Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il management

Super Resolution di immagini con reti neurali convoluzionali

Tesi di Laurea in Statistica Numerica

Relatore:
Chiar.ma Prof.ssa
Elena Loli Piccolomini

Correlatore:
Dottore
Pasquale Cascarano

Presentata da:
Elena Polini

I Sessione
Anno Accademico 2018-2019

Introduzione

La ricerca rivolta alla scoperta di nuove tecniche di elaborazione e rielaborazione di immagini è in continua evoluzione, ciò va di pari passo sia con l'avanzare del progresso tecnologico sia con la crescente esigenza di fruire di contenuti in formato foto e video di alta qualità. La richiesta di immagini ad alta definizione non sempre trova riscontro positivo: varie possono essere le cause, tra queste le limitazioni dovute alla banda di trasmissione, al rumore o ad altre cattive condizioni di acquisizione, alla bassa qualità degli strumenti fotografici, ecc. Il dominio del problema è sicuramente ampio, si pensi ad esempio alle immagini di videosorveglianza, a quelle mediche, alle riprese aeree per scopi di mappatura geografica e in generale a tutti gli ambiti in cui le immagini vengono acquisite a scopo di riconoscimento di dettagli o in cui subiscono ridimensionamenti notevoli. Per risoluzione si intende, in termini tecnici, la densità di pixel contenuti in un'immagine. Un'immagine ad alta risoluzione avrà un numero di pixel maggiore rispetto alla sua corrispettiva a bassa risoluzione. In termini visivi, aumentare la risoluzione di un'immagine significa migliorarne la definizione dei particolari, specialmente delle linee e dei contorni, per ottenere un risultato quanto più nitido e simile alla realtà. Le tecniche proposte nel tempo per risolvere il problema del miglioramento di risoluzione sono molteplici, diverse fra loro e in continuo aggiornamento. Lo scopo di questa tesi è quello di fornire una panoramica generale sul problema di super-risoluzione e sugli approcci possibili, e di presentare dei modelli di soluzione al problema basati sull'uso di reti neurali artificiali. Nel primo capitolo verrà illustrato il concetto di risoluzione e presentato il problema

di super-risoluzione, insieme alle tecniche ad oggi più comuni per risolverlo. Il secondo capitolo sarà dedicato alle reti neurali, in particolare a quelle convoluzionali (CNN), delle quali saranno descritti struttura, funzionamento e i principali parametri da tenere in considerazione per la loro costruzione. In seguito verrà presentata nel dettaglio la rete Very-Deep Super Resolution (VDSR) nata appositamente per risolvere il problema di super-risoluzione, e ad essa saranno proposti modelli alternativi che andranno a modificarne la composizione interna. Il terzo ed ultimo capitolo presenta i risultati ottenuti durante la fase sperimentale, mettendo a confronto le performance ottenute per le reti selezionate.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

La super risoluzione

In questo primo capitolo verrà presentato nel dettaglio il problema di super risoluzione, insieme a diverse categorie di approcci per affrontarlo. Nella parte conclusiva sono contenuti cenni alla bibliografia presa in analisi per questo lavoro, elencando i modelli di deep learning ad oggi più rilevanti e mettendoli a confronto in modo da avere un punto di riferimento per la discussione dei capitoli successivi.

1.1 Risoluzione di immagini

Il termine "risoluzione" indica il rapporto fra il numero di pixel contenuti in un'immagine e le sue dimensioni e si misura in pixel per pollice (PPI) per le immagini digitali, e in punti per pollice (DPI) per quelle stampate. Si tratta di una caratteristica fortemente determinante della qualità di un'immagine: immagini ad alta risoluzione hanno infatti un'elevata densità di pixel e di conseguenza una migliore definizione dei dettagli.

Con l'avanzare del progresso tecnologico, sempre più ambiti e discipline necessitano di disporre di immagini di alta qualità su cui basare il proprio lavoro, per questo nel tempo si sono sviluppate molteplici tecniche finalizzate ad ottenere condizioni ottimali di risoluzione. La scelta più semplice consisterebbe nell'impiego di una strumentazione ad alta precisione che permettesse

di aumentare il numero di pixel dell'immagine già in fase di acquisizione. Tali strumenti dispongono di sensori con superfici altamente sensibili alle radiazioni luminose, anche appartenenti a lunghezze d'onda che vanno oltre il campo del visibile. Non sempre però, pur disponendo di certa strumentazione, si creano le condizioni tecniche ottimali per ricavare immagini della qualità desiderata: ad esempio possono verificarsi cali energetici che compromettono l'alimentazione dei dispositivi, condizioni ambientali ostili (si pensi all'acquisizione di immagini satellitari o di videosorveglianza) o limiti nella banda di trasmissione dei dati. Gli alti costi di questo tipo di supporti e i vari impedimenti tecnici hanno reso necessario trovare un modo per ripristinare la risoluzione ideale delle immagini in fase di post produzione: le tecniche di super risoluzione ambiscono proprio a minimizzare i costi di acquisizione mediante algoritmi di elaborazione che agiscono direttamente sulle immagini per restaurare i dettagli persi.

1.2 Definizione del problema

Il problema di super risoluzione consiste nell'ottenere un'immagine ad alta risoluzione (HR) a partire da una o più immagini a bassa risoluzione (LR). Una bassa risoluzione può essere dovuta all'acquisizione mediante strumenti di scarsa precisione, o a operazioni di ridimensionamento o ricampionamento svolte a posteriori sull'immagine: in tutti questi casi si ha una carenza dal punto di vista della qualità e una perdita di informazioni che vanno ricostruite. A seconda del numero di immagini a bassa risoluzione date in input, si distingue fra:

- Single-image super resolution (SISR): prevede un'unica immagine a bassa risoluzione in input dalla quale ottenere la corrispettiva in alta risoluzione mediante algoritmi che ricostruiscono i pixel mancanti limitando l'inserimento di artefatti. Per far ciò modelli di SISR sfruttano feature ricorrenti all'interno della stessa immagine oppure apprendono pattern associativi basandosi su database esterni di immagini a bassa e

alta risoluzione. Un ambito tipico per problemi di questo tipo è quello medico, nel quale spesso non è possibile disporre di più di un'immagine diagnostica per via dei costi o dei rischi per la salute del paziente.

- Multi-image super resolution (MISR): l'immagine ad alta risoluzione viene ricostruita a partire da un set di immagini raffiguranti lo stesso soggetto ma con variazioni, anche lievi, di inquadratura o di definizione dei dettagli. Ogni immagine a bassa risoluzione impone un insieme di constraint non lineari sull'immagine finale. Gli algoritmi di super risoluzione agiscono quindi andando a sommare le differenze fra le immagini LR in modo da ottenere un'immagine ad alta risoluzione che risulti il più simile possibile a quella reale.

Purtroppo raramente si dispone di un sufficiente numero di immagini multiple per poter applicare tecniche di MISR in modo efficiente. Per questo motivo gli algoritmi di SISR sono generalmente i più utilizzati e sono anche quelli presi in considerazione in questa tesi.

