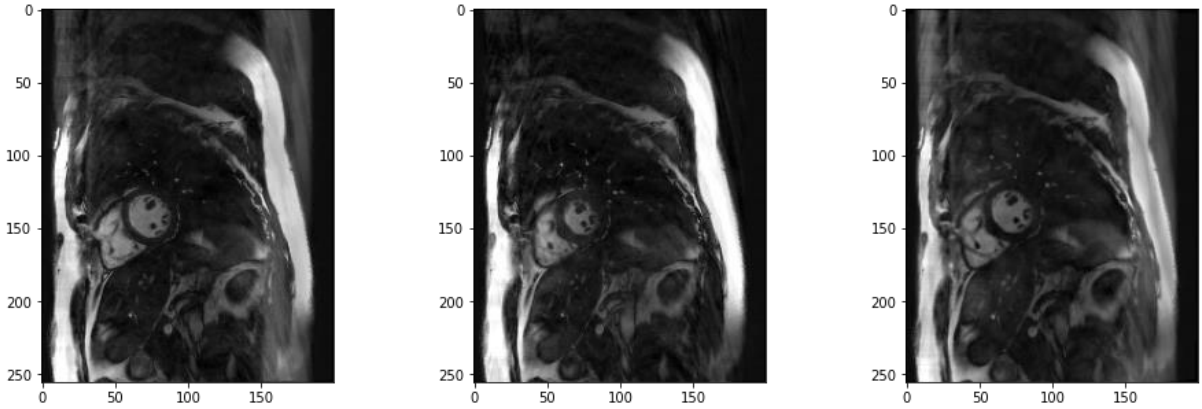


Fig. 19 40 righe rimosse da un'immagine di un cuore ottenuta da risonanza magnetica

Nonostante dalle immagini possa sembrare che la differenza tra 20 e 40 righe non sia molta non è così. Questo perché la probabilità che vengano sottratte delle righe nelle posizioni più influenti è maggiore, questo porta non solo ad avere immagini più rumorose, ma anche una quantità maggiore.

2.4.3 Rumore con rimozione pixels

Anche quest'ultima tipologia di rumore che andremo a valutare è legata ad imprecisioni durante la fase di acquisizione dati, quando il paziente si trova nello scanner. La differenza con il rumore generato nella sezione precedente è che quando ci si trova nel dominio delle frequenze l'operazione da eseguire non è la rimozione di un'intera riga, bensì di alcuni pixel.

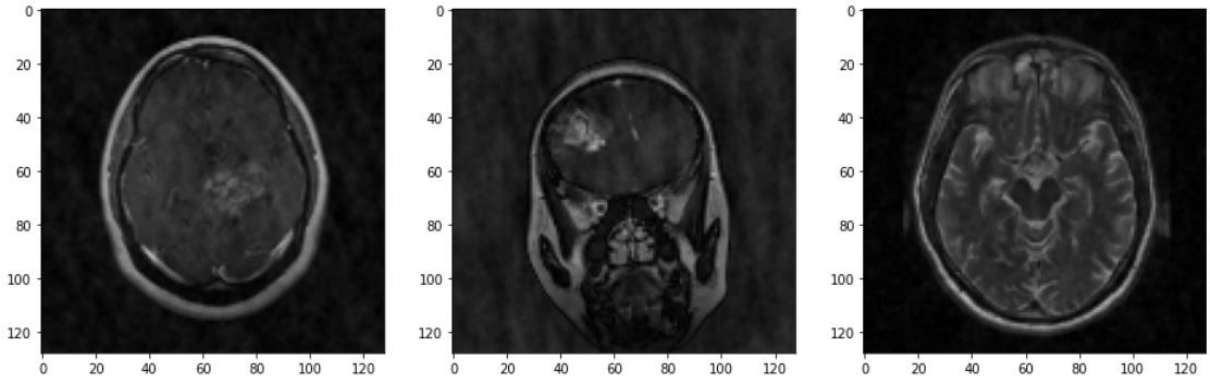


Fig. 20 1 pixel rimosso ogni 16 da un'immagine di un cervello ottenuta da risonanza magnetica

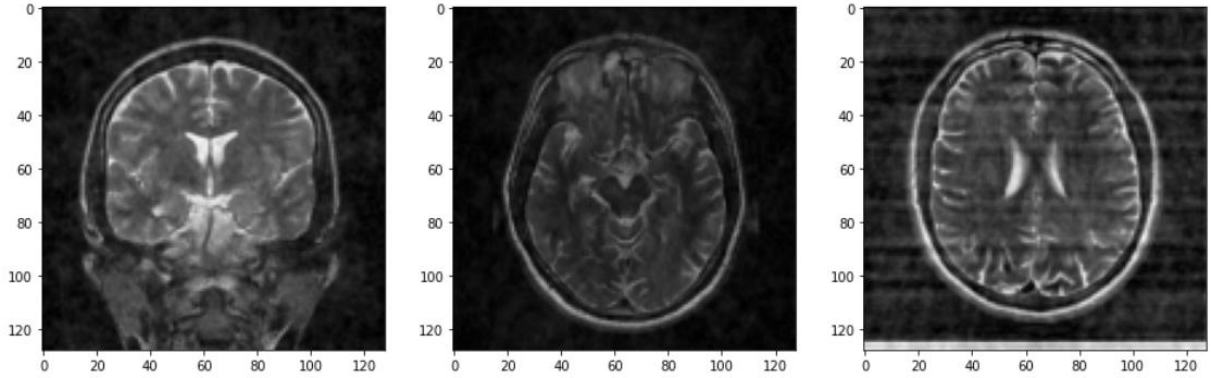


Fig. 21 1 pixel rimosso ogni 8 da un'immagine di un cervello ottenuta da risonanza magnetica

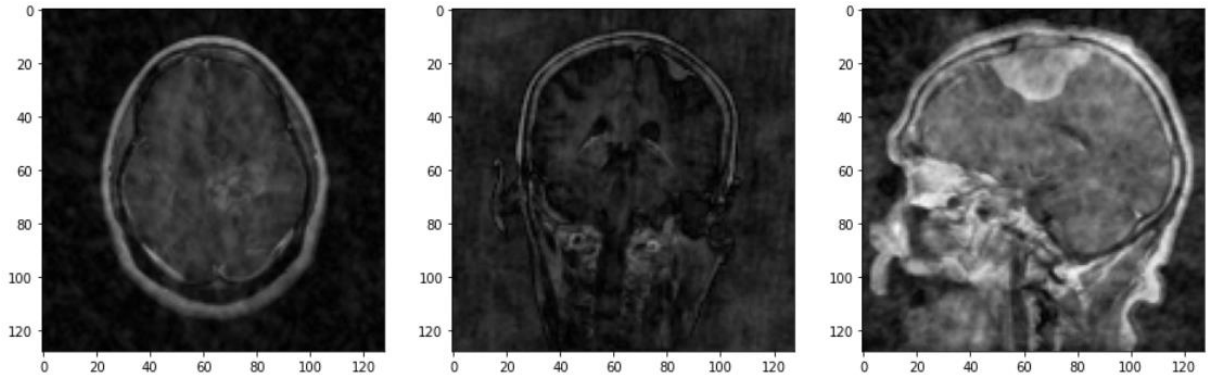


Fig. 22 1 pixel rimosso ogni 4 da un'immagine di un cervello ottenuta da risonanza magnetica

Per l'applicazione di questo rumore la prima decisione che è stata presa è la frequenza di campionamento con cui vengono eliminati i pixel, in questo lavoro abbiamo deciso di vedere gli effetti togliendo 1 pixel ogni 16, ogni 8 e ogni 4. L'eliminazione dei pixel è a sua volta randomica, ossia non sempre un pixel viene eliminato ogni frequenza decisa. Nel caso, ad esempio, di un pixel rimosso ogni 4 vorrà dire che nei primi 4 ci sarà un pixel rimosso, nei seguenti 4 un altro sempre in posizione casuale e così di seguito per tutta l'immagine.

Anche in questo caso il rumore presente in una determinata immagine sarà randomico, nel complesso, ovviamente, aumentando i pixel rimossi il disturbo sarà maggiore.

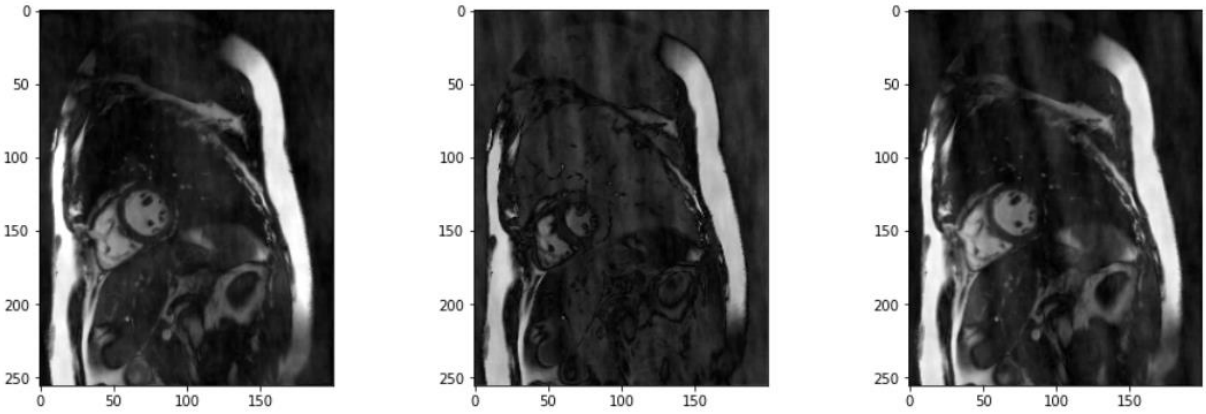


Fig. 23 1 pixel rimosso ogni 20 da un'immagine di un cuore ottenuta da risonanza magnetica

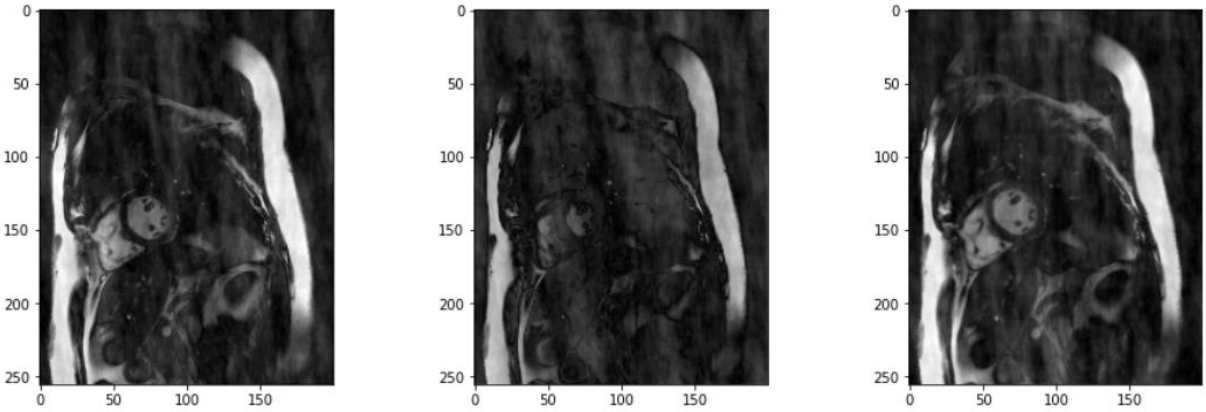


Fig. 24 1 pixel rimosso ogni 8 da un'immagine di un cuore ottenuta da risonanza magnetica

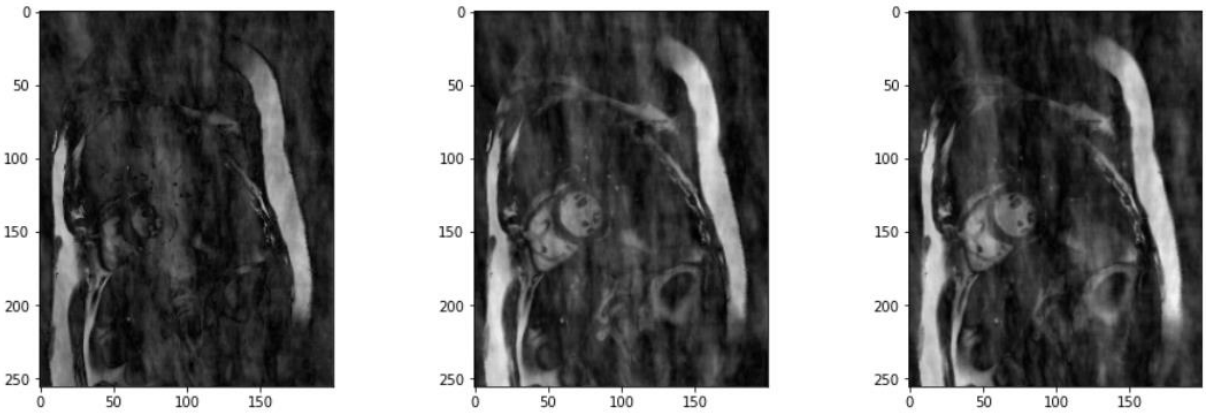


Fig. 25 1 pixel rimosso ogni 4 da un'immagine di un cuore ottenuta da risonanza magnetica

Nel dataset dei cuori c'è una differenza sul numero di pixel tolti data dalle dimensioni delle immagini, per facilitare la rimozione di pixel in un dataset è stato rimosso un pixel ogni 16 mentre in questo ogni 20.

2.5 Creazione reti neurali

Una buona riuscita di questo progetto è data dalla composizione delle reti neurali, una rete piuttosto che un'altra può avere risultati completamente differenti.
In questo lavoro sono state usate principalmente due reti neurali, più specificamente due autoencoders. Queste due, in seguito, sono state modificate leggermente.

2.5.1 Autoencoder vanilla

Si è deciso di fare i primi test con un autoencoder molto semplice, per vedere se già con una struttura poco profonda si potessero ottenere risultati lievemente positivi.
All'interno di questo autoencoder vengono utilizzati solamente layer sequenziali di tipo denso.

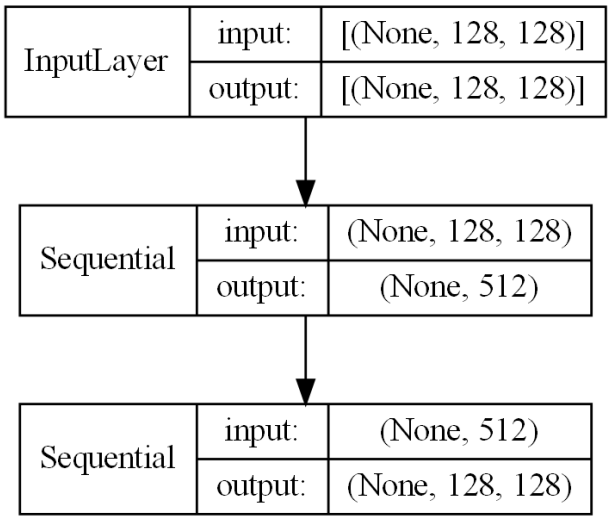


Fig. 26 Layer contenuti all'interno dell'autoencoder vanilla

Osservando la figura 26 si può vedere che, come ultimo strato, è presente un layer con una densità di 512 neuroni. Nei primi esperimenti si era provato ad avere un output molto leggero con soli 128 neuroni, ma non era sufficiente per gestire le immagini dei dataset a nostra disposizione.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 128, 128)]        0

 sequential_2 (Sequential)   (None, 512)               8389120

 sequential_3 (Sequential)   (None, 128, 128)          8404992

=================================================================
Total params: 16,794,112
Trainable params: 16,794,112
Non-trainable params: 0
_________________________________________________________________
```

Fig. 27 Sommario dell'autoencoder vanilla

La figura 25 mostra un riassunto schematico dell'architettura del nostro autoencoder di partenza.
In questo caso tutti i parametri presenti possono essere allenati.
Nonostante sia la prima rete, e la più semplice, la quantità di parametri che vengono ogni volta cambiati durante l'allenamento sono moltissimi.

2.5.2 Autoencoder convoluzionale

La seconda rete è quella più complessa, ossia di tipo convoluzionale.
Contiene infatti diversi layer in più. Il motivo del passaggio da un autoencoder ad un altro è per le questioni di potenza della rete.

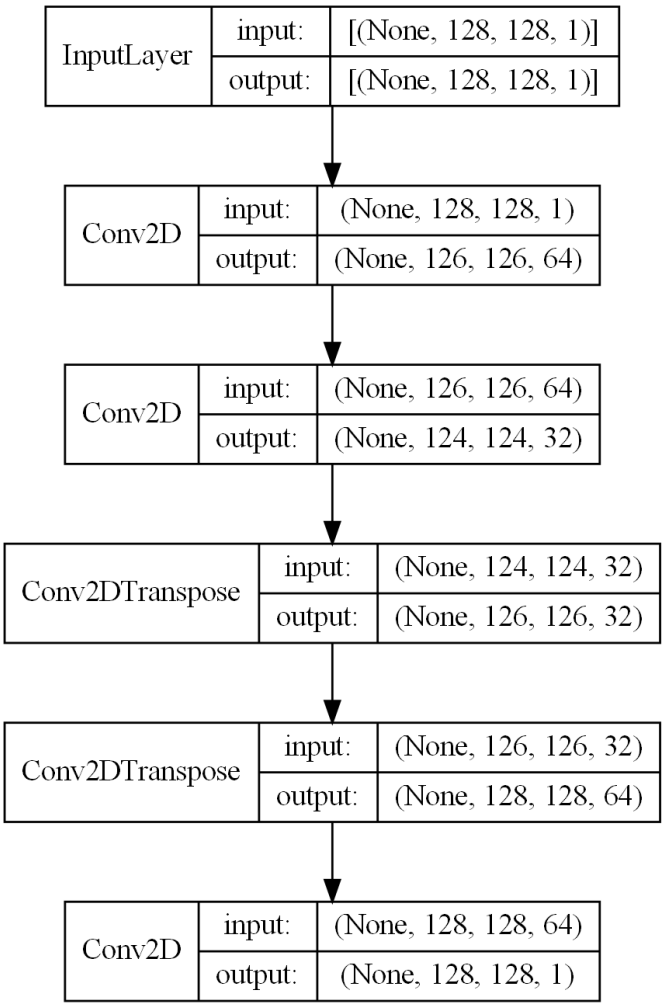


Fig. 28 Layer contenuti all'interno dell'autoencoder convoluzionale

La seconda rete si presenta in questo modo, salta subito all'occhio la differenza di complessità tra le due.
Questa contiene diversi layer in più dell'altra.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 126, 126, 64)      640

 conv2d_1 (Conv2D)            (None, 124, 124, 32)      18464

 conv2d_transpose (Conv2DTra  (None, 126, 126, 32)      9248
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 128, 128, 64)      18496
 ranspose)

 conv2d_2 (Conv2D)            (None, 128, 128, 1)       577

=================================================================
Total params: 47,425
Trainable params: 47,425
Non-trainable params: 0
_________________________________________________________________
```

Fig. 29 Sommario dell'autoencoder convoluzionale

Anche in questo caso sono mostrati i diversi parametri presenti in ogni singolo layer.