1.3 Algoritmi di single image super-resolution

Il problema di SISR rientra nella classe dei problemi mal posti: una singola immagine a bassa risoluzione data in input può derivare da una serie di possibili immagini ad alta risoluzione, a seconda dei parametri e metodologie utilizzati per la ricostruzione della qualità. Per questo motivo sono stati proposti numerosi modelli e algoritmi che possono essere classificati in tre principali categorie: interpolation based, reconstruction based e learning based.

1.3.1 Interpolation based

Gli approcci interpolation based fanno parte della famiglia degli algoritmi predittivi. Essi sono i più semplici e veloci ma mancano di accuratezza in quanto basati sull'applicazione di formule matematiche predefinite senza far riferimento a dataset preaddestrati.

Gli algoritmi di interpolazione utilizzano dati noti per stimare valori in punti sconosciuti. In termini matematici, dato un insieme di coppie di valori $(x_1, y_1), ..., (x_n, y_n)$, si tratta di trovare la funzione f tale che:

$$f(x_i) = y_i$$

per i = 1,...,n.

Nel concreto, a partire da un campione di dati è possibile determinare una funzione che passi per tutti i punti e poi utilizzare tale funzione per determinare nuove coppie di valori.

Nel campo dell'imaging questa tecnica consiste nell'ottenere per ogni pixel sconosciuto di un'immagine a bassa risoluzione la migliore approssimazione del colore, dell'intensità e dei dettagli geometrici degli oggetti basandosi sui valori dei pixel limitrofi, senza introdurre artefatti. Maggiore è il numero di pixel che vengono considerati ad ogni passo dell'interpolazione per stimare quelli incogniti, migliore sarà la risoluzione dell'immagine HR ottenuta. Si possono distinguere due categorie di algoritmi di interpolazione:

- Algortimi adattativi: prevedono l'applicazione di versioni diverse dello stesso algoritmo in base all'area dell'immagine che si sta interpolando, tenendo conto della struttura locale e delle variazioni di frequenza tra le sue parti. Questa categoria di algoritmi richiede l'impiego di maggiori risorse hardware ed ha un costo computazionale elevato. Per questo la maggior parte dei problemi reali viene risolta utilizzando algoritmi non adattativi.

- Algoritmi non adattativi: lo stesso algoritmo è applicato uniformemente su tutta l'immagine. Il difetto principale di questa categoria

di algoritmi è l'introduzione di artefatti (linee spezzate, blurring, ecc) in corrispondenza degli edge dell'immagine, cioè dei punti a più alta frequenza. Appartengono a questa categoria due tra gli algoritmi di interpolazione più noti ed utilizzati: l'interpolazione nearest neighbor e quella bicubica.

L'interpolazione nearest neighbor (Figura 1.1) assegna al pixel da interpolare lo stesso valore del pixel più vicino, ignorando tutti gli altri.

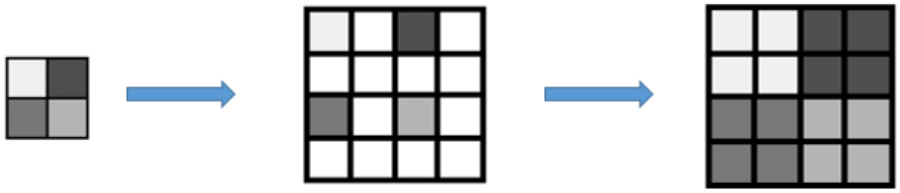


Figura 1.1: Interpolazione nearest neighbor

È sicuramente l'algoritmo più semplice da applicare in quanto non fa uso di regole complicate per determinare il valore statisticamente più corretto da assegnare al nuovo pixel, ma si limita ad individuare il pixel più vicino per copiarne il valore.

I difetti di questo algoritmo stanno nella qualità del risultato finale: per immagini con molti dettagli (ad esempio fotografie) specialmente contenenti tratti curvi o obliqui, l'output presenta linee frastagliate e l'immagine risulta "a blocchi". Questo fenomeno viene chiamato "aliasing" e può essere minimizzato semplicemente adottando un metodo di interpolazione che preveda l'assegnazione di valori intermedi rispetto ai pixel adiacenti.

L'interpolazione bicubica (Figura 1.2), ad esempio, è un'ottima alternativa in quanto considera una matrice 7x7 costruita intorno al pixel da interpolare, e di questa i valori dei 16 pixel ad esso più vicini. In questo modo il risultato finale dipenderà da più osservazioni circostanti e restituirà un valore medio più prossimo a quello reale, riducendo in modo sostanziale il problema dell'aliasing. Naturalmente, più vicini sono i pixel rispetto a quello centrale, maggiore sarà il loro peso sul risultato.

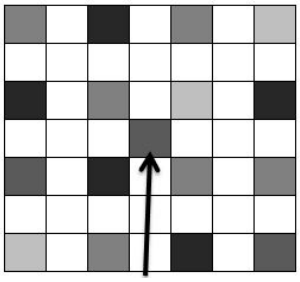


Figura 1.2: Interpolazione bicubica

Questo algoritmo è considerato il migliore metodo di interpolazione dal punto di vista della qualità e del tempo impiegato, per questo motivo è ad oggi lo standard utilizzato nei programmi commerciali di editing, uno su tutti Adobe Photoshop.

1.3.2 Reconstruction based

Gli algoritmi di tipo reconstruction based utilizzano dati già noti per restringere il campo delle possibili soluzioni, generando immagini più dettagliate rispetto ai risultati dei metodi di interpolazione. Il principio su cui si basano è che esista una forte relazione fra l'immagine LR e quella HR, e che tale rapporto possa essere riassunto nell'equazione:

$$y = DHx$$

dove y è l'immagine a bassa risoluzione, x quella ad alta risoluzione, mentre D e H sono due matrici, rispettivamente quella di sampling (o sottocampionamento) e quella di blur. Lo scopo della super risoluzione è far sì che la differenza tra x e y sia ridotta il più possibile, e questo si può tradurre in un problema di minimo. Per i modelli reconstruction based tale problema ha la forma:

$$argmin_x \|y - DHx\|_2^2$$

ovvero il minimo al variare di x della norma 2 del residuo tra l'immagine LR e quella ad alta risoluzione. Poichè le matrici D e H sono spesso mal condizio-

nate, le soluzioni ottenute dalla semplice applicazione di questa formula non sono attendibili. È necessario perciò introdurre un termine di regolarizzazione: lo scopo è quello di imporre dei limiti allo spazio delle possibili soluzioni, fornendo informazioni "a priori" riguardanti la soluzione finale attesa. La formula viene quindi aggiornata:

$$argmin_x \|y - DHx\|_2^2 + \lambda R(x)$$

dove R(x) è il termine di regolarizzazione, mentre λ è un parametro che determina quanto peso dare al termine R(x). Gli algoritmi reconstuction based sono vari e si differenziano fra loro per la scelta di diversi termini di regolarizzazione.

1.3.3 Learning based

Gli algoritmi basati sull'apprendimento utilizzano tecniche derivate dal machine learning per ottenere dati statistici che leghino immagini a bassa risoluzione alle corrispettive immagini ad alta risoluzione. Per far ciò prevedono una fase antecedente di training basata su esempi e risultati noti in modo da stabilire pattern associativi tra i dati in input e quelli in output. Rispetto ai precedenti modelli, la super risoluzione per apprendimento risulta migliore in termini di trade-off fra tempi di computazione e qualità dei risultati. La parte centrale di questo lavoro si concentra su quest'ultimo tipo di approccio al problema, scegliendo tra gli algoritmi di apprendimento quelli basati sul deep learning e in particolare sull'utilizzo delle reti neurali, dato il sempre crescente interesse su queste ultime e gli ottimi risultati ottenibili.

In tempi recenti sono state proposte diverse reti per la soluzione dei problemi di SISR, di seguito verranno elencate alcune delle soluzioni più note tra quelle presentate in un recente articolo[4] (agosto 2018) da alcuni ricercatori della Tsinghua University della Cina e del Universtity College di Londra.

La prima architettura proposta è stata la Super-Resolution CNN (SRCNN) [14], una rete composta da tre layer che considera solo i componenti di luminosità dell'immagine e applica tre trasformazioni non lineari sui pixel:

estrazione di patches, mappatura non lineare e ricostruzione. I tre filtri conisderati sono di dimensioni 5x5, 1x1 e 9x9 e la loss function scelta è l'errore quadratico medio. La semplicità di questa rete e la sua capacità di operare su grandi quantità di dati ha fatto sì che per molto tempo venisse considerata il metodo migliore per risolvere il problema di SISR, a scapito dei modelli basati su interpolazione e ricostruzione. Tuttavia alcuni dei suoi limiti hanno portato alla creazione di reti alternative che successivamente si sono rilevate molto più funzionali rispetto alla SRCNN.

La prima alternativa alla SRCNN, chiamata FSRCNN[15], ha permesso di modificare l'input accettato dalla rete. Il modello SRCNN, infatti, opera su immagini LR precedentemente sottoposte a un processo di interpolazione il quale ne riduce (downsampling) e poi aumenta (upsampling) le dimensioni. Partire da immagini interpolate può essere sconveniente sia dal punto di vista dei tempi di computazione sia perchè le alterazioni apportate ai dettagli dagli algoritmi di interpolazione possono comportare errori di stima aggiuntivi sul risultato finale. Per questi motivi, il modello FSCRNN accetta in input immagini LR ed esegue le operazioni di down e upsampling direttamente all'interno della rete, mediante layer convoluzionali che implementano pooling e stride per aumentare le dimensioni dell'immagine e un layer di deconvoluzione per ritornare alle dimensioni originali. Il layer di deconvoluzione è posto alla fine della rete ed utilizza un operatore di interpolazione (solitamente nearest neighbor) seguito da uno di convoluzione per ripristinare i dettagli ad alta risoluzione dell'immagine. A parità di dataset utilizzato per il training, l'architettura FSRCNN risulta più accurata dal punto di vista dei risultati e permette di ridurre notevolmente i tempi di computazione, dal momento che l'incremento di risoluzione viene spostato nella parte finale della rete.

Un'alternativa per quanto riguarda le dimensioni della rete è stata data dal modello Very Deep Super-Resolution (VDSR)[16, 20], che costituirà il soggetto centrale di questa tesi. Il difetto principale della rete SRCNN risiede nella sua semplicità: è stato dimostrato come incrementando il numero di layer sia possibile ottenere risultati migliori. La rete VDSR è una

rete denominata “very deep”, in quanto composta da ben 20 layer, che ha riportato risultati notevoli nell'aumento della performance. Essa inoltre ha il merito di avere introdotto l'addestramento di modelli con diversi fattori di scala in modo simultaneo e di aver implementato un apprendimento basato sui residui, che velocizza ulteriormente la computazione.

Una variante della rete VDSR, proposta dai suoi stessi autori, è la Deeply-Recursive Convolutional Network (DRCN)[17]. Per poter ridurre il numero di parametri in gioco durante l'addestramento, la rete DCRN elimina parte dei layer della VDSR utilizzando lo stesso kernel convoluzionale per 16 volte in modo ricorsivo. Ad ogni passo della ricorsione, il risultato ottenuto viene passato all'ultimo layer della rete responsabile di ricostruire l'output finale, che sarà dato dall'unione dei 16 risultati intermedi. Questo tipo di strategia viene definita “multi supervisionata” e ha dimostrato di apportare miglioramenti sia nella fase di backpropagation (evitando variazioni indesiderate del valore del gradiente) sia nella qualità finale del risultato (l'immagine HR finale è data dalla somma di molteplici immagini HR intermedie).

Altri modelli degni di nota sono l'Enhanced Deep Super-Resolution (EDSR) [18] e il Multi-scale Deep Super-Resolution (MDSR), composti rispettivamente da 69 e 166 layer. Entrambi hanno permesso di ottenere ottime performance grazie all'impiego di un numero elevato di layer e all'implementazione di un apprendimento multi scala. A questo proposito, una novità introdotta è stata quella di eseguire la fase di training su una rete preaddestrata con un fattore di scala inferiore a quello corrente, tentativo che ha comportato un notevole miglioramento nella velocità di apprendimento e nella bontà dei risultati.

Un ultimo modello da citare è il Residual Dense Network (RDN)[19], che combina una struttura di tipo deep (149 layer totali) con l'apprendimento basato sui residui. Inoltre questa rete ha introdotto un fattore di densità, impiegando un'architettura fully connected: la rete è suddivisa in blocchi all'interno di ognuno dei quali i layer convoluzionali sono densamente connessi fra loro (ogni nodo presenta connessioni con tutti gli altri). Prima di arrivare

al layer di ricostruzione finale, gli output di tutti i blocchi vengono sommati mediante residual learning a formare un'unica immagine.

Nella tabella della Figura 1.3, tratta direttamente dall'articolo di riferimento[4], sono mostrati i risultati numerici ottenuti da diverse reti, insieme a dettagli sulla loro struttura.

Models	PSNR/SSIM(×4)	Train data	Parameters	Layers
SRCNN_EX	30.49/0.8628	ImageNet subset	57K	3
ESPCN	30.90/-	ImageNet subset	20K	3
FSRCNN	30.71/0.8657	G100+Yang91	12K	5
VDSR	31.35/0.8838	G200+Yang91	665K	20
DRCN	31.53/0.8838	Yang91	1.77M(recursive)	20
DRRN	31.68/0.8888	G200+Yang91	297K(recursive)	54
LapSRN	31.54/0.8855	G200+Yang91	812K	24
SRResNet	32.05/0.9019	ImageNet subset	1.5M	37
Memnet	31.74/0.8893	G200+Yang91	677K(recursive)	80
RDN	32.61/0.9003	DIV2K	21.9M	149
EDSR	32.62/0.8984	DIV2K	43M	69
MDSR	32.60/0.8982	DIV2K	8M	166
DBPN	32.47/0.898	DIV2K+Flickr+ ImageNetb subset	10M	46

Figura 1.3: Confronto fra reti di SISR

È evidente come al crescere della profondità e del numero di parametri, la performance finale migliori, anche se dopo un certo valore il fattore di crescita si stabilizza. Per questo attualmente si punta sempre più su modelli con quantità di parametri contenute per risparmiare sui costi di computazione, piuttosto che introdurre pesanti complicazioni strutturali che inciderebbero di poco sul miglioramento del risultato finale.