Il valore maggiore di parametri nella prima rete rispetto alla seconda può ingannare. Nel primo autoencoder è così alto perché è stato inserito un layer centrale di 512 e quindi il numero viene alzato da questo.

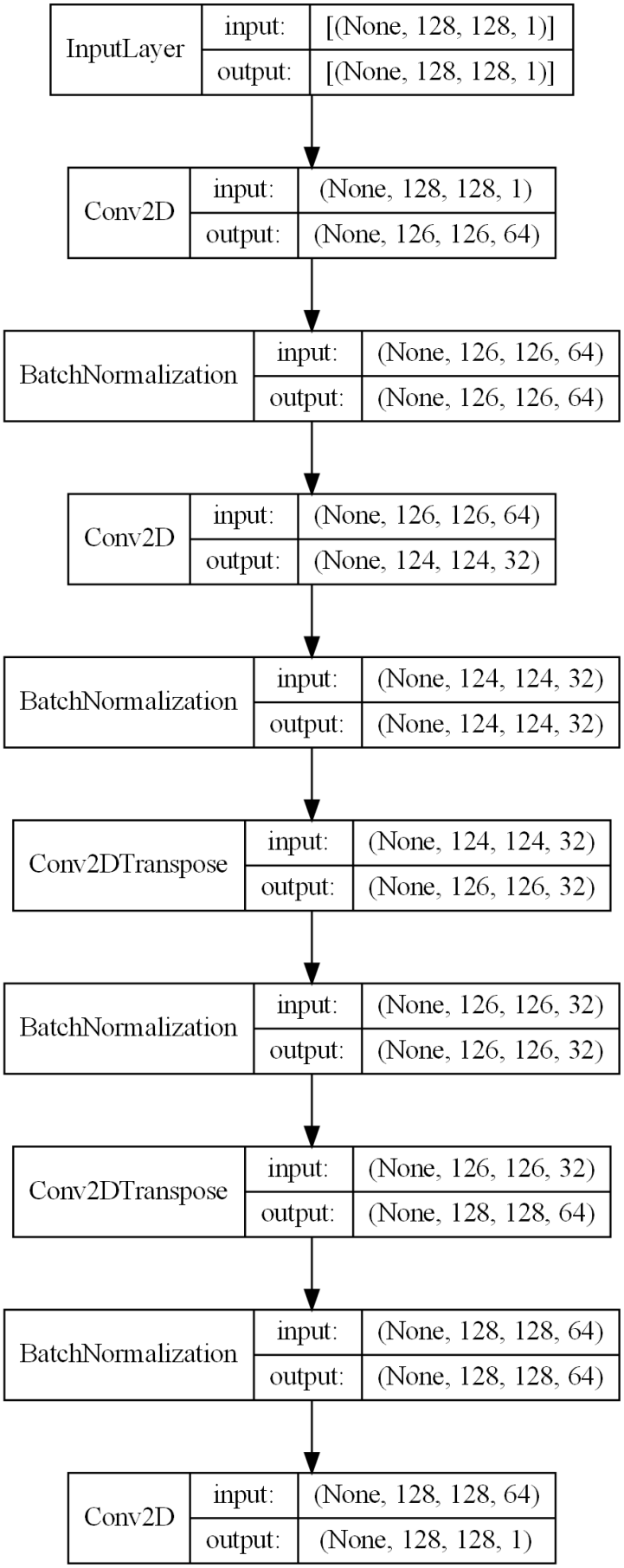


Fig. 30 Autoencoder convoluzionale con layer aggiuntivi

Questa è la rete precedente ma con un'aggiunta, i layer di BatchNormalization.
È stata effettuata questa modifica per delle necessita presentate in seguito nei test, i layer aggiunti si occupano di normalizzare i valori prima di passarli al layer successivo.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 126, 126, 64)      640

 batch_normalization (BatchN  (None, 126, 126, 64)     256
 ormalization)

 conv2d_4 (Conv2D)           (None, 124, 124, 32)      18464

 batch_normalization_1 (Batc  (None, 124, 124, 32)     128
 hNormalization)

 conv2d_transpose_2 (Conv2DT  (None, 126, 126, 32)     9248
 ranspose)

 batch_normalization_2 (Batc  (None, 126, 126, 32)     128
 hNormalization)

 conv2d_transpose_3 (Conv2DT  (None, 128, 128, 64)     18496
 ranspose)

 batch_normalization_3 (Batc  (None, 128, 128, 64)     256
 hNormalization)

 conv2d_5 (Conv2D)           (None, 128, 128, 1)       577

=================================================================
Total params: 48,193
Trainable params: 47,809
Non-trainable params: 384
_________________________________________________________________
```

Fig. 31 Sommario autoencoder convoluzionale con layer aggiuntivi

Esaminando il riassunto della rete è visibile come anche i layer di Batch contengono dei parametri e come a differenza degli autoencoder precedenti questo abbia parametri non allenabili.

2.6 Allenamento reti neurali

Quando si allena una rete neurale bisogna valutare diversi parametri in base anche quale si sceglie.
Nel caso di questo progetto avendo due reti distinte verranno utilizzati due approcci differenti.
La prima cosa è fornire all'autoencoder i dati necessari, in questo caso sia le immagini rumorose che quelle originali.
Questo viene fatto per mostrare alla rete come dovrebbero diventare le immagini in base a quelle rumorose che gli vengono fornite.
Vengono inoltre passati anche i dati di validation per il controllo dell'overfitting.

Un'altra decisione importante da prendere è il numero di epochs.
Questo valore rappresenta le volte che la rete si allenerà con i dati a disposizione.
Un'epoca è infatti un'iterazione sull'intero dataset di training.

Un'aggiunta che è stata fatta all'interno di questo allenamento sono le callbacks.
Sono delle funzioni che vengono eseguite a determinate condizioni.
Le due principali utilizzate in questo progetto sono state: una che permettesse ad ogni epoch di salvare il modello, per avere dei salvataggi intermedi.
La seconda invece con il compito di monitorare la loss della validation e stoppare il modello qualora aumentasse.
Quest'ultima funzione doveva inoltre occuparsi, una volta fermato l'allenamento, di salvare il modello nel momento in cui vi erano i risultati migliori, ossia una loss minore.

3. Procedura di test e risultati

3.1 Risultati test senza rumore

La prima rete neurale testata in questo progetto è stata quella più semplice, ossia contenente unicamente layer densi.
È stata allenata tramite i dati dei cervelli, in questo caso, per iniziare a capire le potenzialità della rete, senza alcun rumore.
L'allenamento ha impiegato relativamente poco, occupando un tempo di circa 10 secondi a epoch, questo ha permesso di poter eseguire 100 epochs totali.

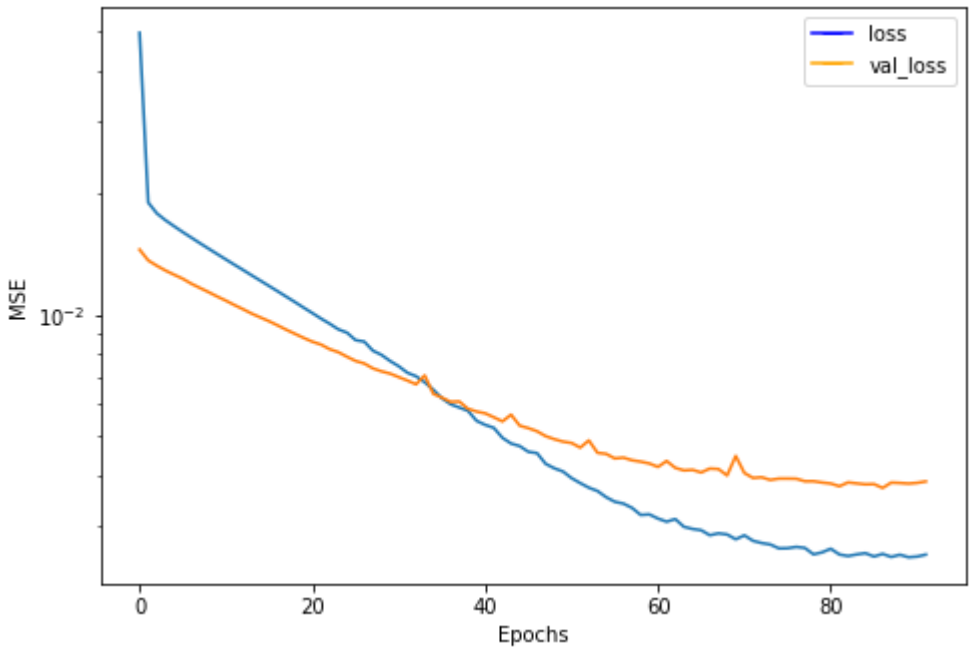


Fig. 32 Grafico loss di training e validation dell'autoencoder vanilla

Terminato l'allenamento della rete si può osservare l'andamento della loss in figura 32 durante tutte le epochs.
Sono mostrate le curve della loss di validation e training.
Si può osservare come entrambe scendano, inizialmente di più fino a poi scendere lentamente, la parte iniziale così verticale è dovuta alla scala logaritmica.
La discesa non è regolare, si nota infatti dei picchi, mentre alla fine la curva si sta rialzando, sintomo di un possibile overfitting, per questo l'allenamento viene fermato.

Fig. 33 Risultato del denoise effettuato dall'autoencoder vanilla su dataset dei cervelli senza rumore

Si può osservare dalla figura 33 che, sebbene non vi sia alcun rumore applicato, la rete non è abbastanza potente da poter ricreare le immagini con tutti i loro dettagli.

Questo risultato è stato ottenuto incrementando il layer centrale a 512, partendo dai 128 iniziali.
Di conseguenza la rete era inizialmente ancora meno potente.

Come parametro per valutare l'efficacia dei vari modelli si utilizza il mean squared error (MSE), ossia la media del quadrato degli errori.
In questo progetto lo utilizzeremo per verificare quanto un'immagine processata dalla rete sia simile a quella originale.
Il suo procedimento consiste nel calcolare l'errore su ogni singolo pixel in base all'intensità di colore e determinarne quello totale.
Questo viene fatto sull'immagine in output dalla rete e quella originale, per valutare quanto le due immagini si somigliano.
Minore è l'MSE migliore è la qualità dell'immagine.
Con questa rete l'MSE risulta essere 0.01989.

3.2 Risultati rumore con distribuzione normale

Conseguentemente ai risultati ottenuti con la rete neurale più semplice e appurato che non è abbastanza potente per sostenere i dettagli delle immagini, si procede testano un autoencoder più complesso e potente, ossia quello convoluzionale.

Viene utilizzato nuovamente il dataset dei cervelli.
Questa volta alle immagini presenti viene applicato un rumore, quello con distribuzione normale.
Per testare la rete gradualmente si comincia con rumori più lievi, dapprima con 0.05.
Allenando una rete più complessa si avrà un tempo di attesa maggiore.
L'allenamento del modello con input immagini di cervelli impiega poco meno di quattro minuti a epoch.