Nella scelta sul tipo di rete da analizzare nel dettaglio in questa tesi, si è tenuto conto di due fattori sostanziali: l'efficienza finale e il costo per la fase di training. La rete VDSR è sembrata un buon compromesso fra questi due aspetti, dato il numero di layer consistente ma comunque compatibile con l'addestramento su una macchina comune in tempi accettabili.

Capitolo 2

Le reti neurali

In questo capitolo verranno presentati la struttura e il funzionamento delle reti neurali e, nel dettaglio, quelli delle reti convoluzionali. Quindi sarà descritta la rete VDSR selezionata per il problema di super risoluzione, mettendone in risalto le differenze con altri modelli esistenti. Nell'ultima parte verranno proposte alternative alla struttura base della rete che corrispondono a tentativi di migliorarne la performance eseguiti nella fase sperimentale di questo lavoro.

2.1 Struttura di una rete neurale

Le reti neurali artificiali (ANN) sono modelli di calcolo matematico nati intorno agli anni '40 del secolo scorso come riproduzione delle reti neurali biologiche. Esse svolgono funzioni di predizione ed elaborazione su insiemi estesi di dati basandosi sull'ipotesi dell'esistenza di pattern e interconnessioni fra le informazioni oggetto di studio. Il principale ambito di applicazione delle reti neurali è il machine learning, in quanto esse permettono di implementare un sistema di apprendimento da parte di una macchina e di sfruttarlo per il riconoscimento o la classificazione di dati in problemi futuri. In anni recenti il Dipartimento di Ingegneria dell'Università di Hong-Kong ha dimostrato come

una rete neurale possa essere usata per creare una mappa tra le immagini LR e quelle HR per risolvere il problema di super risoluzione in modo efficiente.

Lo schema base di una rete neurale consiste in un numero elevato di neuroni, che costituiscono le unità computazionali di base, collegati tra loro a formare una struttura non lineare. Ogni neurone è di per sè dotato di capacità di elaborazione e può trasformare un dato in input in un output tramite l'applicazione di una regola matematica chiamata "funzione di attivazione". Per poter applicare tale funzione, il valore in entrata del neurone deve superare un certo valore, detto "soglia di attivazione", il quale viene prefissato nel momento della costruzione della rete; se il valore di soglia non viene superato, il nodo rimarrà inattivo.

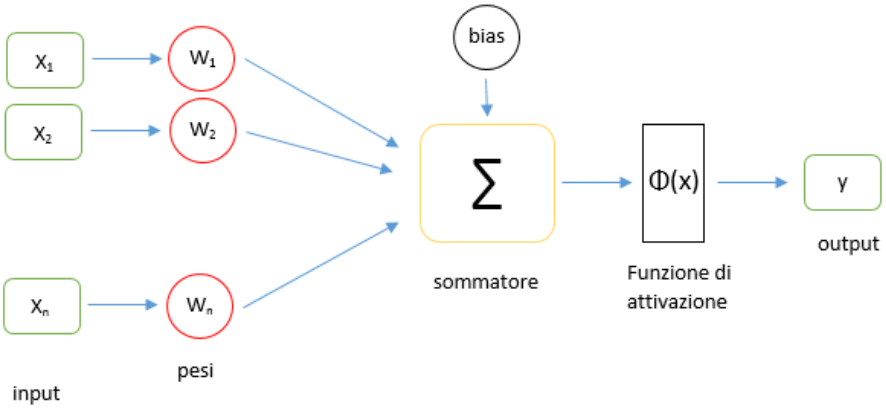


Figura 2.1: Modello di neurone artificiale

Ogni neurone all'interno di una rete è collegato agli altri tramite delle connessioni. Esse sono orientate e pesate (ad ogni connessione è assegnato un valore numerico definito "peso"), e ad ogni neurone corrispondono una connessione in entrata e una o più connessioni in uscita. La funzione di attivazione permette la trasmissione di segnali tra neuroni collegati, la sua formula base può essere scritta come:

$$output = attivazione \sum (input * pesi) + bias$$

Applicare una funzione di questo tipo significa quindi fare una sommatoria dei prodotti fra i valori in input e i pesi corrispondenti ed aggiungere ad essa un valore costante. Quest'ultimo, chiamato "bias", permette di traslare la funzione in modo da poter ottenere valori di output appartenenti ad un range più esteso. Più elevato sarà il peso del bias, più esso influirà sullo spostamento del risultato finale, mentre per bias tendenti a 0 l'output dipenderà solamente dai valori di input e dei pesi.

Un certo numero di neuroni che risiedono sullo stesso livello nella rete formano un "layer". Una rete costituita da due soli layer, quello di input e quello di output, viene chiamata "percettrone". Questa particolare struttura permette di eseguire operazioni di approssimazione di funzioni e di classificazione, ma solamente nel caso in cui le classi di output siano linearmente separabili. Data la semplicità e le limitazioni del percettrone, soprattutto legate al limitato campo di applicazione, sono state introdotte reti neurali più complesse, chiamate "multistrato". Esse presentano dei layer intermedi tra lo strato di input e quello di output, detti "hidden layer", in numero variabile. La rete proposta in questa tesi, ad esempio, viene definita "very deep" in quanto costituita da ben 18 layer nascosti. A differenza del percettrone, le reti multistrato permettono di agire su un qualunque insieme di dati, anche non linearmente separabili, e risultano essere più efficienti dal punto di vista della precisione dei risultati, sebbene in molti casi con una notevole perdita in termini di tempo necessario per l'addestramento.

Tra le reti multistrato è possibile distinguere tra reti feedforward e reti ricorrenti:

- le reti feedforward possono essere rappresentate come un grafo orientato aciclico, nel quale le connessioni fra i nodi seguono una stessa direzione e propagano il segnale in avanti, partendo dallo strato di input verso quello di output passando per i layer nascosti. Sono il tipo di rete più diffusa e utilizzata, anche per la semplicità della fase di addestramento.

- le reti ricorrenti, dette anche feedback, prevedono la presenza di cicli, ovvero di connessioni che propagano il segnale sia in avanti che indie-

tro tra i neuroni. Ogni nodo della rete tiene quindi conto del segnale proveniente da nodi dello stesso layer, di quelli precedenti e successivi. Questa caratteristica rende la rete ricorrente molto dinamica e permette di mantenere una "memoria" dello stato interno della rete, d'altra parte tuttavia l'apprendimento risulta essere molto complicato e dispendioso.