Le epochs totali vengono infatti diminuite da 100 a 50.

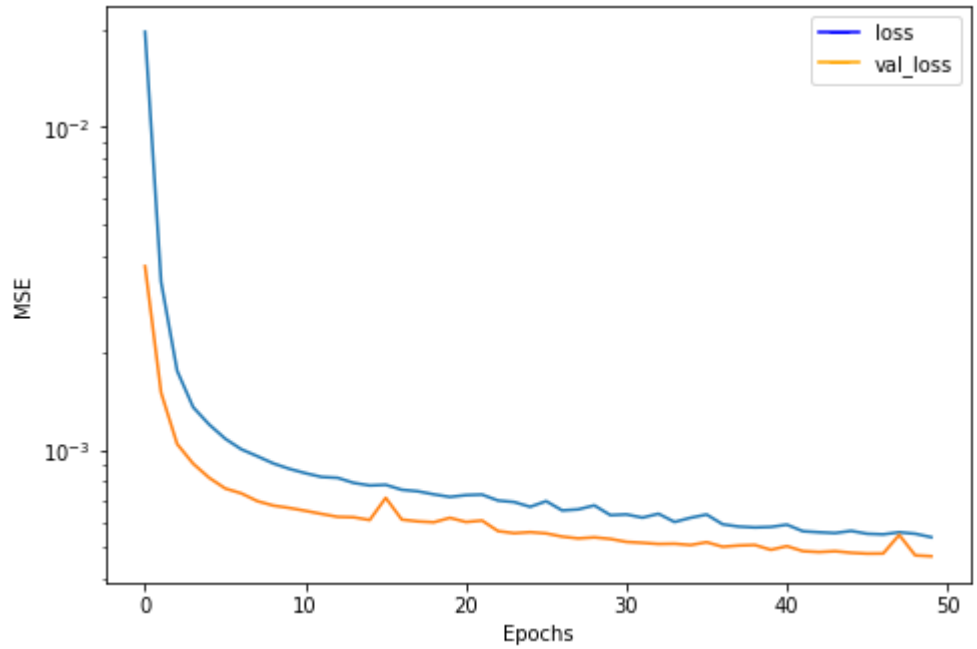


Fig. 34 Grafico loss di training e validation dell'autoencoder convoluzionale con il dataset dei cervelli e aggiunta di una distribuzione normale

Come nello scorso allenamento anche in questo l'andamento è lo stesso.
La curva che discende, inizialmente lentamente e in seguito più lentamente, questa volta però vengono fatte tutte e 50 le epochs senza essere interrotta prima.
Questo potrebbe significare che il modello potrebbe migliorare ancora.

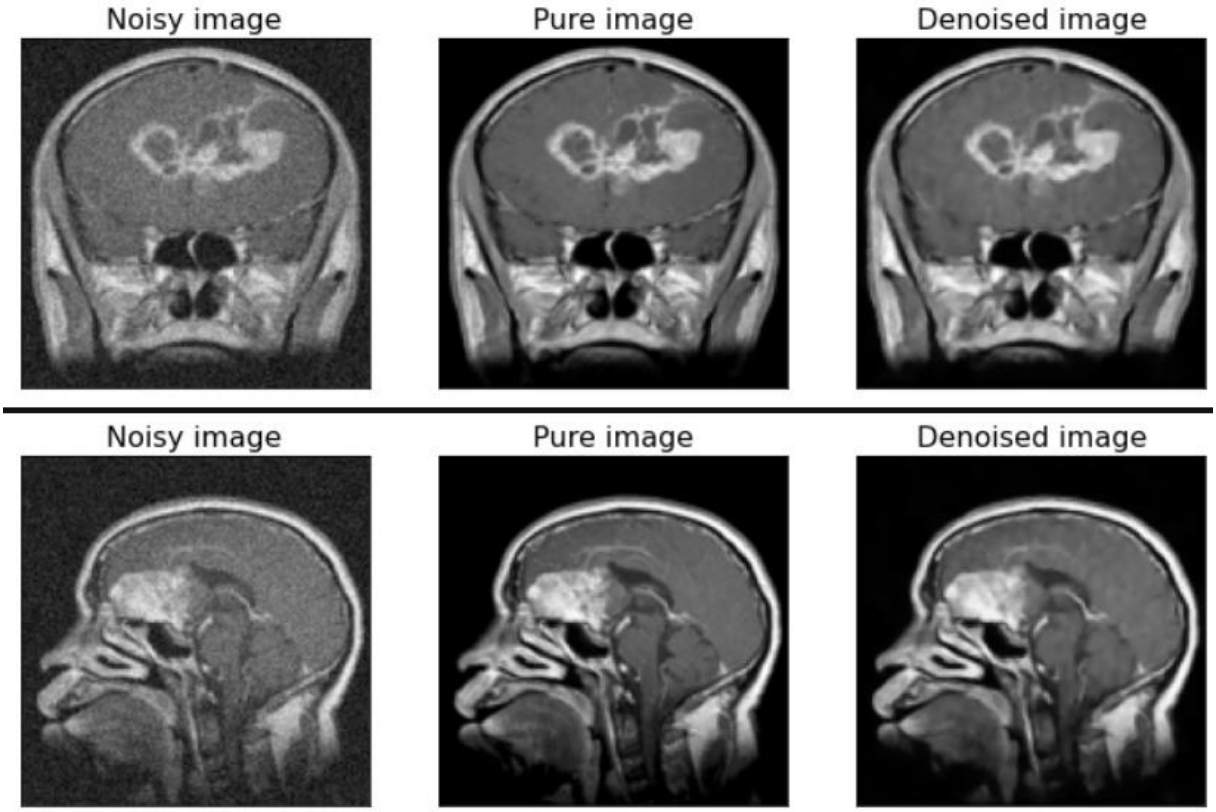


Fig. 35 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.05

Osservando la figura 35 si può subito notare la potenza superiore della rete.
Nonostante vi sia stato applicato del rumore l'autoencoder riesce a ricreare le immagini molto simili a quelle originali.
Le immagini presenti nella colonna centrale sono sconosciute alla rete neurale, vengono utilizzate solamente come confronto con quelle in uscita dal modello.

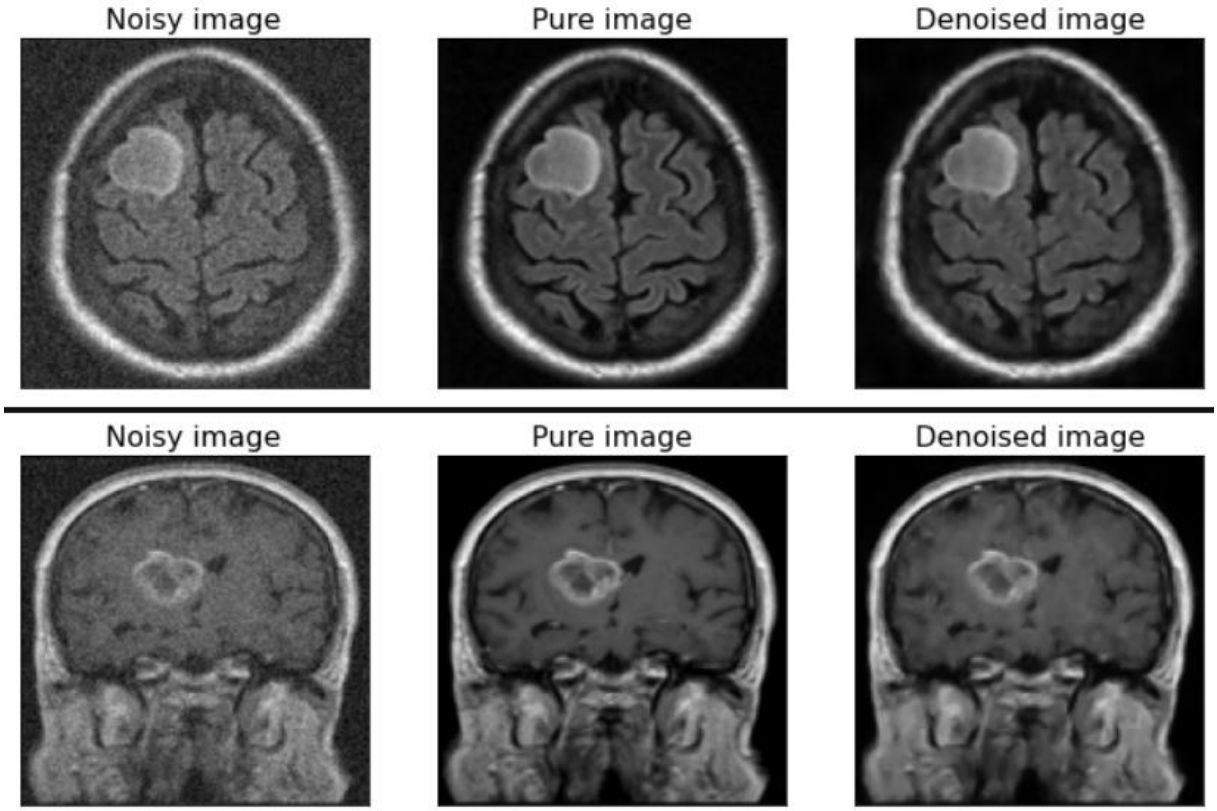


Fig. 36 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.05

Vedendo i risultati si può concludere che la rete convoluzionale sia visibilmente più potente di quella che utilizza solamente layer densi.
La qualità del lavoro svolto dall'autoencoder si può anche quantificare tramite l'MSE.
Con questo modello il valore è di 0.00066.
Confrontandolo con quello trovato utilizzando la rete più semplice si può subito notare quanto sia diminuito.

Una volta stabilito che la rete è abbastanza potente per riuscire a correggere un errore lieve si procede testandone di più gravi.

Proviamo quindi a cambiare il modello sostituendo i dati dell'allenamento con immagini più rumorose, questa volta con 0.1.

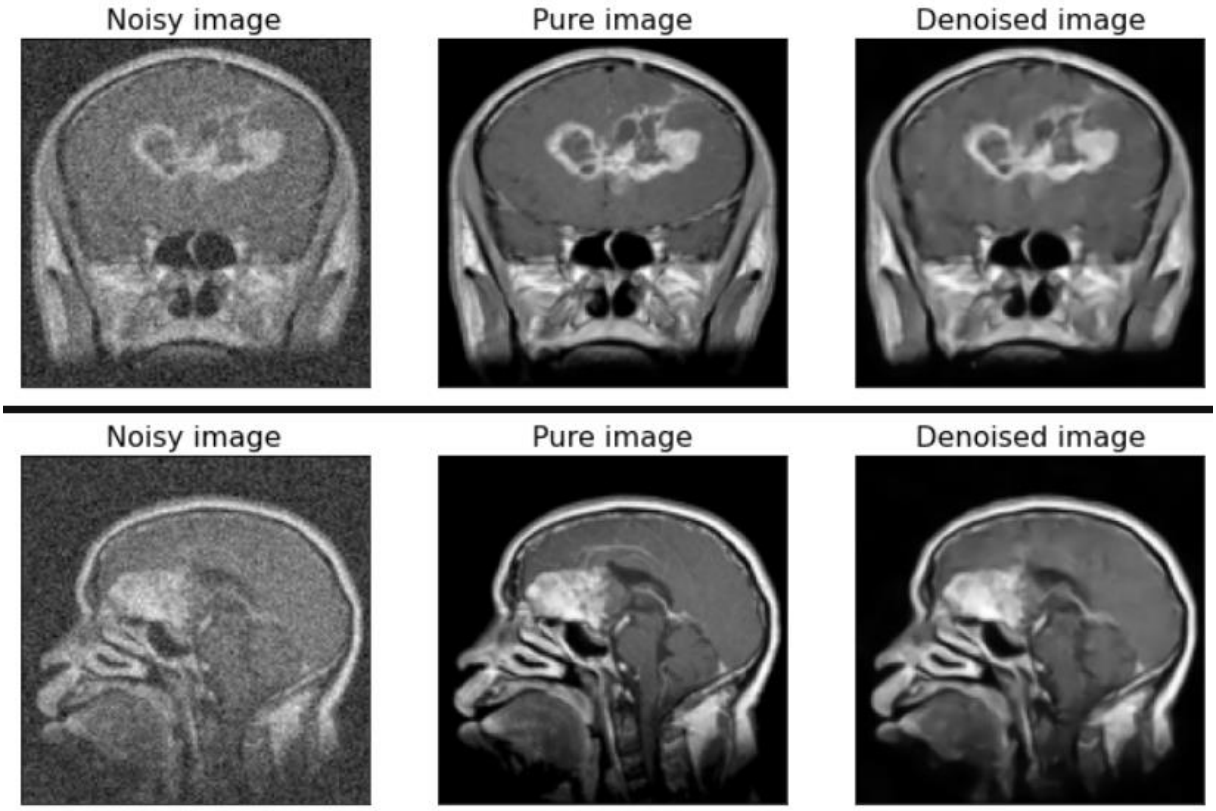


Fig. 37 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.1

Osservando la figura 37 si può notare come le immagini rumorose siano meno comprensibili delle scorse e più prive di dettagli.
Nonostante questo, la rete riesce a svolgere bene il suo lavoro, le immagini processate, infatti, hanno una buona qualità dato l'errore.

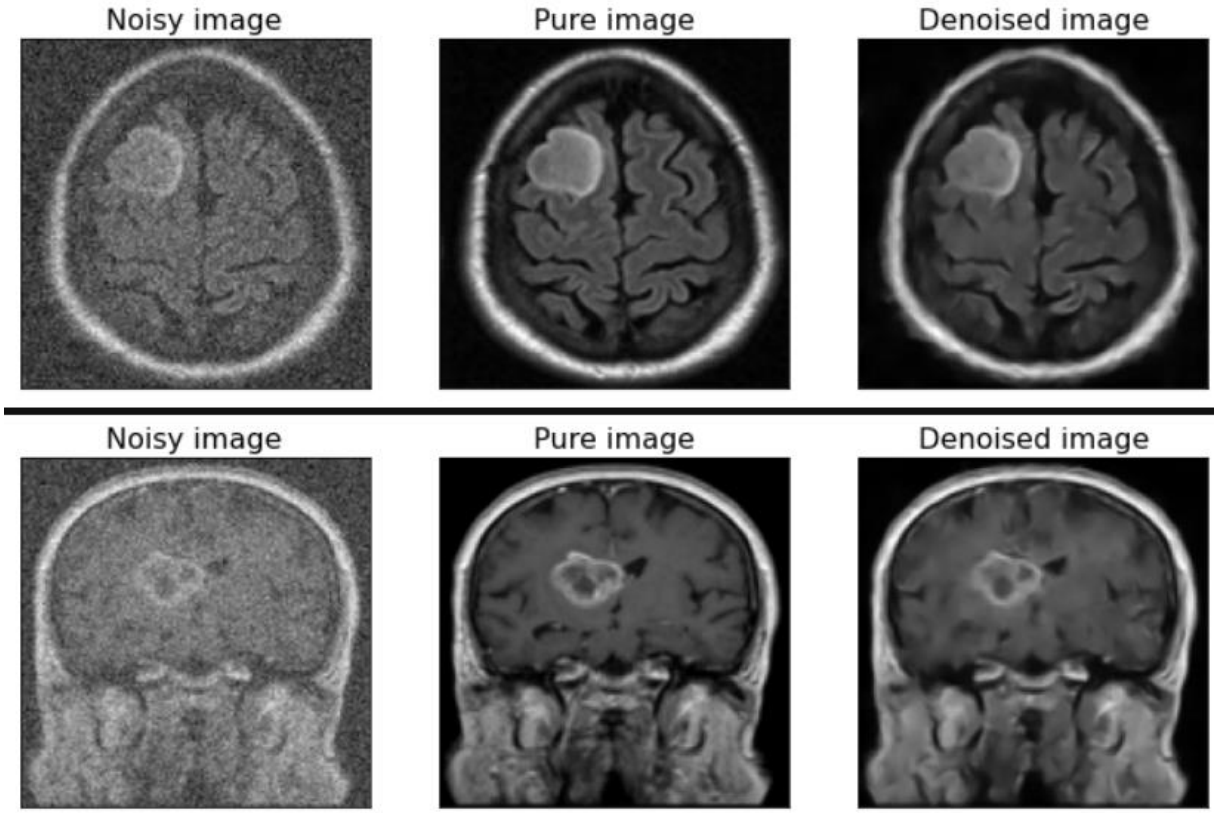


Fig. 38 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.1

Utilizzando il metro di confronto, ossia l'MSE, otteniamo un valore di 0.00125.

Viste le potenzialità di questo autoencoder si è deciso di testare un ulteriore rumore, ancora maggiore del precedente.
I dati di input saranno sempre gli stessi, con eccezione per le immagini rumorose. Quest'ultime questa volta avranno un valore di 0.2, il doppio dello scorso.

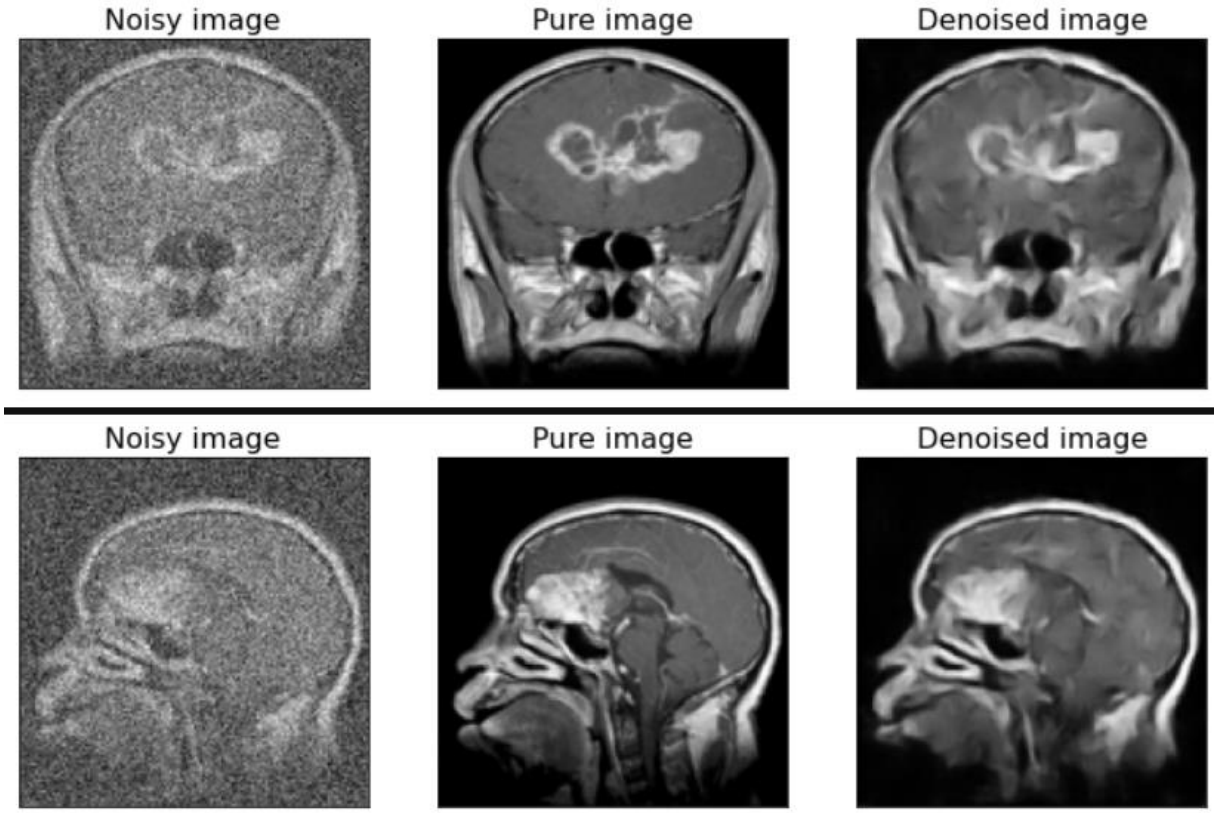


Fig. 39 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.2

Questa volta il rumore è presente in grandissime quantità, le immagini infatti sono difficilmente riconoscibili e vengono nascosti la maggior parte dei dettagli.
Questa situazione, infatti, è spinta agli estremi unicamente per verificare le potenzialità della rete.
In un caso come questo l'immagine è praticamente compromessa, dato che tutti i dettagli vengono persi.
È comunque interessante vedere come si comporta l'autoencoder.
L'immagine processata infatti conserva delle proprietà importanti.
Il modello ha imparato a riconoscere che lo spazio esterno alle teste è interamente nero e viene rimosso tutto il rumore.
La struttura viene mantenuta e alcuni dettagli ricostruiti.
Data la gravità dell'immagine di partenza il risultato è considerabile buono.

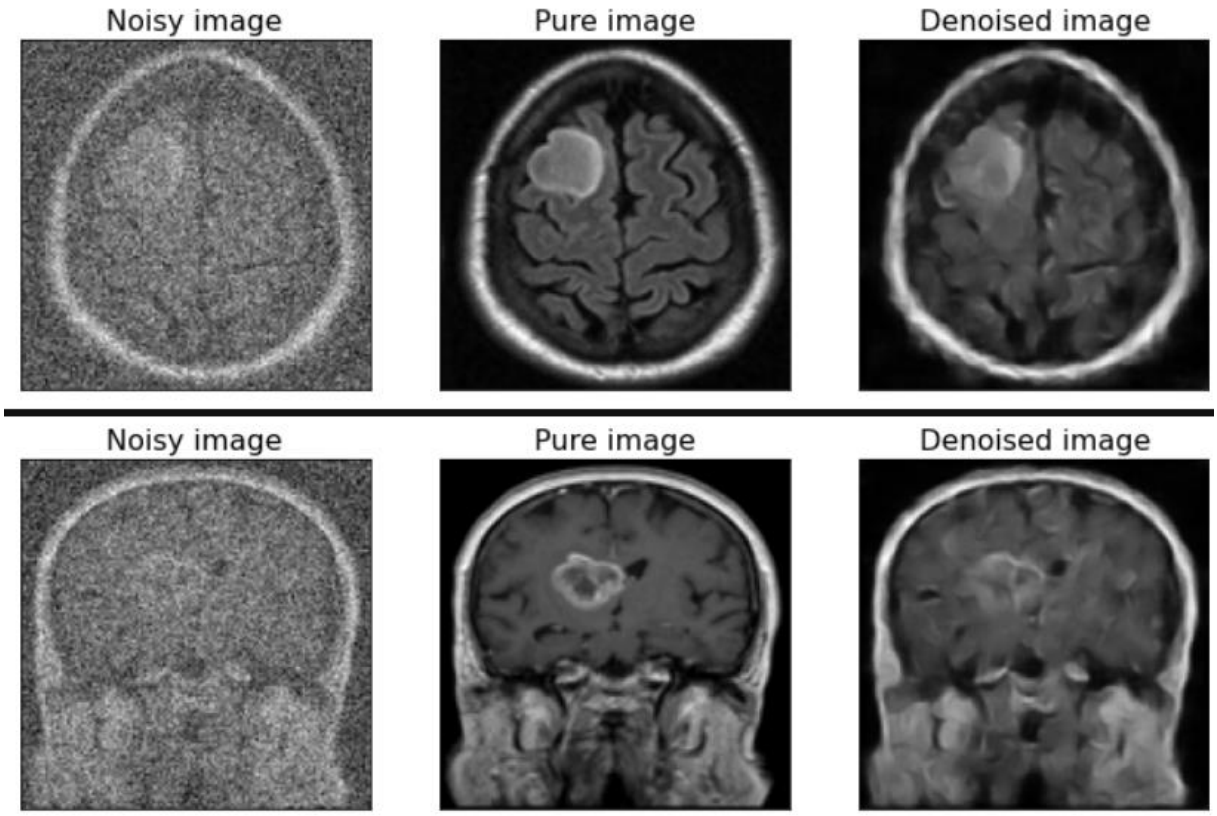


Fig. 40 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con distribuzione normale di valore 0.2

L'MSE di questo modello è di 0.00307.
Nonostante sembri un numero molto piccolo è più che raddoppiato rispetto al modello precedente.

Terminati i test concernenti il rumore dato dalla distribuzione normale si può confrontare i vari risultati.
Nel grafico sottostante sono presenti quattro barre sull'asse X raffiguranti le quantità di rumore applicato alle immagini, mentre sull'asse Y si trova il valore di MSE.
La prima barra rappresenta i risultati dati dalla prima rete neurale; invece, le altre tre raffigurano la seconda rete.

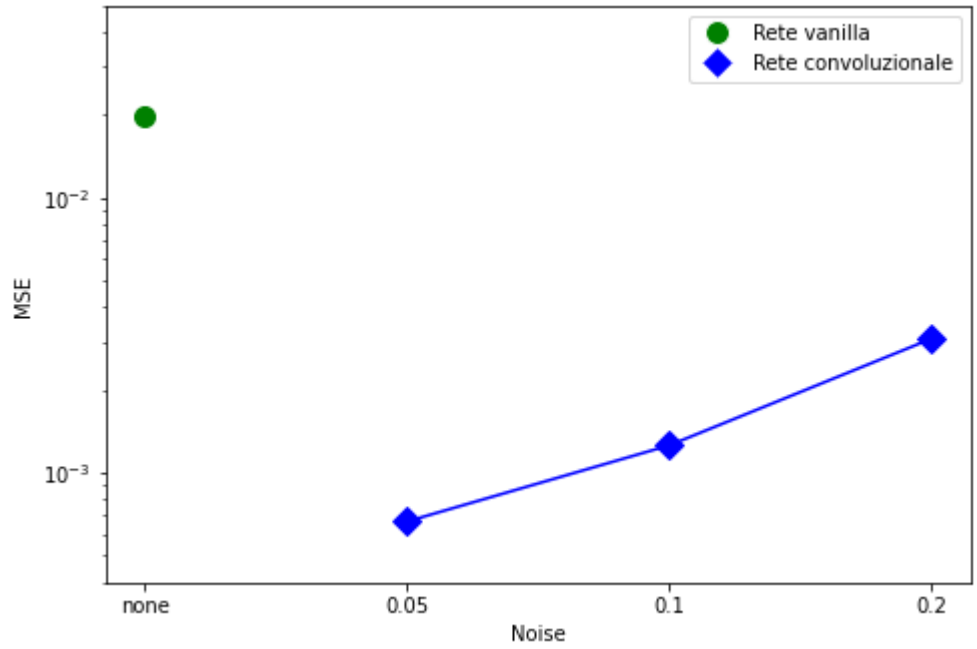


Fig. 41 Grafico della distribuzione normale in relazione al MSE

Osservando la figura 41 la sproporzionalità della prima barra è chiara.
La rete utilizzata per il primo test, ossia quella densa, nonostante sia stata allenata con un dataset contenente immagini prive di rumore, risulta possedere un MSE molto più grande degli altri.
Le altre barre appartengono alla rete convoluzionale e nella situazione dove le immagini sono state alterate con molto rumore, il modello riesce comunque ad ottenere un MSE quasi sette volte minore rispetto la rete densa.

I risultati della rete convoluzionale mostrano inoltre un aumentare del MSE con l'aumentare del rumore. Questo andamento è utile per capire che la rete funziona correttamente, dato che nelle immagini con più rumore vi sono meno dettagli.

La sperimentazione con le immagini aventi del rumore dato da distribuzione normale sono state effettuate solamente sul dataset dei cervelli.
Questo è stato fatto perché, visti i risultati positivi, era supponibile che la rete funzionasse allo stesso modo anche sull'altro dataset.
Inoltre, si voleva concentrarsi maggiormente sulla rimozione di righe e pixel poiché rappresentano meglio gli errori che si presentano nella realtà.

3.3 Risultati rumore con rimozione righe

3.3.1 Dataset Cervelli

La rete convoluzionale ha ottenuto ottimi risultati con l'allenamento mediante immagini con rumore dato da distribuzione normale.
Una volta testata l'efficacia della rete si può procedere testando lo stesso autoencoder con rumori differenti.
Il test che viene effettuato questa volta è sempre sul dataset dei cervelli, solamente questa volta il rumore applicato sarà conseguenza della rimozione delle righe.
Come i test precedenti si andrà a step, incrementando il rumore.
Il primo che viene mostrano è il risultato della rimozione di 10 righe.

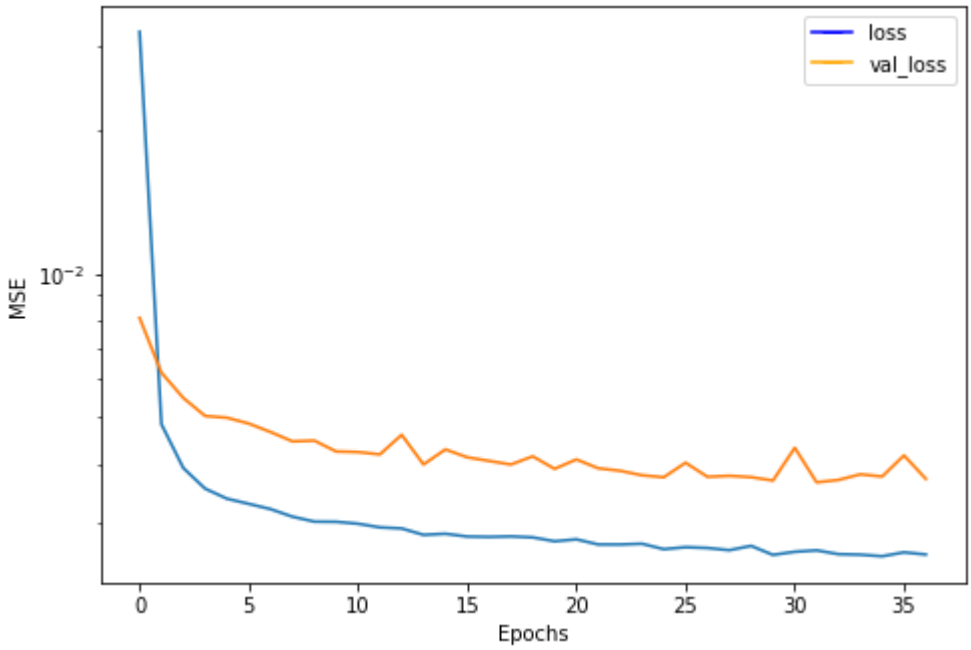
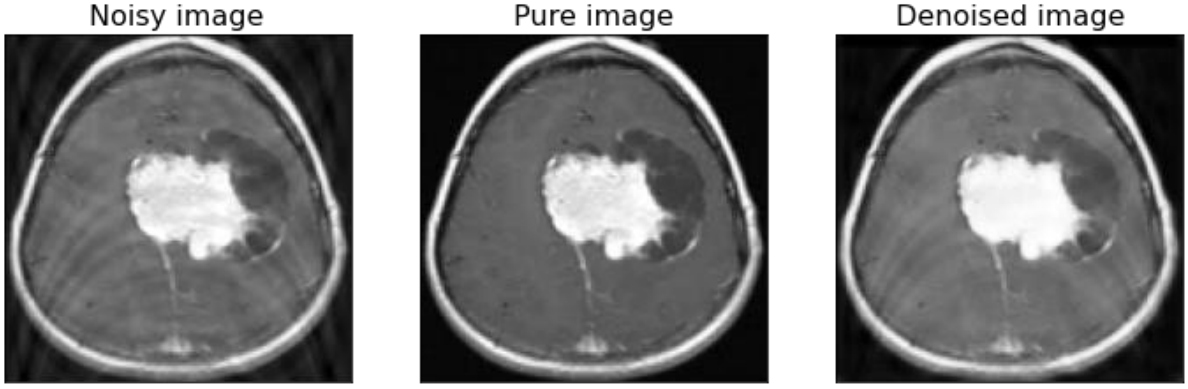


Fig. 42 Grafico loss di training e validation dell'autoencoder convoluzionale con il dataset dei cervelli e rimozione di righe

Nel riassunto di questo allenamento si può riconoscere una somiglianza con lo scorso.
Questa volta, a differenza della scorsa, l'allenamento viene interrotto prima a causa di una loss che non diminuiva.



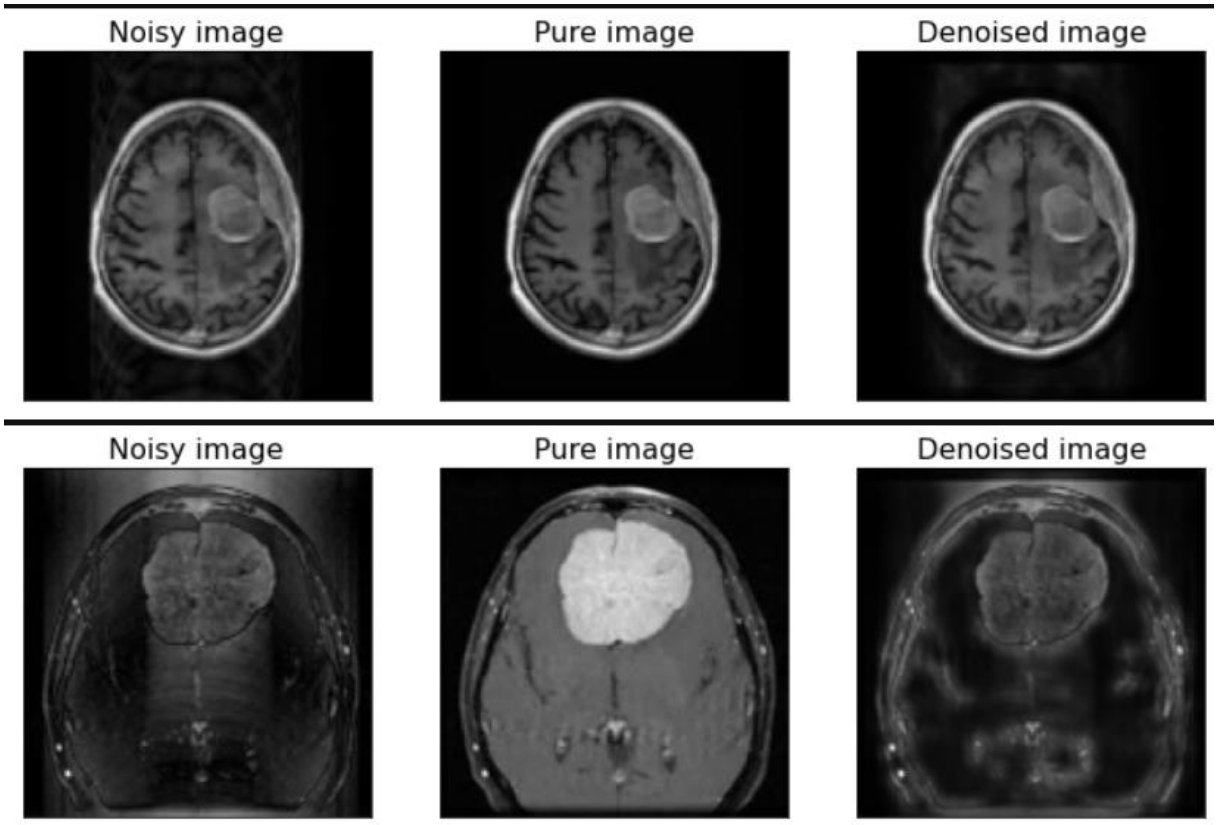


Fig. 43 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con rimozione di 10 righe

Come si può osservare dalla figura 43, i rumori dovuti alla rimozione di righe sono completamenti differenti l'uno dall'altro.
Questo fenomeno rende più difficoltoso l'apprendimento della rete, i risultati, infatti, sono molto variabili.
Vengono mostrati tre casi distinti: nel primo caso l'autoencoder riesce a ridurre il rumore presente e ad avere un'immagine più simile all'originale.
Nel secondo caso il risultato è molto simile all'immagine rumorosa.
Nell'ultimo caso invece, la rimozione delle righe ha preso in causa linee che hanno peggiorato visibilmente l'immagine e questo non ha permesso al modello di svolgere un buon lavoro.
Si può osservare infatti che nonostante ci siano stati cambiamenti da quella rumorosa a quella processata non vi sono miglioramenti che la portano ad essere più simile all'originale.
Valutando i risultati dal punto di visto dell'MSE si ottiene un valore di 0.00457.

Dopo aver analizzato il comportamento della rete con un determinato rumore si può provare ad aumentarlo.
Cambiamo quindi il numero di righe rimosse da 10 a 20.

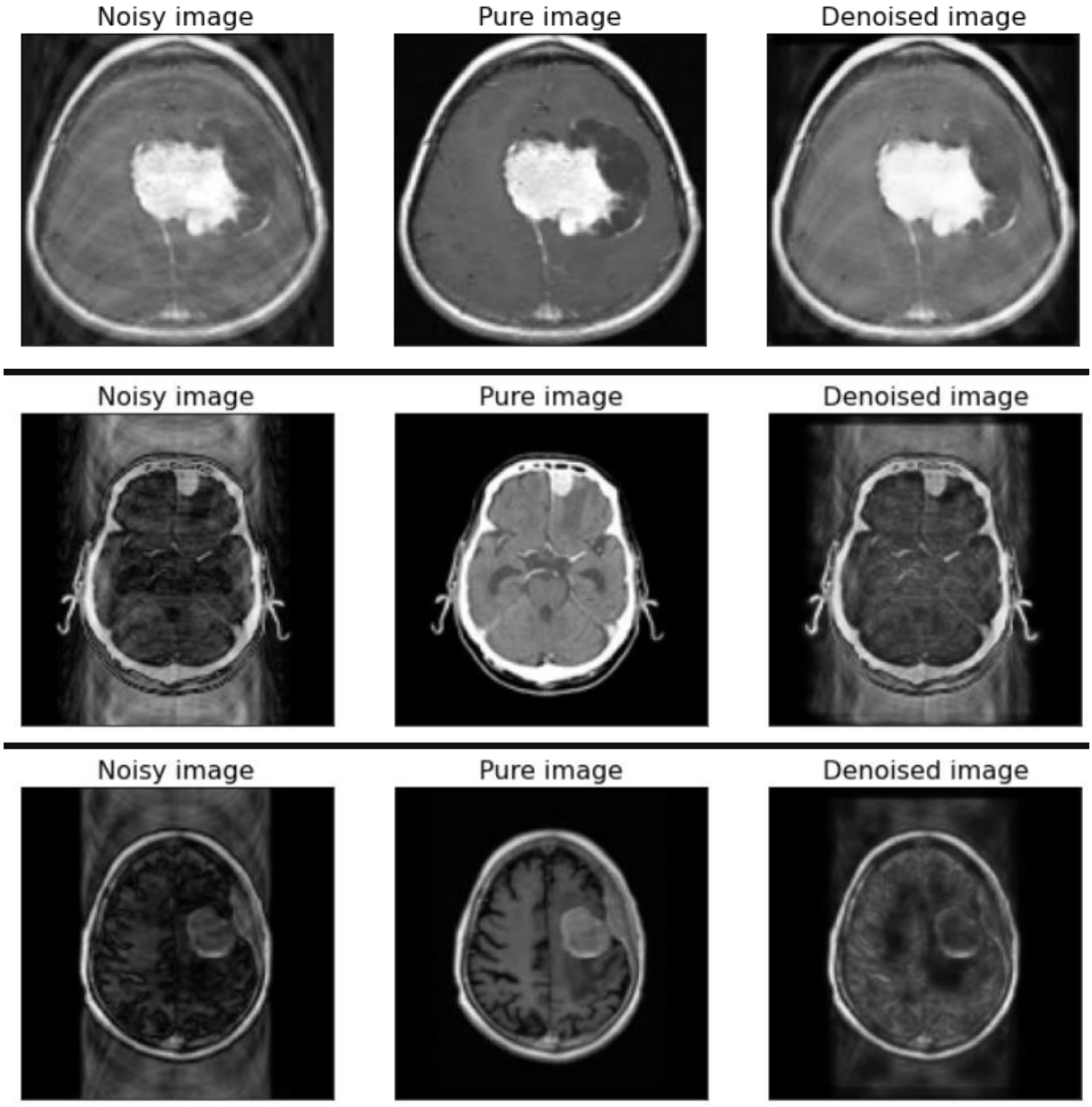


Fig. 44 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con rimozione di 20 righe

La situazione è simile alla precedente.
Diversi casi che a dipendenza delle righe eliminate ci si trova con immagini più o meno rumorose.
In alcune circostanze la rete riesce a togliere buona parte del rumore, mentre in altre il rumore è talmente tanto che non riesce a correggerlo.
Il valore di MSE che si calcola è di 0.0093.
Questo aumento di errore del doppio è sintomo dell'aumento delle immagini rumorose.

A questo punto procedere per questa strada, continuando ad aumentare il numero di righe rimosse, è poco speranzoso.

Ci si troverebbe ad avere più immagini rovinate con una quantità di rumore maggiore e la rete in difficoltà, con l'aumentare dell'MSE.
L'autoencoder risulta già essere poco efficiente con un numero basso di righe rimosse, con un numero maggiore la situazione sarebbe peggiore.

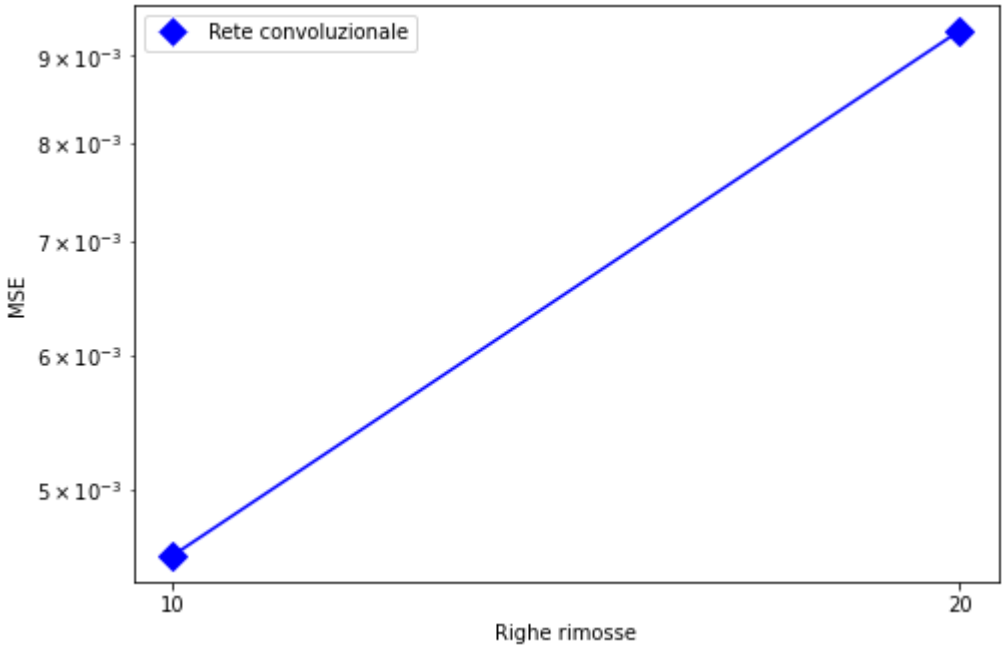


Fig. 45 Grafico delle righe rimosse in relazione al MSE per il dataset dei cervelli

In questo grafico viene mostrato l'errore (MSE) in relazione alle righe rimosse.
Come era prevedibile rimuovendo più righe l'errore aumenta, in questo caso togliendo il doppio delle righe l'errore aumenta del doppio.

Un ulteriore test da fare è verificare se la rete può ottenere risultati migliori lavorando con il dataset dei cuori.

3.3.2 Dataset cuori

Andiamo quindi a testare il dataset dei cuori per vedere se ci sono risultati migliori rispetto al dataset dei cervelli.
Avendo delle immagini con dimensioni differenti bisogna lavorare in rapporto con le righe rimosse.
Il dataset dei cuori ha immagini con un'altezza di 256 pixels, il doppio di quello dei cervelli.