In questo lavoro saranno prese in considerazione solo le reti di tipo feedforward.

2.2 L'appprendimento

La costruzione di una rete neurale si definisce "apprendimento": si tratta di un processo adattivo e incrementale che consiste nella modifica dei pesi (e dei bias, se presenti) delle connessioni al fine di minimizzare la differenza di valori tra l'output atteso e quello ottenuto. Questa differenza viene definita "loss function", o funzione di errore, e in quanto tale il suo valore minimo potrà essere calcolato mediante l'uso delle derivate. Prima di descrivere nel dettaglio i passi dell'algoritmo di apprendimento è necessario definirne il punto di partenza, ovvero i dati utilizzati per addestrare la rete. Tali dati sono raccolti in un dataset di dimensioni variabili, chiamato training set, che registra coppie input/output già correttamente classificate. A seconda dei dati contenuti nel training set è possibile distinguere almeno 3 tipi diversi di apprendimento:

- Supervisionato: il più comune e anche quello preso in considerazione in questo elaborato, viene applicato quando alla rete vengono forniti sia campioni di input che di output in modo tale che possa adattare i suoi parametri e stabilire un nesso fra i due insiemi
- Non supervisionato: i campioni di output non sono noti lasciando che la rete apprenda in modo autonomo cercando una logica di classificazione a partire dai soli dati di input

- Per rinforzo: ispirato alla teoria di apprendimento per premio/punizione, consiste in una serie di tentativi di associazione tra input e output da parte della rete, a ognuno dei quali corrisponde un responso negativo o positivo che permette all'algoritmo di determinare per quale direzione proseguire.

Una volta definito il training set, il processo di apprendimento si sviluppa in una serie di fasi, così riassumibili:

1. Alle connessioni fra i nodi della rete vengono inizialmente assegnati dei pesi casuali. Vengono quindi sottoposti alla rete dei campioni di dati in input. Per ogni input trasmesso in avanti fino all'ultimo layer la rete elaborerà un output corrispondente applicando la funzione di attivazione scelta. Questa prima fase è anche detta "di forward", in quanto il segnale si propaga in avanti dal primo all'ultimo strato.

2. Una volta completata la fase forward, viene scelta una funzione di errore (loss function) che calcoli la differenza tra l'output ottenuto e quello atteso per ogni input. Lo scopo dell'apprendimento è minimizzare questa funzione di errore e per fare ciò è necessario calcolarne la derivata parziale rispetto a ciascuno dei pesi. Questa funzione derivata viene anche detta "gradiente".

3. Per poter calcolare il gradiente associato ad ogni nodo della rete si applica l'algoritmo di backpropagation: esso permette di propagare l'errore all'indietro verso lo strato di input. Sostanzialmente viene calcolato il contributo di ogni peso all'errore finale di computazione dell'output.

4. A questo punto è possibile applicare un algoritmo di ottimizzazione per aggiustare i pesi della rete in base al valore del gradiente calcolato per ogni nodo. La maggioranza degli algoritmi di ottimizzazione si basa sul metodo della discesa del gradiente, che può essere riassunto dalla formula:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\delta E}{\delta w_{ij}}(t)$$

il peso della connessione tra i neuroni i e j al passo t+1 è dato dal peso della connessione al passo t diminuito del gradiente (la derivata parziale). Il gradiente è moltiplicato per un coefficiente η, chiamato "fattore di apprendimento" o "learning rate".

Stabilire un valore appropriato di η è fondamentale, in quanto determina quanto l'errore influisca sulla modifica dei pesi: un learing rate basso può portare la rete a convergere troppo lentamente, mentre uno troppo alto rischia di modificare in modo sconsiderato il valore dei pesi (fenomeno chiamato "esplosione del gradiente"), allontanando la soluzione dal valore di minimo cercato. In alcuni casi il learning rate non rimane costante per tutta la fase di apprendimento, ma assume valori più alti all'inizio per poi decrescere con l'avvicinarsi al valore minimo della loss function. Altro fattore da tenere in considerazione è la scelta della funzione di attivazione nella fase di forward, in quanto da essa dipende l'output ottenuto ad ogni passo dell'apprendimento, quindi anche il valore della loss function e di conseguenza del gradiente. Alcune funzioni di attivazione possono portare ad un valore del gradiente tendente a zero, rendendo difficile completare in modo efficiente l'addestramento. Questo problema, noto come "scomparsa del gradiente", verrà affrontato più avanti nella sezione riguardante i layer di attivazione di una rete convoluzionale.

2.3 Reti neurali convoluzionali

La rete VDSR, come vedremo, rientra in una particolare categoria di reti neurali definite "covoluzionali". Le reti convoluzionali (CNN) costituiscono uno dei più noti algoritmi del deep learning, sono impiegate nello specifico nel campo delle applicazioni di Computer Vision e sono specializzate nella classificazione di immagini, video, testi e altri formati multimediali. Grazie a

questo tipo di rete è possibile riconoscere dettagli e pattern visivi all'interno di immagini mediante l'apprendimento automatico delle feature. Nel nostro caso, la rete VDSR prende in input un'immagine interpolata a bassa risoluzione e ha come scopo quello di modellarne i dettagli per risalire alla migliore approssimazione possibile dell'immagine ad alta risoluzione originale.

Nel mondo digitale un'immagine viene rappresentata come un tensore di 3 dimensioni (altezza, larghezza, numero di canali di colore). In una rete neurale semplice, un'immagine data in input può occupare un volume troppo grande, in quanto le sue dimensioni vanno moltiplicate per determinare il numero di connessioni fra ciascun neurone. Ad esempio data un'immagine di dimensioni 100x100x3, solamente il primo layer della rete presenterebbe 30000 connessioni. Per una rete di tipo fully connected questo porterebbe ad un numero impensabile di parametri da tenere in considerazione. Le reti CNN risolvono questo problema in quanto i neuroni di uno strato, organizzati in strutture tridimensionali, sono connessi solamente ad un ristretto numero di neuroni dello strato precedente.

I layer che compongono una rete CNN sono di vario tipo, ciascuno con una funzione ben determinata. I layer convoluzionali sono quelli responsabili del riconoscimento di feature nell'immagine e sono presenti in tutti i tipi di CNN. Gli altri layer sono di supporto alla convoluzione e possono servire ad aggiustare i pesi delle connessioni, a ridimensionare i filtri di convoluzione, ecc. In questo lavoro verranno descritti i layer necessari alla costruzione della rete VDSR.