Di conseguenza verranno tolte il doppio delle righe per avere dei risultati in rapporto più simili.
Il primo test che viene svolto è con 20 righe random rimosse.

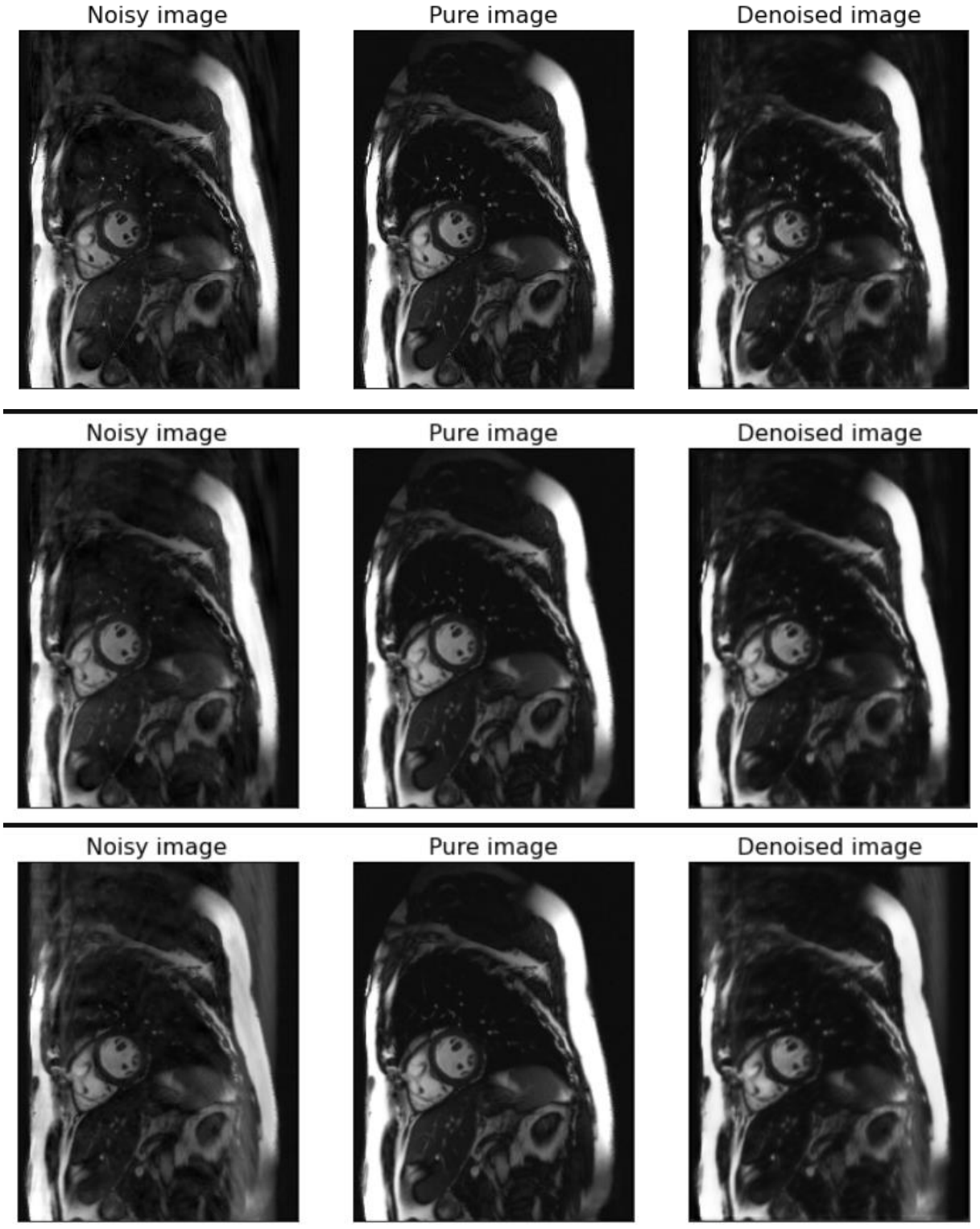


Fig. 46 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cuori con rimozione di 20 righe

I disturbi sono variabili ma tendenzialmente le immagini sono riconoscibili e i dettagli sono presenti.

Quando il rumore è più lieve la rete riesce a rimuoverlo parzialmente o in buona parte, mentre quando è più presente non riesce ad ottenere un'immagine simile a quella originale.
Calcolando l'MSE si trova un valore di 0.00607.
Questo modello sembra quindi funzionare in parte, per testarlo meglio si prova a cambiare input passando immagini più rovinate.
In questo caso si utilizzeranno immagini con 40 righe rimosse, ossia il doppio di prima.

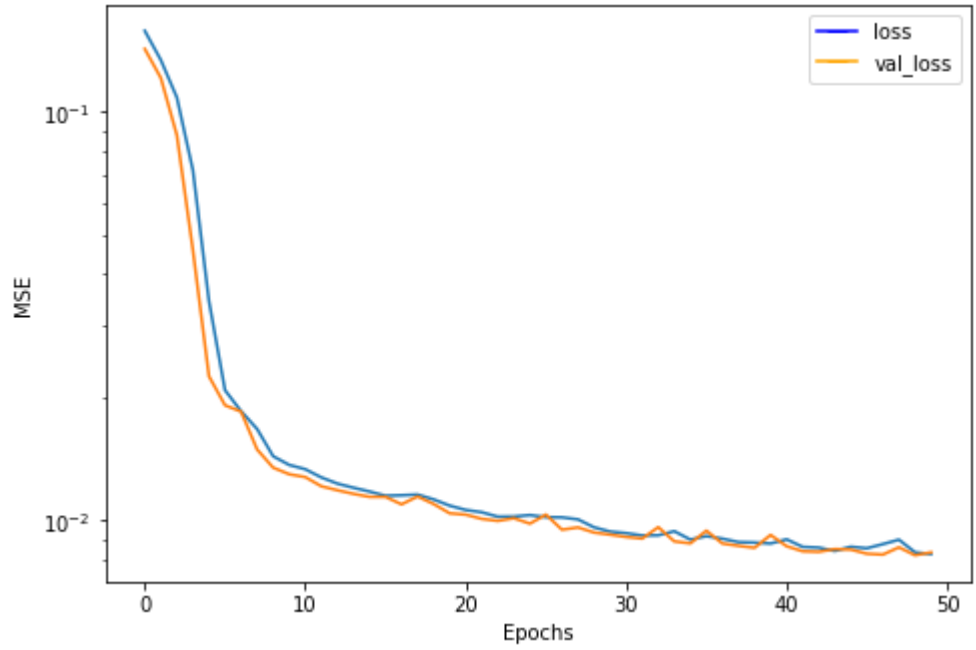
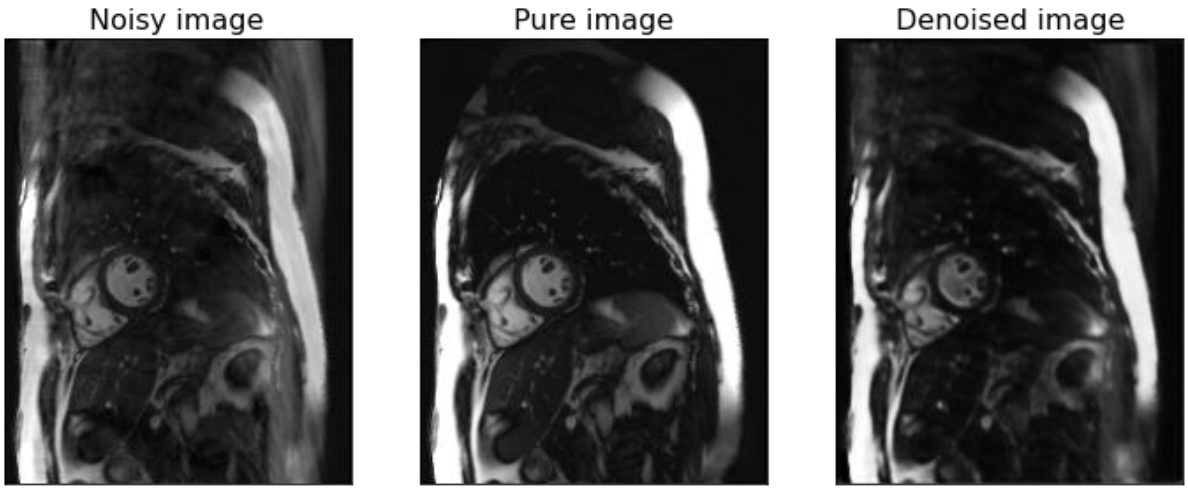


Fig. 47 Grafico loss di training e validation dell'autoencoder convoluzionale con il dataset dei cuori e rimozione di righe

Le loss di questo allenamento, differentemente dalle scorse, vanno di pari passo.
Entrambe diminuiscono con dei picchi fino alla cinquantesima epoch.



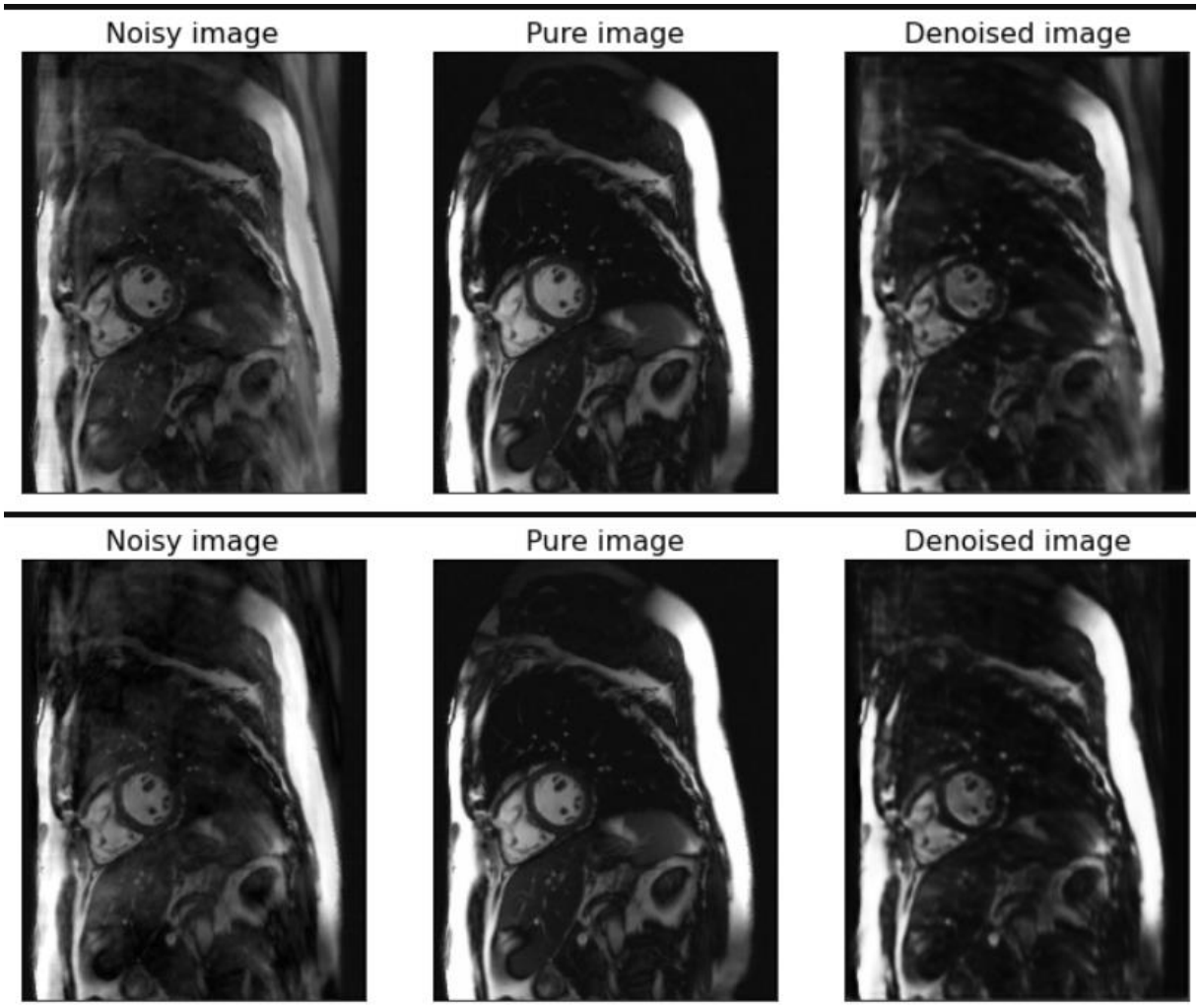


Fig. 48 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cuori con rimozione di 40 righe

Si possono osservare immagini più rovinate e con alcuni dettagli affievoliti.
Valutando il lavoro della rete si può notare come paradossalmente sembri funzionare meglio del modello precedente.
Infatti, nella maggior parte dei casi, si vede un miglioramento delle immagini, che vengono poi a somigliare di più a quelle originali.
Anche nelle situazioni con più rumore questo modello migliora le immagini, cosa che lo scorso più volte non riusciva a fare.
Nonostante questo, l'MSE misurato è maggiore, ossia di 0.00890.

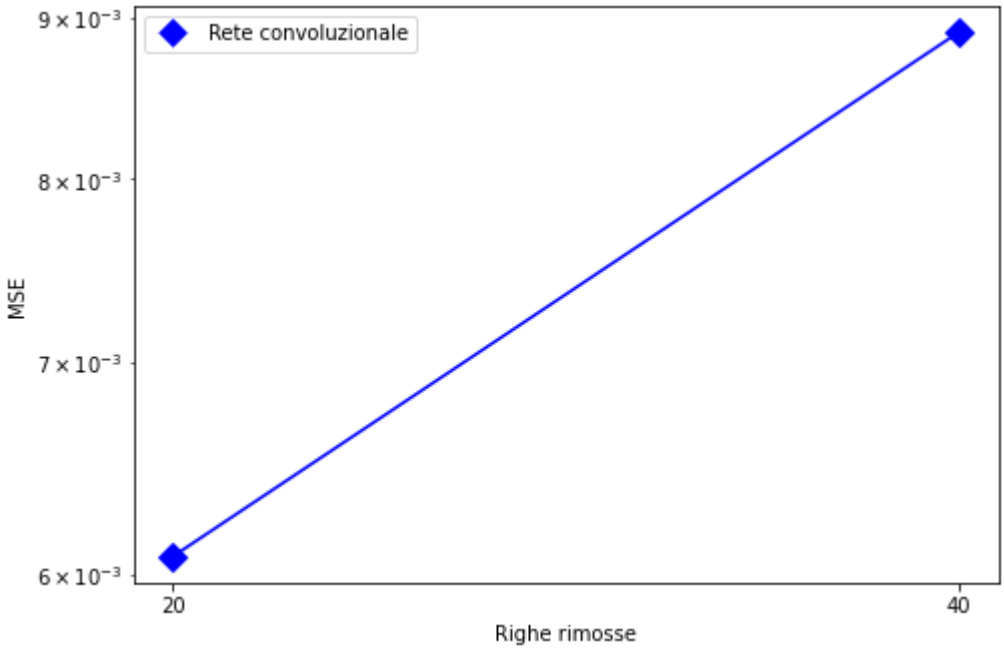


Fig. 49 Grafico delle righe rimosse in relazione al MSE per il dataset dei cuori

Grafico molto simile al precedente, mostra come l'errore aumenta all'aumentare delle righe rimosse.

3.4 Risultati rumore con rimozione pixel

3.4.1 Dataset cervelli

Un altro test che si può effettuare per simulare una situazione reale è la rimozione di pixels.
Come la rimozione delle righe, anche questa, mostra meglio gli errori nei quale potrebbe incorrere una risonanza magnetica.
L'allenamento verrà svolto con gli stessi parametri degli scorsi test e il dataset di input sarà inizialmente quello di cervelli.
Con questo tipo di rumore in questo determinato dataset si sono riscontrati dei problemi con la rete.
Durante l'allenamento la loss non scendeva, di conseguenza l'autoencoder non stava imparando.
Analizzando le possibili cause si è potuto costatare che probabilmente il problema era legato alla funzione relu, una funzione legata ai layer convoluzionali, che azzerava i pesi impedendo l'allenamento.
Per ovviare a questo imprevisto sono stati aggiunti i layer di BatchNormalization mostrati in precedenza.
Questa situazione si è presentata solamente nei modelli di 1 ogni 4 e 8 pixels rimossi.
Quindi per primo caso, ossia 1 ogni 16, è stata usata la rete originale, per le altre quella che presenta le aggiunte.
Per testare se i layer portassero un vantaggio anche a livello di loss è stato allenato il modello di 1 pixel rimosso ogni 16 anche con la rete modificata.
Sono stati effettuati due allenamenti e in entrambi la loss era maggiore, in uno anche di molto.

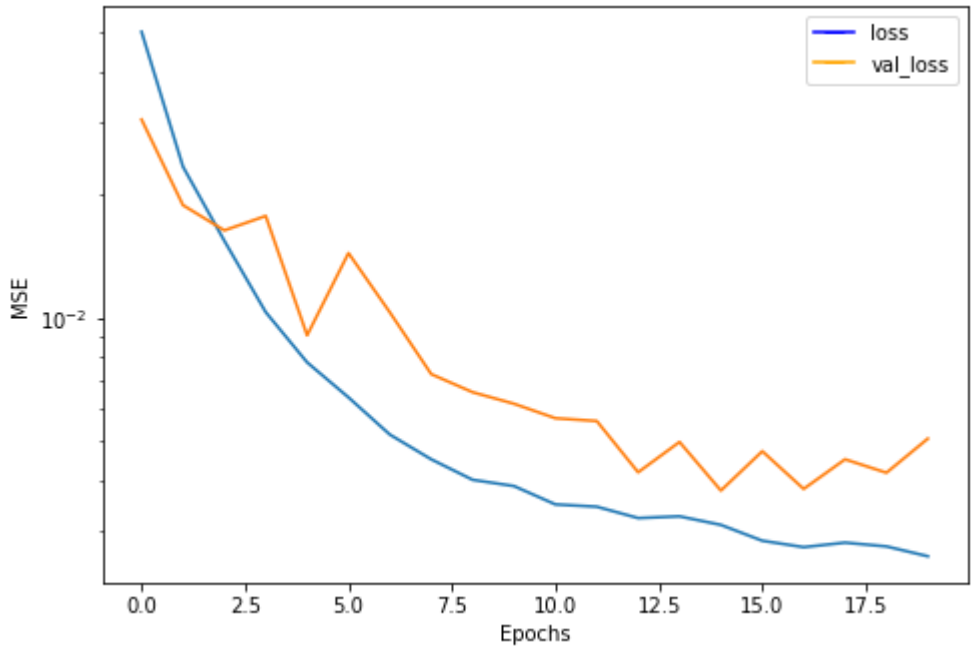
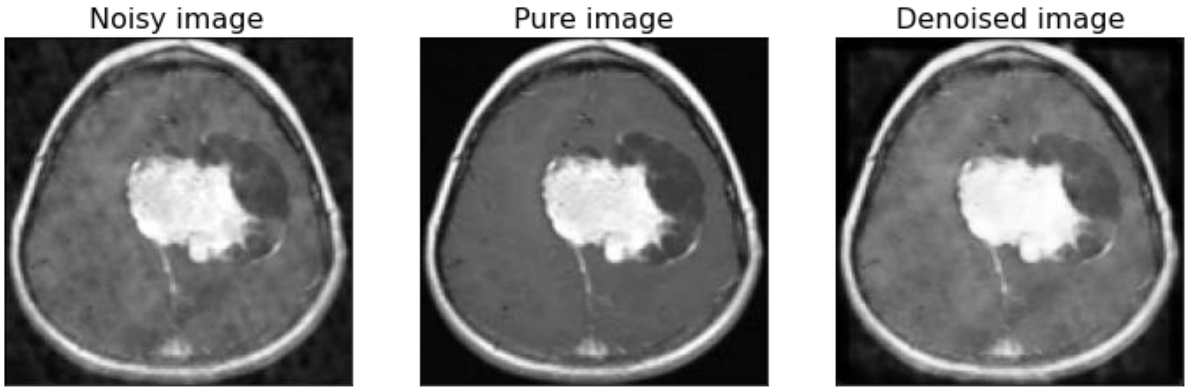


Fig. 50 Grafico loss di training e validation dell'autoencoder convoluzionale con il dataset dei cervelli e rimozione di pixels

Viene mostrato il grafico dell'allenamento del modello, questa volta con la rete contenente i layer aggiuntivi.
Si può osservare una differenza con gli altri grafici, questo presenta molti più picchi. Mentre la curva della loss del training scende regolare quella della validation è molto irregolare, ha picchi molto superiori a quelli visti con la rete convoluzionale iniziale.

Nel primo risultato mostrato è stato tolto 1 pixel ogni 16 in modo randomico.



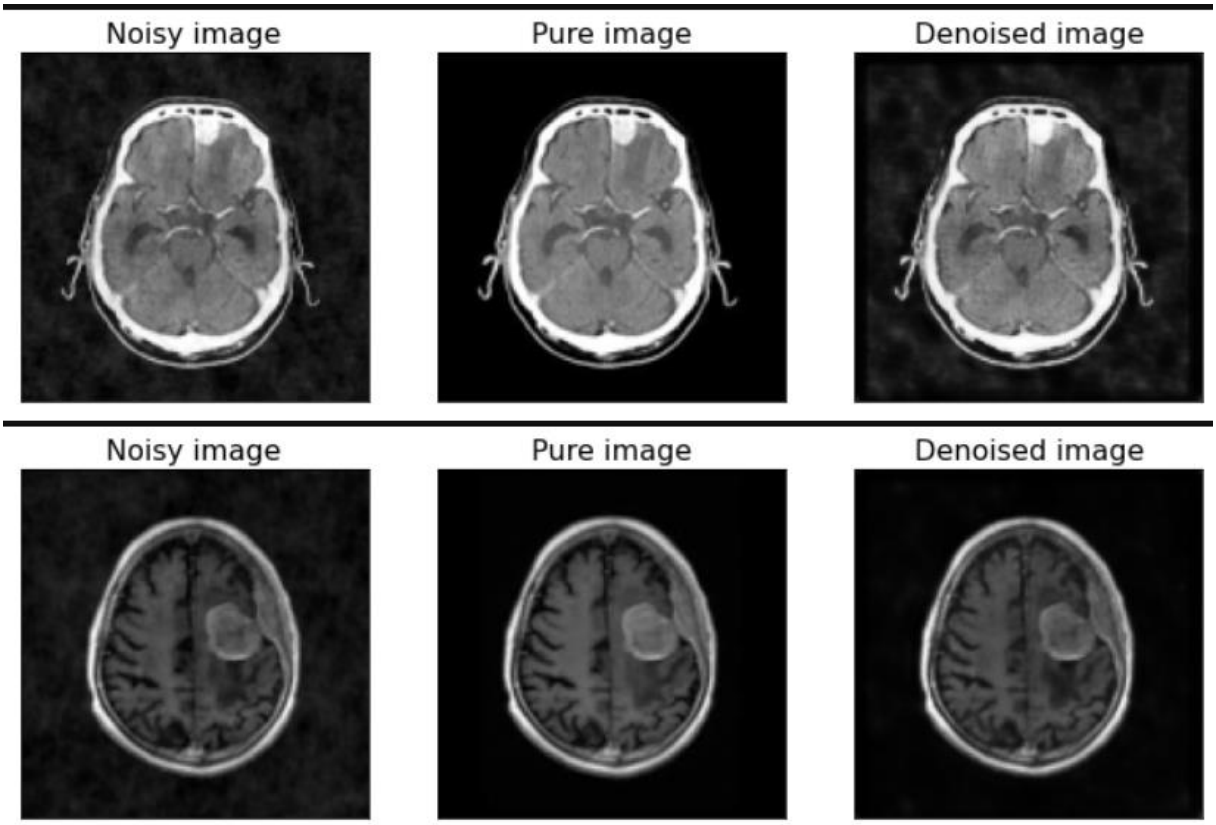


Fig. 51 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con rimozione di 1 pixel ogni 16

Gli output di questo modello sono molto differenti tra loro.
Le situazioni presenti sono accomunate in più casi come i modelli con la rimozione di righe.
Nella figura 51 sono riassunti alcuni dei casi più frequenti.
Ci sono molte immagini alle quali il rumore viene tolto solo parzialmente e in determinate zone.
Ci sono invece occasioni in cui gli input sono praticamente uguali agli output, il risultato è quindi che il rumore non viene rimosso.
Infine, nell'ultima all'ultima riga della figura si può osservare come l'autoencoder abbia svolto il suo lavoro rimuovendo buona parte dei disturbi, ottenendo un'immagine più chiara.
Il calcolo del MSE è stato fatto tre volte, una per la rete convoluzionale iniziale e due per quella con l'aggiunta di layer.
Il valore di quella iniziale è di 0.00417, mentre per l'altro autoencoder il primo test ha avuto come risultato 0.00442, l'altro 0.00557.

A questo punto si prova a testare delle immagini con un rumore maggiore.
Questa volta viene tolto 1 pixel ogni 8, quindi verranno rimossi il doppio dei pixel rispetto alla situazione precedente.
Come detto in precedenza da ora verrà usata la rete con i layer aggiuntivi.

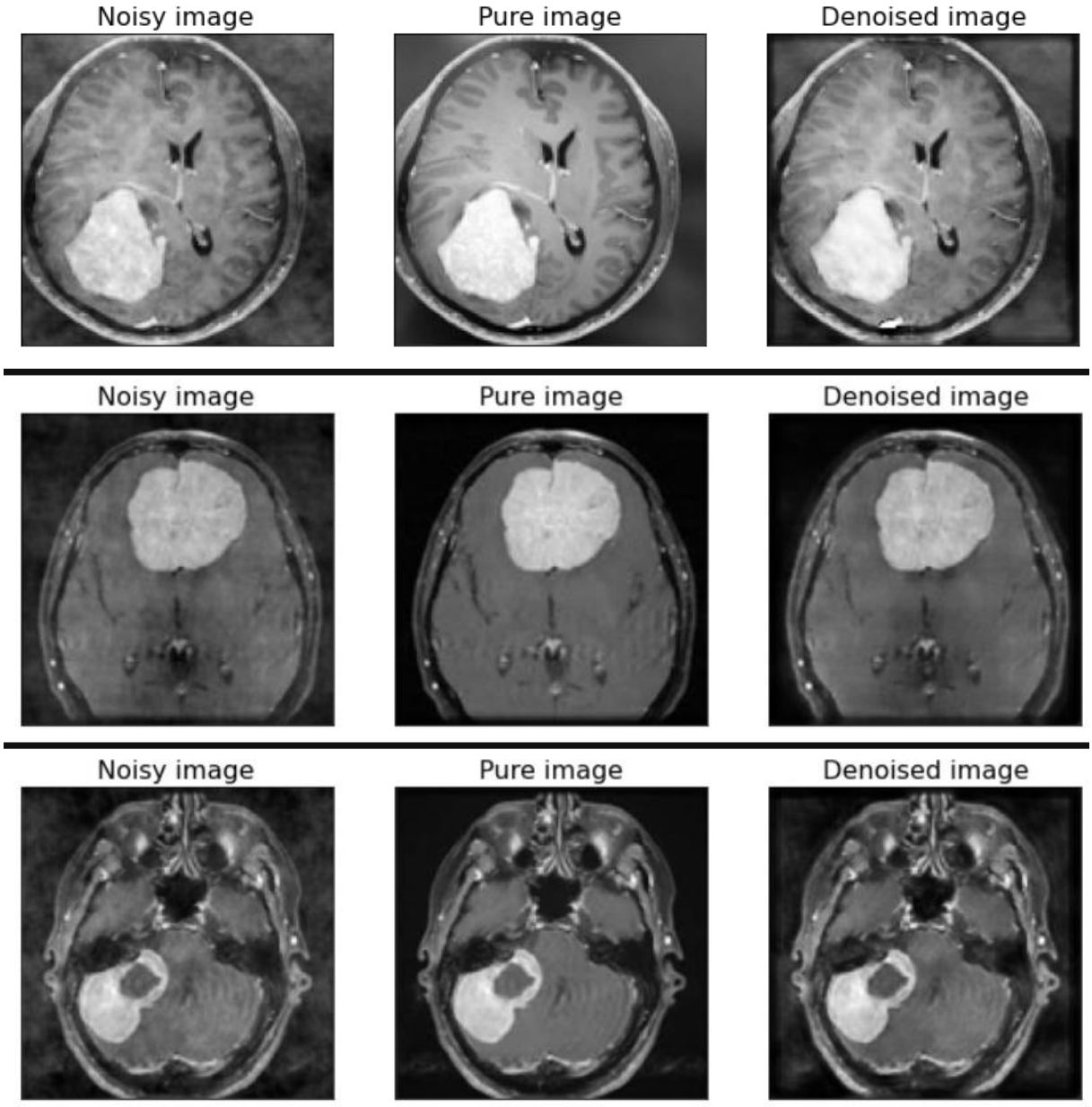


Fig. 52 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con rimozione di 1 pixel ogni 8

I risultati non di discostano molto da quelli precedenti, essi sono sempre molto variabili. La rete reagisce diversamente in base all'immagine e al tipo di errore.
Il valore di MSE calcolato è di 0.00862.

Si può procedere con l'ultimo test, togliendo il doppio dei pixels rispetto al modello precedente.
Di seguito le immagini del modello con 1 pixel rimosso ogni 4.

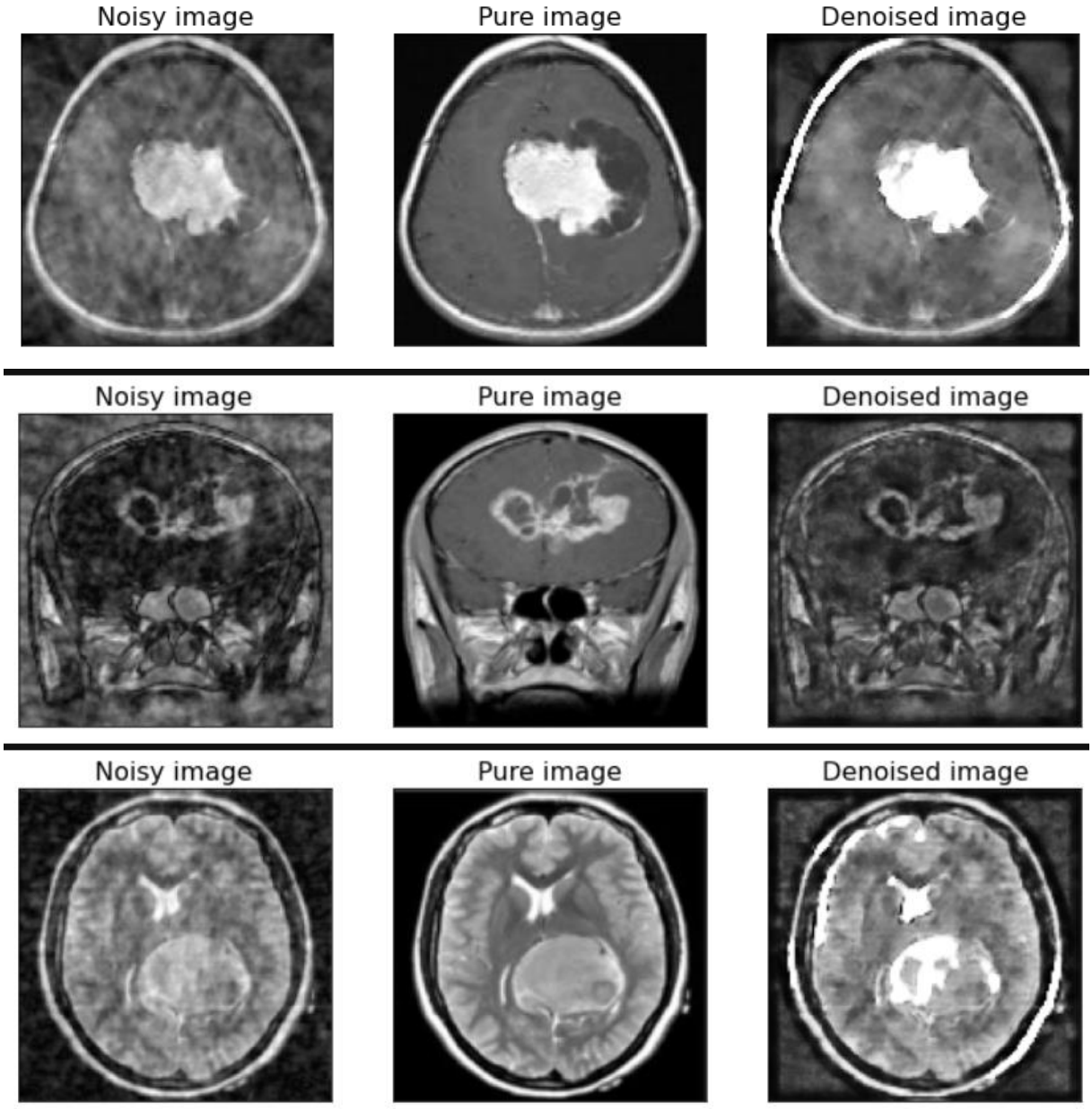


Fig. 53 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cervelli con rimozione di 1 pixel ogni 4

Le immagini con questo modello risultano molto più rovinate rispetto quelle precedenti. I dettagli sono quasi completamente oscurati dal rumore, la correzione diventa quindi molto complicato.

Infatti, i risultati non si avvicinano per niente alle immagini originali.

L'MSE calcolato è di 0.01894.

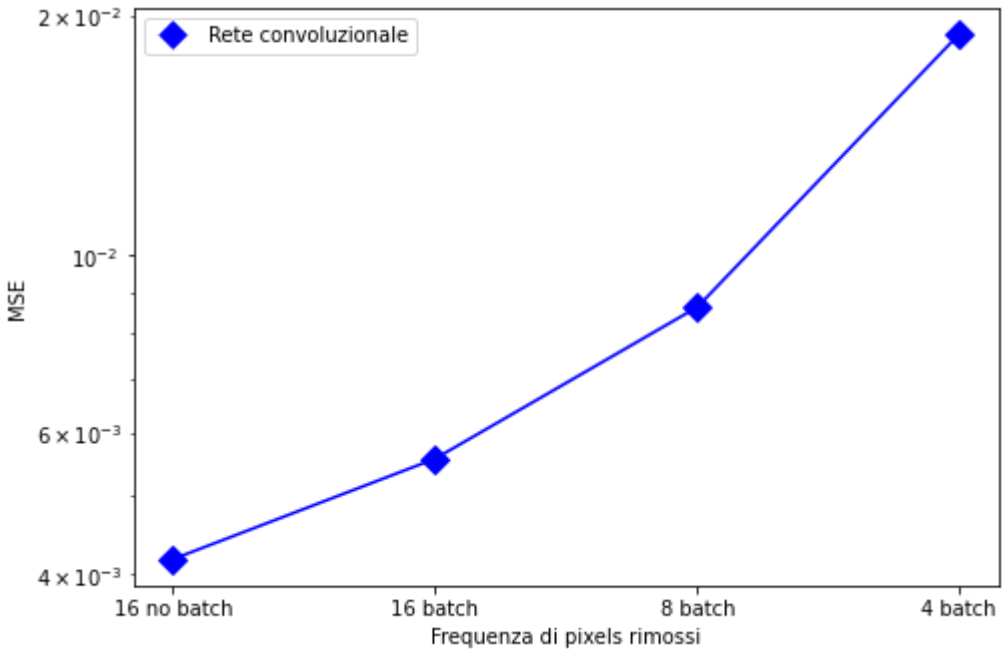


Fig. 54 Grafico delle fraquenza di pixels rimossi in relazione al MSE per il dataset dei cervelli

In questo grafico viene mostrato l'MSE in relazione alla quantità di pixels rimossi.

Sono mostrati i tre step: 1 pixel rimosso ogni 16, ogni 8 e ogni 4.
Per la prima situazione è mostrato il risultato sia con sia senza i layer aggiuntivi.
Come accennato in precedenza l'MSE senza i layer aggiuntivi risulta minore rispetto a quello con.
Di seguito l'errore tende ad aumentare parecchio con la rimozione di più pixels.

3.4.2 Dataset cuori

Lo stesso test può essere effettuato sul dataset dei cuori.
La frequenza con i quali i pixel verranno rimossi sono: uno ogni 20, uno ogni 8 e uno ogni 4.
Il primo valore è cambiato da 16 a 20 per via della dimensione delle immagini; infatti, non essendo 20 un divisore di 128 il valore è stato aggiustato a 20.
Il primo test che viene effettuato è quello con meno rumore, ossia con 1 pixel rimosso random ogni 20.

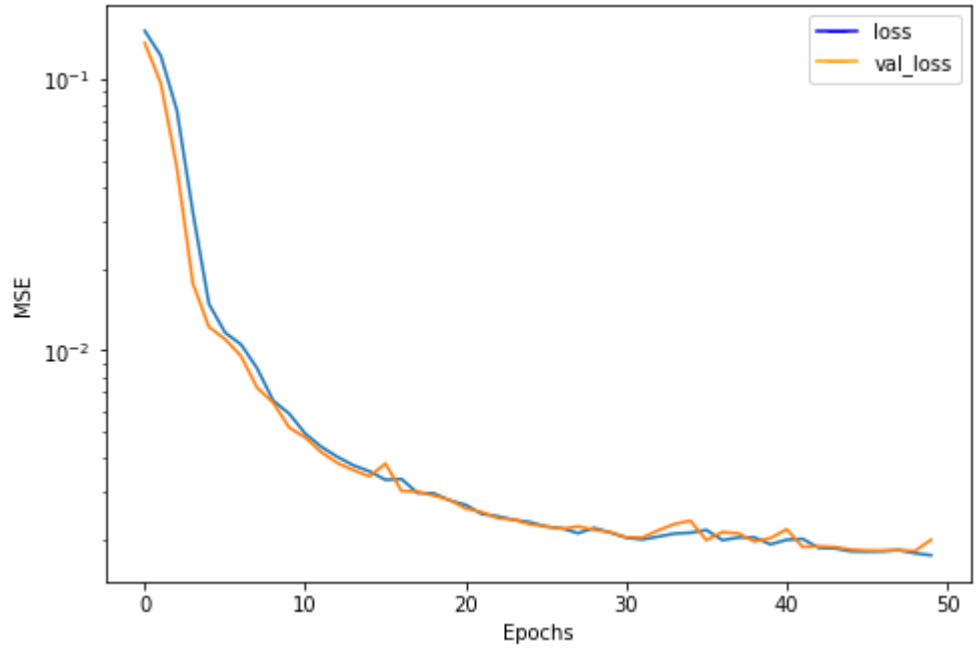


Fig. 55 Grafico loss di training e validation dell'autoencoder convoluzionale con il dataset dei cuori e rimozione di pixels

L'allenamento anche questa volta riporta lo stesso andamento.

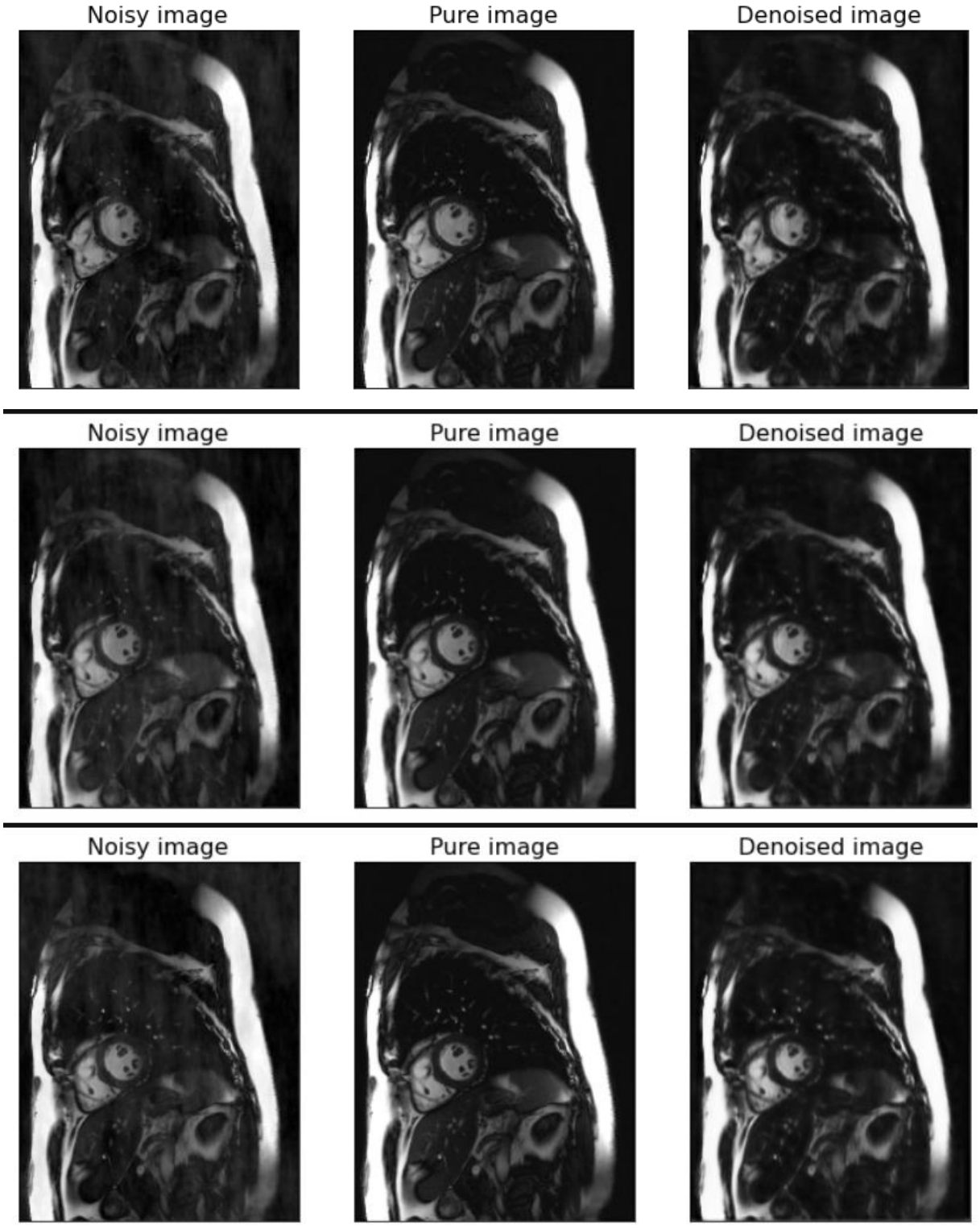


Fig. 56 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cuori con rimozione di 1 pixel ogni 20

In questo caso i risultati sono più positivi rispetto all'eliminazioni di intere righe.
Si può infatti osservare come i rumori presenti vengono rimossi piuttosto bene.
I disturbi che rendono l'immagine come se fosse sfocata vengono ridotti a tal punto da portare l'immagine in output dalla rete neurale ad essere molto simile all'originale.
Calcolando l'MSE di questo modello si ottiene un valore di 0.00185.

Avendo ottenuto dei buoni risultati con questo tipo di errore si procede aggiungendone per vedere come si comporta la rete.
Lo step successivo è di rimuovere 1 pixel ogni 8.

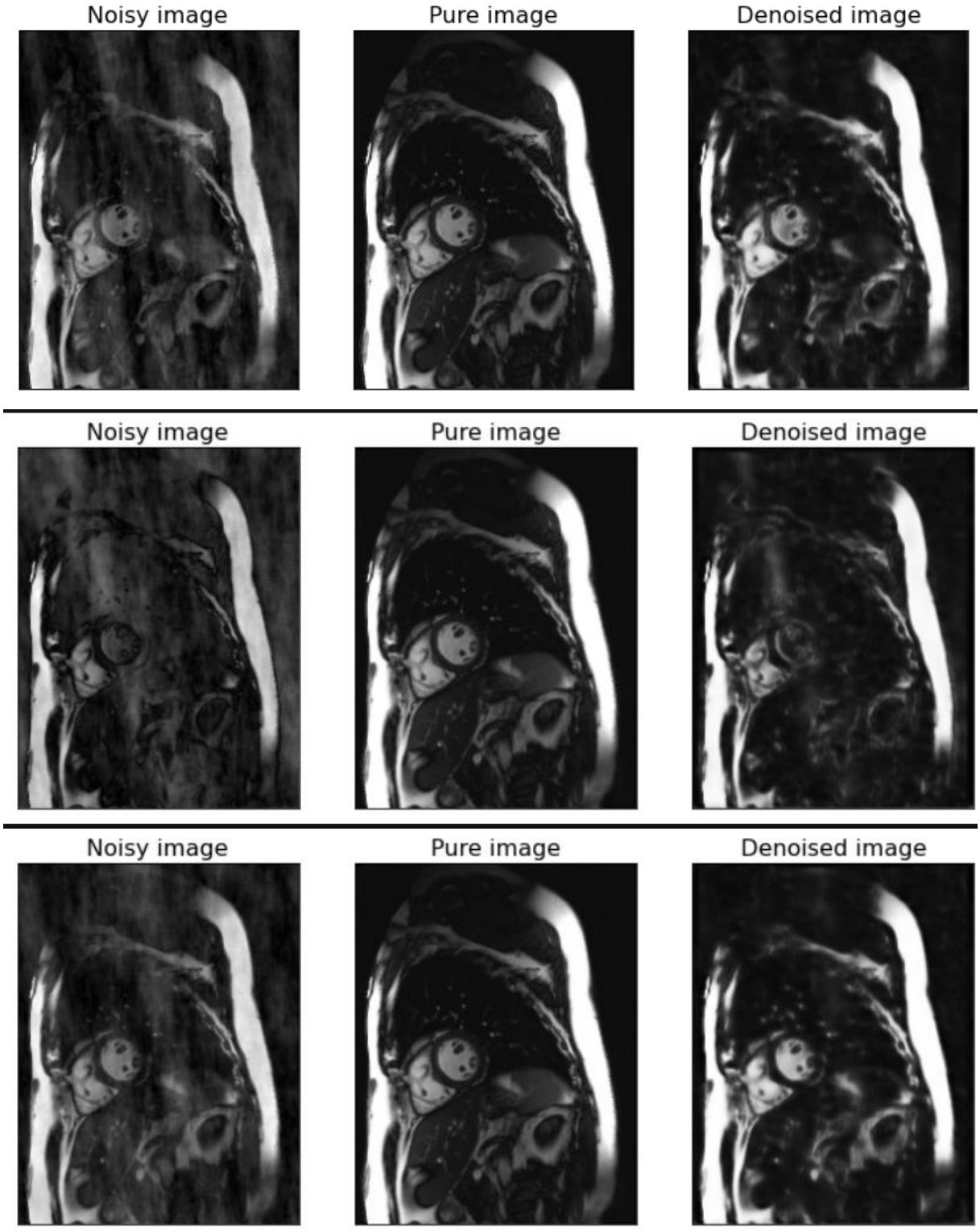
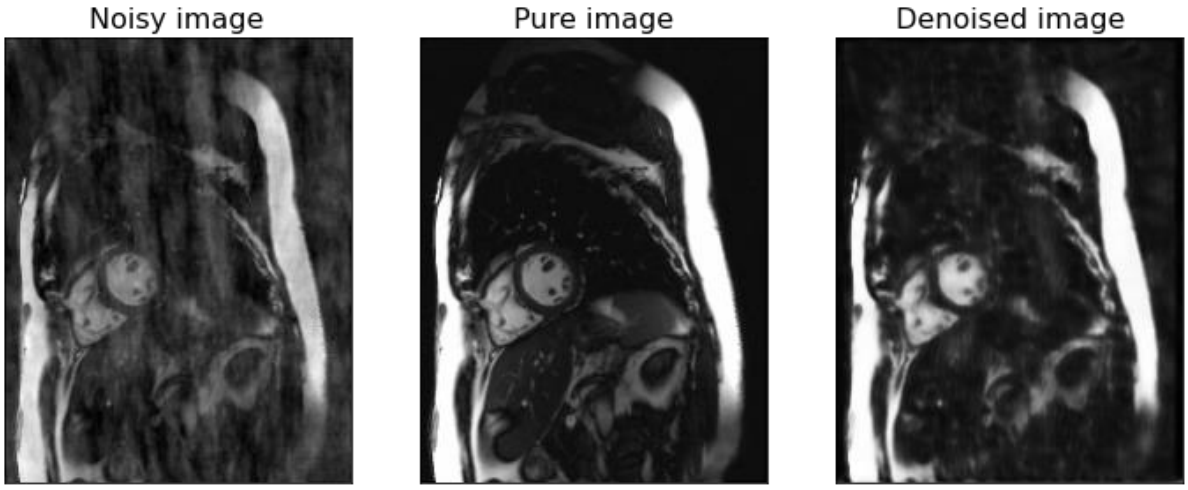


Fig. 57 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cuori con rimozione di 1 pixel ogni 8

In questo caso il rumore è aumentato molto, sono stati rimossi più del doppio dei pixels.

Similmente alla situazione precedente la sfocatura dell'immagine viene rimossa.
La differenza è che essendoci più rumore vengono persi dei dettagli importanti che in seguito la rete non riesce a recuperare.
Il lavoro dell'autoencoder è comunque buono dato che svolge il suo compito di rimuovere il rumore.
In questo caso l'MSE calcolato è di 0.00424.

Come ultimo test si può estremizzare ancora di più la situazione e vedere come si comporta la rete con 1 pixel rimosso ogni 4.



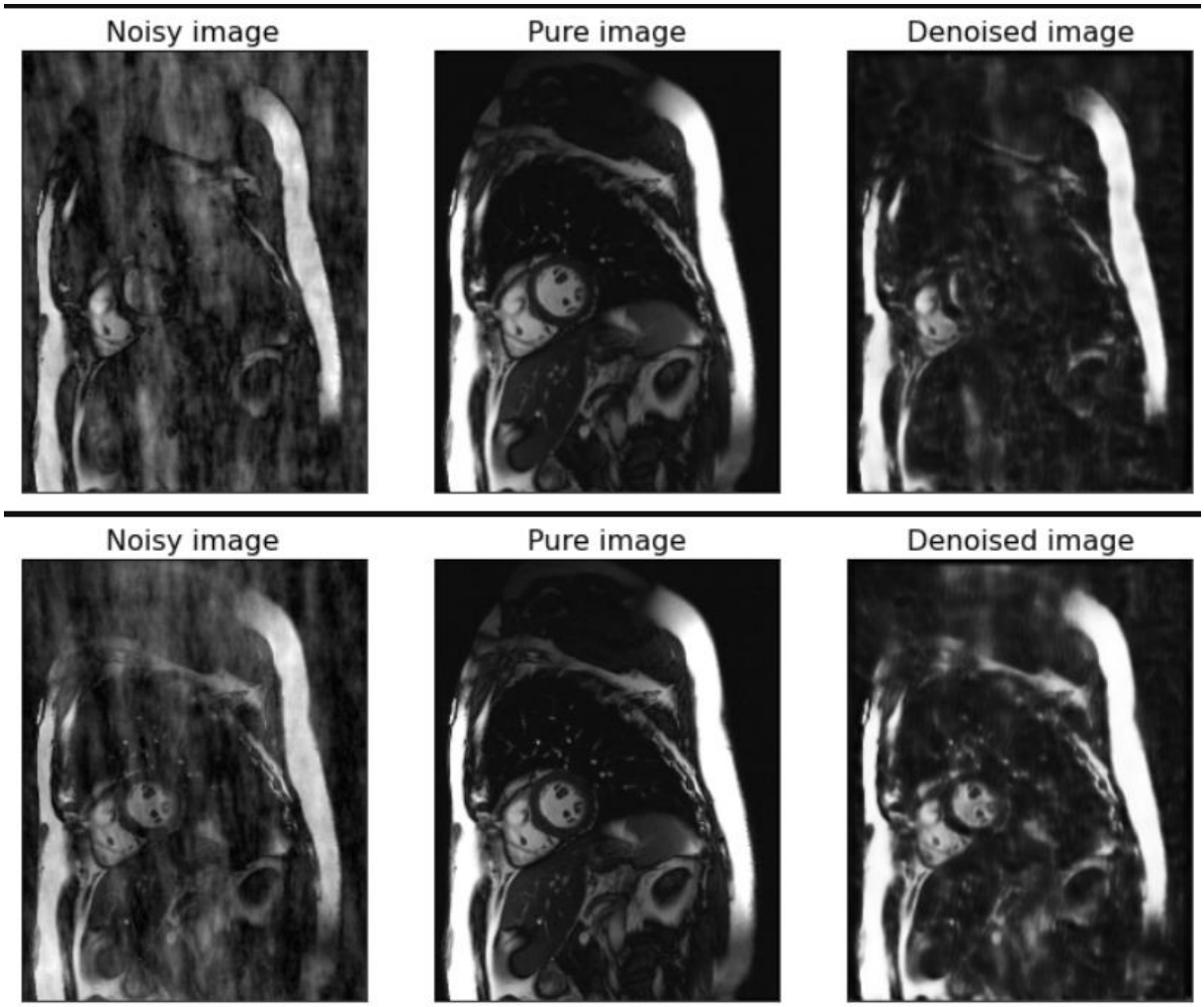


Fig. 58 Risultato del denoise effettuato dall'autoencoder convoluzionale su dataset dei cuori con rimozione di 1 pixel ogni 4

È stato ancora raddoppiata la quantità di pixels rimossi e infatti le immagini rumorose sono molto più rovinate e prive di dettagli.
Stesso discorso per il caso precedente, la rete riesce comunque a completare il suo obbiettivo di rimuovere il rumore.