2.3.1 Layer convoluzionale

È il layer fondamentale di una rete convoluzionale, in quanto è responsabile del riconoscimento di una particolare feature dell'immagine. Matematicamente è possibile dare una definizione dell'operazione di convoluzione: date due funzioni, essa consiste nell'integrare la prima rispetto alla seconda. L'integrale ottenuto esprime quanto le due funzioni si sovrappongono l'una con l'altra. Nel nostro caso, le due funzioni coincidono con l'immagine di par-

tenza a bassa risoluzione data in input e con una matrice detta "filtro". Il filtro ha dimensioni ridotte rispetto all'immagine di partenza in altezza e larghezza ma si applica su tutta la profondità: se l'immagine ha solo un canale il risultato della convoluzione sarà una matrice bidimensionale, altrimenti lo stesso filtro dovrà essere utilizzato per ogni canale e l'output sarà dato dalla somma dei risultati per tutti i canali. Ogni filtro è associato ad una feature, cioè una caratteristica dell'immagine che si vuole isolare ed identificare. Ad esempio ci può essere un filtro per le linee verticali, uno per le orizzontali, e così via.

L'apprendimento avviene effettuando una traslazione (una convoluzione appunto) del filtro sull'immagine a bassa risoluzione, in modo tale da rilevare le feature su porzioni piccole dell'immagine. È importante sottolineare come ad ogni passo della traslazione, solo una porzione dell'immagine LR venga considerata: ciò significa che all'interno di un layer convoluzionale ogni nodo è collegato solo ad un numero limitato di nodi di input, e ricordiamo che questa è la caratteristica che differenzia una rete convoluzionale da una rete tradizionale fully connected. La regione di input che di volta in volta viene considerata viene chiamata "receptive field". Man mano che il filtro scorre sull'immagine viene eseguita una moltiplicazione scalare elemento per elemento tra i valori della matrice filtro e quelli del receptive field. Ad ogni passo i risultati dei prodotti appena eseguiti vanno sommati per determinare il valore dell'elemento che va a completare la matrice risultato. La matrice ottenuta in output ha le stesse dimensioni del filtro ed è chiamata "feature map". L'output finale del livello è dato dalla sommatoria delle feature map ottenute dallo stesso filtro su un particolare receptive field. Ovviamente per filtri diversi verranno prodotte mappe di attivazione diverse a partire dalla stessa immagine LR. Maggiore è il numero di filtri utilizzati, maggiore sarà il numero di feature estratte e migliore sarà la performance finale della rete nel ricostruire l'immagine ad alta risoluzione. Nella costruzione di un layer convoluzionale è fondamentale tener conto di tre parametri, anche chiamati iperparametri: profondità (depth), stride e padding.

- Profondità: o depth, fa riferimento al numero di filtri per ogni layer convoluzionale. Il numero di filtri utilizzati definisce la profondità della feature map, o in alternativa il numero di neuroni del livello convoluzionale collegati alla stessa regione di input (cioè allo stesso receptive field).

- Stride: indica il passo secondo il quale far traslare il filtro sull'immagine originale, cioè il criterio con il quale individuare di volta in volta il receptive field. Ad esempio se lo stride è pari a 1, allora il filtro si muoverà di un pixel alla volta, considerando tutta l'immagine, se lo stride è uguale a 2, verrà saltato un pixel ogni due a ciascuno spostamento del filtro, e così via. Maggiore il valore assegnato allo stride, minori saranno le dimensioni dell'output, questo perchè il filtro impiegherà meno passi per coprire l'intera immagine riducendo la sovrapposizione fra i campi recettivi tra un passo e quello successivo. Al contrario uno stride basso crea overlapping delle regioni di input considerate e di conseguenza un aumento delle dimensioni dell'output finale.

- Padding: consiste nell'aggiungere all'immagine originale un bordo esterno composto da pixel con valori tutti uguali a 0 (si parla infatti di "zero-padding"). Questa procedura è utile in quanto le regioni più esterne dell'immagine non vengono sottoposte al filtro durante la sua traslazione tanto quanto quelle più interne, che invece possono essere considerate molteplici volte a seconda del passo scelto. Applicando un bordo all'input è possibile mappare correttamente e con precisione anche i dettagli che stanno ai bordi dell'immagine. Un'altra funzionalità del padding è quella di poter controllare artificialmente le dimensioni dell'output finale, andando a modificare quelle dell'input: è infatti possibile controllare il numero di pixel di padding da inserire settando il parametro di padding al valore desiderato.

Generalmente una rete convoluzionale presenta molteplici layer di questo tipo, disposti in ordine crescente rispetto ai dettagli delle feature che possono

riconoscere. Ad esempio un layer situato all'inizio della rete sarà predisposto al riconoscimento di feature di basso livello, come linee, luci, curve, mentre gli ultimi layer prossimi allo strato di output permetteranno di riconoscere feature di alto livello, fino a comprendere interi oggetti.

2.3.2 Layer di attivazione

Un layer di attivazione svolge lo stesso compito delle funzioni di attivazione dei neuroni nelle reti neurali tradizionali: solo le feature che vengono attivate passano al layer successivo. All'interno di una rete CNN, tipicamente viene inserito un layer di attivazione dopo ogni layer convoluzionale al fine di introdurre un fattore di non linearità. Le operazioni svolte dal layer convoluzionale infatti, sono del tutto lineari in quanto semplici moltiplicazioni tra valori del filtro e del receptive field. È possibile dimostrare che una rete con molti layer che effettuano operazioni lineari equivale ad una rete con un solo layer lineare. Un modello lineare non permette di ottenere risultati soddisfacenti per decisioni complesse, come può essere il riconoscimento di feature di immagini, ma funziona solo per input linearmente separabili.

È possibile scegliere fra numerose funzioni di attivazione, ma come precedentemente accennato alcune di esse, tra cui la sigmoide, possono portare al problema della scomparsa del gradiente. Ciò è dovuto al fatto che queste funzioni comprimono i dati di input in un range di valori limitato (ad esempio fra 0 e 1), riducendo di conseguenza anche il valore delle derivate. Per reti con molti layer questo diventa un problema nel momento in cui le derivate vengono moltiplicate fra di loro molteplici volte mediante backpropagation, portando ad avere un valore del gradiente tendente a zero e impedendo così di avere un apprendimento efficiente. Per questo motivo vengono preferite altre funzioni di attivazione che permettono di aggirare il problema. Tra queste ne vengono qui presentate tre, che saranno impiegate all'interno del modello VDSR.

Rectified Linear Unit

ReLU è il layer di attivazione in assoluto più utilizzato, esso applica la funzione:

$$f(x) = max(0, x)$$

eliminando tutti i valori negativi assunti dai pixel delle feature map provenienti dal layer precedente e sostituendoli con 0. La derivata di questa funzione ha la forma:

$$1, \quad se\, x > 0$$

$$0, \quad se\, x \leqslant 0$$

In questo modo vengono esclusi sia valori negativi, che moltiplicati fra loro porterebbero alla scomparsa del gradiente, sia valori troppo elevati che potrebbero causare il problema inverso, l'esplosione del gradiente. Si noti come per valori positivi la ReLU si comporti come una funzione lineare, pertanto ne conserverà caratteristiche positive come la semplicità e la velocità di calcolo. Inoltre il fatto di non prevedere nessun parametro modificabile o allenabile facilita notevolmente la fase di backpropagation. Un altro pregio del layer ReLU è quello di ridurre la densità della rete: per reti molto complesse, infatti, avere un numero elevato di neuroni sempre attivi può essere dispendioso e appesantire il modello rendendolo inefficiente. L'attivazione ReLU permette di settare a 0 circa la metà dei valori dei nodi, i quali non verranno attivati e non subiranno trasformazioni lineari, rendendo la rete più leggera. Questa caratteristica della funzione ReLU, tuttavia, può anche avere un risvolto negativo nel momento in cui troppi nodi vengono disattivati e avendo un gradiente nullo non possono apportare più nessun contributo all'addestramento, rallentando la convergenza. Tale problematica, chiamata "dying Relu" ha portato all'introduzione di nuove funzioni di attivazione, come la LeakyReLU e la ELU descritte di seguito.