Il fatto che le immagini sono così rovinate da nascondere la maggior parte dei dettagli non toglie niente al lavoro fatto dall'autoencoder che ricevendo immagini mai viste prima non può ricreare i dettagli mancanti.
L'MSE calcolato con questo modello è di 0.00657.

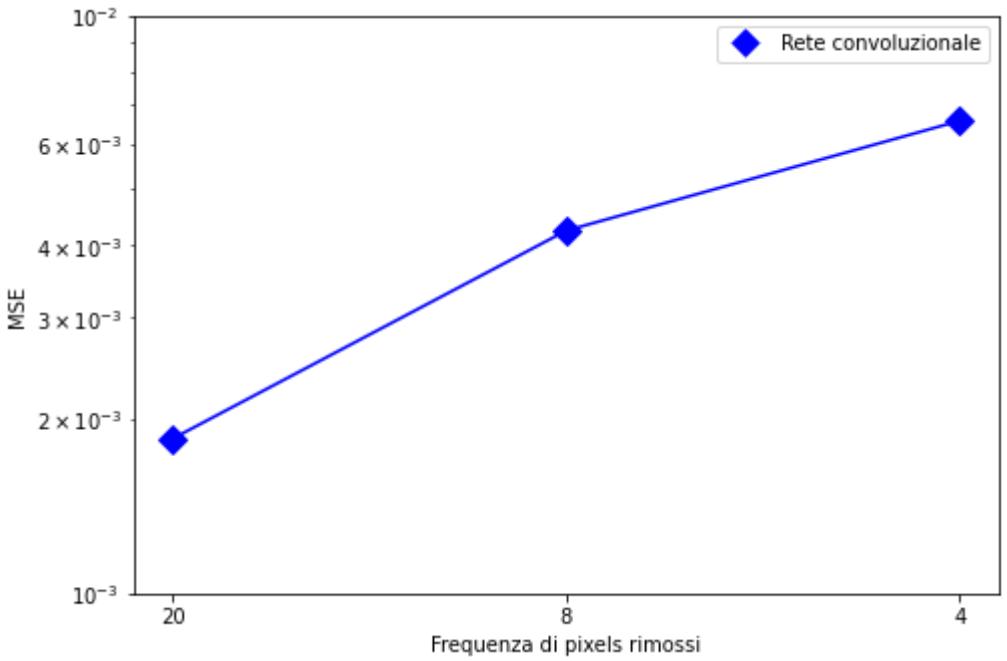


Fig. 59 Grafico delle fraquenza di pixels rimossi in relazione al MSE per il dataset dei cuori

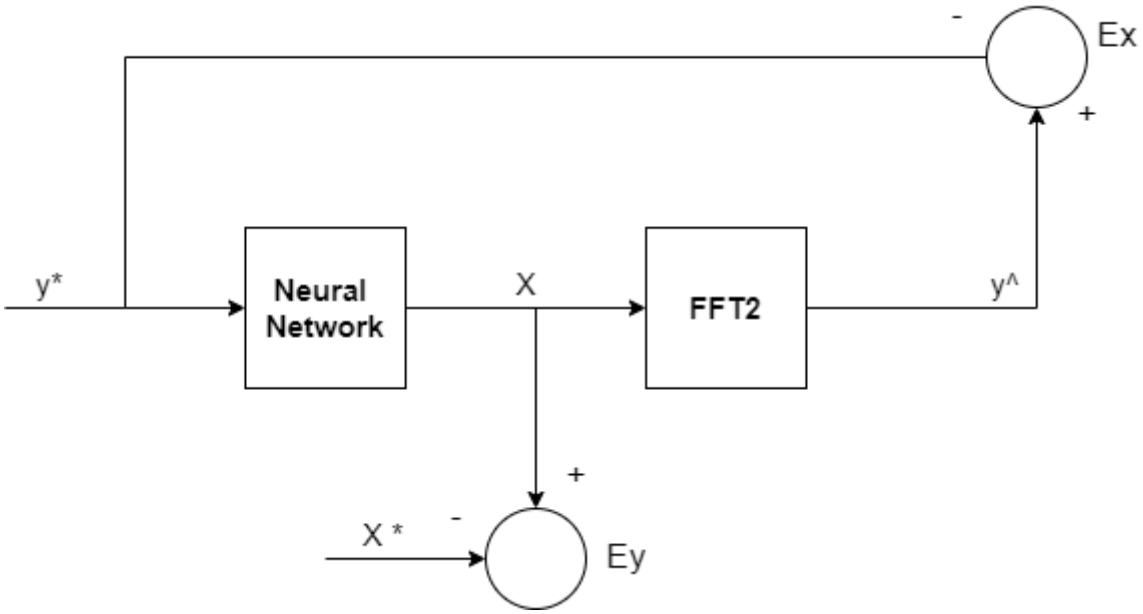
Il grafico dei MSE mostra l'aumento dell'errore con l'aumentare dei pixels rimossi.

4. Progetti futuri

In questo progetto sono state provate solo alcune delle idee per risolvere questo problema.
Il primo tentativo svolto è stato infatti quello di utilizzare un autoencoder, data la sua predisposizione al problema che ci siamo posti.
Il suo compito era quello di, preso in input un'immagine presente nel dominio delle immagini, togliere la maggior parte del rumore possibile.
Utilizzando diversi tipi di disturbi la rete si sarebbe allenata per mandare in output un'immagine più simile possibile all'originale.

Vi sono diversi tipi di test che si possono implementare per poter ottenere risultati migliori.
Uno di questi è chiaramente rendere la rete più potente, aggiungendo layers, come ad esempio, il BatchNormalization, che è stato aggiunto nell'ultima fase di progetto.
Un altro test che si sarebbe potuto svolgere è effettuale l'allenamento con K-Fold cross-validation..

Questo metodo consiste nel dividere il dataset in alcune parti a propria scelta e in seguito di prendere una di queste parte per la validation e le altre per il training. Questo procedimento avviene con tutte le parti, viene fatto questo per non allenare il modello sempre con gli stessi dati.



Infine, un'ultimo possibile interessante progetto futuro consisterebbe nello sviluppare un'architettura neurale simile a un autoencoder, ma che incorpori al suo interno la relazione tra dominio della frequenza e dominio delle immagini, oltre ad alcune regolarizzazioni ad-hoc per il problema MRI.

5. Conclusione

Questo progetto ha avuto due scopi principali: uno didattico e uno più rivolto alla ricerca.
La prima parte è stata rivolta alla raccolta delle informazioni principali.
Ho dovuto occuparmi della comprensione dei principi e funzionamento della risonanza magnetica e in seguito lo studio di reti neurali e autoencoder.
Un altro obiettivo didattico è stato quello di riuscire a gestire il progetto in maniera autonoma ed efficiente, con il supporto del relatore.
Oltre all'ampliamento delle mie conoscenze nell'ambito delle reti neurali sono stati svolti diversi test più in ambito di ricerca.
L'obiettivo principale di questo progetto era infatti di sviluppare una rete neurale capace di ridurre il rumore.
Ho svolto dapprima un preprocessing dei dati, nella quale ho scelto il formato delle immagini che funzionasse meglio con le reti neurali scelte.
In seguito dopo essermi occupato della creazione e scelta degli autoencoder ho effettuato diversi test allenando la rete con differenti dataset e tipi di errori.
I risultati sono stati molto altalenanti, variavano molto a dipendenza del tipo di errore.
Con errori più costanti come l'aggiunta di una normale o la rimozione di pixel distanti tra loro si sono ottenuti risultati più positivi.
Mentre con la rimozione di righe intere susseguivano perdite di dettagli dell'immagine maggiore e quindi risultati meno positivi.
Riassumendo il tutto ci sono comunque dei risultati soddisfacenti quindi l'obiettivo può dirsi raggiunto.
In questo campo non si otterranno quasi mai i risultati sperati ai primi tentativi, per questo servono molti più test di quelli fatti in questo progetto.
È positivo il fatto di avere la conferma di essere nella direzione giusta, essendo in un campo ancora piuttosto inesplorato.
In conclusione, nonostante i risultati e gli obiettivi raggiunti c'è ancora molto lavoro da fare.
Le idee e i test che si possono fare sono ancora tanti e si può puntare a raggiungere risultati ancora migliori.

Bibliografia

[1] M. Doneva, «Mathematical Models for Magnetic Resonance Imaging Reconstruction: An Overview of the Approaches, Problems, and Future Research Areas», *IEEE Signal Process. Mag.*, vol. 37, n. 1, pagg. 24–32, gen. 2020, doi: 10.1109/MSP.2019.2936964.
[2] A. S. Lundervold e A. Lundervold, «An overview of deep learning in medical imaging focusing on MRI», *Z. Für Med. Phys.*, vol. 29, n. 2, pagg. 102–127, mag. 2019, doi: 10.1016/j.zemedi.2018.11.002.
[3] L. Gondara, «Medical Image Denoising Using Convolutional Denoising Autoencoders», in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, dic. 2016, pagg. 241–246. doi: 10.1109/ICDMW.2016.0041.