Leaky Rectified Linear Unit

La LeakyReLU è una funzione nata come tentativo di risolvere il problema di dying ReLU. Applicata ad un valore x risulta:

$$f(x) = x \quad se\, x > 0$$

$$f(x) = \alpha x \quad se\, x \leqslant 0$$

La differenza sostanziale con la ReLU è che non annulla i valori negativi ma li moltiplica per una costante $\alpha = 0.01$. È tuttavia possibile scegliere diversi valori da assegnare ad α, in tal caso la funzione LeakyReLU viene definita anche "Randomized Relu". La sua derivata assume i valori:

$$1, \quad se\, x > 0$$

$$\alpha, \quad se\, x \leqslant 0$$

Questa funzione permette di mantenere attivi tutti i neuroni della rete fino all'ultima fase di addestramento, dal momento che il gradiente sarà piccolo ma non pari a zero, così che possano contribuire all'aggiustamento progressivo dei pesi. Tuttavia sperimentalmente non mostra grandi miglioramenti nei risultati ottenuti rispetto alla ReLU, per questo motivo non è tra le più comuni funzioni utilizzate.

Exponential Linear Unit

L'attivazione ELU è un'ulteriore alternativa alla ReLU che, come la LeakyRelu, permette preservare i valori negativi ma lo fa introducendo un termine esponenziale:

$$f(x) = x \quad se\, x > 0$$

$$f(x) = \alpha(e^x - 1) \quad se\, x \leqslant 0$$

la sua derivata è:

$$1, \quad se\, x > 0$$

$$f(x) + \alpha, \quad se\, x \leqslant 0$$

con $\alpha = 1$. La peculiarità della funzione ELU è portare i valori ad avere una media centrata in 0 per velocizzare la computazione e di conseguenza i tempi di addestramento.

2.3.3 Layer Dense

Il dense layer è anche detto layer fully connected: ciò significa che ogni neurone di questo strato è connesso con tutti i neuroni attivati appartenenti al layer precedente, o in altri termini il receptive field coincide con l'intero strato precedente. Esso corrisponde esattamente alla struttura di un hidden layer delle reti neurali tradizionali fully connected, pertanto la computazione svolta è la stessa, di tipo lineare, che consiste in un prodotto scalare fra matrici.

La dimensione dell'output di un layer di questo tipo può essere scelta arbitrariamente. Ad esempio in una rete costruita per risolvere problemi di classificazione, tale dimensione sarà pari al numero di classi tra le quali è possibile scegliere: in un'applicazione per il riconoscimento di cifre, ci saranno 10 potenziali classi ciascuna corrispondente ad un numero tra 0 e 9, pertanto il parametro che definisce le dimensioni dell'output sarà fissato a 10. Per questa sua caratteristica, molto spesso il layer fully connected viene inserito alla fine della rete convoluzionale, in quanto rende possibile la suddivisione in classi dei risultati ottenuti fino a quel punto.

A differenza dei layer di attivazione, le connessioni di un layer dense sono pesate, pertanto i parametri di questo strato sono coinvolti attivamente nella fase di training e i pesi sono soggetti a modifiche dovute all'applicazione di backpropagation e discesa del gradiente.

2.4 Algoritmi di ottimizzazione

Come già anticipato, nella fase di addestramento di una rete neurale una parte rilevante è svolta dall'algoritmo di ottimizzazione, che permette di aggiornare i parametri (pesi e bias) in modo da minimizzare la funzione

di errore. Gli algoritmi di ottimizzazione possono essere suddivisi in due principali categorie:

- Algoritmi di ottimizzazione di primo ordine: utilizzano il valore della derivata prima parziale della funzione di errore, basandosi sul metodo della discesa del gradiente. Il gradiente infatti può essere definito come la derivata prima di una funzione dipendente da più di una variabile.

- Algoritmi di ottimizzazione di secondo ordine: utilizzano le derivate di secondo ordine. Poichè questo tipo di derivata richiede calcoli più complessi per essere risolta, l'aggiornamento dei pesi tramite retro propagazione risulta molto più lento, pertanto algoritmi di questo tipo sono raramente utilizzati.

Da questa distinzione si ricava che gli algoritmi ad oggi più comuni sono quelli basati sul metodo di discesa del gradiente. L'applicazione standard dell'algoritmo calcola per ogni iterazione il gradiente di tutti gli elementi del dataset, introducendo un pesante fattore di rallentamento al training della rete per volumi di dati molto estesi. Per questo motivo nel tempo si è cercato di migliorare la performance studiando alternative al metodo standard, tra queste ne vedremo tre tra le più significative: Stochastic Gradient Descent (SGD), Mini-batch gradient descent e Adaptive moment estimation (Adam).

Prima di entrare nel merito di questi metodi, è bene definire alcuni parametri comuni a tutti gli algoritmi di ottimizzazione basati sulla discesa del gradiente:

- Batch: campione ristretto di dati provenienti dal training set. Il batch size indica la dimensione del batch, ovvero il numero di elementi che contiene. L'algoritmo standard di ottimizzazione, anche detto Batch Gradient Descent, come già anticipato utilizza un solo batch con dimensione pari a quella dell'intero dataset per compiere un solo aggiornamento.

- Epoch: indica il numero di volte in cui la rete viene attraversata dall'intero training set. Una sola epoca indica che ogni elemento del dataset ha attraversato la rete una sola volta ed ha avuto una sola opportunità di influenzare l'aggiornamento dei parametri di apprendimento. È molto comune utilizzare un numero di epoche maggiore di uno, in modo tale da permettere alla rete di sistemare i pesi nel modo più preciso possibile, minimizzando di volta in volta la loss function. La dimensione e il numero di batch considerati ad ogni iterazione, insieme con il numero di epoche sono fattori che permettono di distinguere tra i vari algoritmi di ottimizzazione.

2.4.1 Stochastic Gradient Descent

L'algoritmo di ottimizzazione SGD, prevede che venga effettuato un aggiornamento dei parametri della rete per ogni singola istanza del training set. In altri termini, ad ogni iterazione non si considera l'intero dataset ma un suo singolo elemento: il batch size di questo algoritmo sarà quindi pari a 1. Per training set di grandi dimensioni risulta vantaggioso in quanto aggiornamenti così frequenti permettono da un lato di occupare meno memoria (i calcoli sono meno complessi, dato che viene considerato un valore di gradiente alla volta) e dall'altro di poter migliorare ad ogni passo l'approssimazione del minimo, date le continue oscillazioni della funzione di errore. Quest'ultimo punto tuttavia rappresenta anche un difetto di questo algoritmo: aggiornamenti frequenti dei parametri possono causare difficoltà nell'ottenere la convergenza al minimo, aumentando di conseguenza le tempistiche della fase di training.

Una variante di questo algoritmo utilizza il cosiddetto "momento" per regolare le oscillazioni della loss function: viene definita una costante γ (generalmente appartenente all'intervallo [0, 1]) che moltiplica il valore assunto dai parametri (pesi e bias) nel passo precedente. L'equazione di discesa del

gradiente diventa quindi:

$$w_{ij}(t+1) = \gamma w_{ij}(t) - \eta \frac{\delta E}{\delta w_{ij}}(t)$$

Grazie a questo accorgimento viene incrementata la velocità di convergenza nella direzione corretta, mentre le oscillazioni verso quella opposta vengono smorzate. Un altro perfezionamento che è possibile introdurre nell'algoritmo di discesa del gradiente è il "weight decay". Esso consiste nel moltiplicare ogni peso ottenuto per un certo valore costante e di poco inferiore a 1. In questo modo è possibile evitare che i pesi assumano valori troppo grandi appesantendo l'addestramento.

2.4.2 Mini-batch Gradient Descent

Questo metodo prevede la divisione del dataset in batch di dimensioni variabili. Ad ogni iterazione un solo batch attraversa la rete e contribuisce al training dei parametri. In questo caso il batch size sarà compreso tra 1 e la dimensione del training set, estremi esclusi. Poichè ad ogni passo solo un piccolo campione di dati viene considerato, questo algoritmo presenta gli stessi vantaggi del metodo SGD: migliore gestione dello spazio di memoria, semplicità computazionale e possibilità di evitare minimi locali.

Sebbene questo algoritmo, così come il precedente, sia ad oggi molto utilizzato, presenta caratteristiche che possono essere migliorate. Ad esempio, abbiamo già evidenziato l'importanza della scelta del valore del learning rate, il quale può influenzare negativamente la fase di training dal momento che l'aggiornamento dei pesi dipende fortemente da esso. Un learning rate troppo basso o troppo alto può causare problemi di lenta o mancata convergenza. A tal proposito, è importante notare che gli algoritmi sopra descritti utilizzano un unico valore di learning rate che rimane costante per ogni iterazione: sarebbe molto più appropriato assegnare learning rate diversi a seconda della frequenza con la quale le feature si presentano, assegnando un peso maggiore a quelle più rare e uno minore a quelle ricorrenti.

2.4.3 Adaptive moment estimation (Adam)

Uno dei più recenti algoritmi nati per risolvere il problema della scelta del learning rate è Adam. Esso estende i risultati di altri due metodi proposti per lo stesso scopo, l'Adaptive gradient algorithm (AdaGrad) e il Root mean square propagation (RMSProp).

Adagrad consiste nell'utilizzo di valori diversi di learning rate per ogni parametro, in base alla frequenza con cui essi si presentano in fase di addestramento. Per far ciò il termine η viene scalato in base alle derivate parziali ottenute per lo stesso parametro nelle iterazioni precedenti:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\eta}{\sqrt{G_{ij} + \epsilon}} g(t) \quad con\, g(t) = \frac{\delta E}{\delta w_{ij}}(t)$$

dove G_{ij} è la sommatoria dei quadrati dei gradienti di w_{ij} calcolati fino al passo t, mentre ϵ è una costante che viene aggiunta in modo da evitare di ottenere un denominatore nullo nel caso in cui G valga 0.

Il metodo RMSProp sostituisce G_{ij} con la media esponenziale del quadrato dei gradienti. Questo perchè considerare la sommatoria di tutti i gradienti durante tutto il training potrebbe portare ad un decrescere eccessivo del learning rate che impedirebbe di proseguire con l'apprendimento.

Adam segue questa linea calcolando sia la media esponenziale del quadrato del gradiente per scalare il learning rate, sia la media esponenziale di primo ordine del gradiente che vada a sostituire il termine g(t). Chiamiamo questi due valori rispettivamente v_{ij} (varianza) e m_{ij} (media), essi sono definiti come:

$$m_{ij} = \beta_1 m_{ij}(t) + (1 - \beta_1) g_{ij}(t)$$

$$v_{ij} = \beta_2 v_{ij}(t) + (1 - \beta_2) g_{ij}(t)^2$$

Poichè v e m vanno inizializzati a 0 e i fattori β_1 e β_2 hanno valori prossimi a 1, è molto probabile che anche le successive stime di v e m siano tendenti a 0. Gli ideatori dell'algoritmo hanno dimostrato che è possibile correggere i valori di questi due termini per ovviare al problema:

$$\bar{m}(t) = \frac{m(t)}{1 - \beta_1(t)}$$

$$\bar{v}(t) = \frac{v(t)}{1 - \beta_2(t)}$$

La formula della discesa del gradiente aggiornata diventa:

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\eta}{\sqrt{\bar{v}(t)} + \epsilon}\bar{m}(t)$$

I vantaggi di questo metodo sono la facilità implementativa, l'efficienza computazionale, la richiesta di poca memoria e la scalabilità per grandi quantità di dati. La rete VDSR, come si vedrà, implementa questo algoritmo nella fase di addestramento.

2.5 Il modello VDSR

La prima rete CNN proposta per il problema di SISR è stata la SRCNN, composta da 3 layer, ognuno con una funzione ben precisa (patch extraction, nonlinear mapping e reconstruction). Ad oggi resta un modello valido, soprattutto per la sua velocità in fase di addestramento, ma nel tempo è stato nettamente superato da altre proposte di reti che si sono rivelate particolarmente efficienti per quanto riguarda l'accuratezza dei risultati. Uno dei problemi fondamentali della rete SRCC è la sua semplicità: è infatti stato dimostrato che reti di tipo deep, con molteplici layer nascosti, permettono di aumentare drasticamente lo spazio delle soluzioni.

La rete VDSR di Simonyan e Zisserman[20] è stata la prima proposta in questo senso, il primo modello DNN utilizzato nel campo della super risoluzione. In questa sezione verrà descritta l'architettura della rete VDSR nel dettaglio, quindi ne verranno analizzati i punti di forza rispetto ad altri modelli. Per la descrizione della struttura della rete si farà riferimento in particolare all'articolo[16] redatto dal Department of Electrical and Computer Engineering della Seoul National University. Infine saranno presentate

alternative alla struttura originale, con modifiche di alcuni parametri di addestramento e della tipologia di layer impiegati. Eventuali miglioramenti e variazioni nella performance saranno valutati nel capitolo successivo.

2.5.1 Struttura della rete

La rete VDSR è composta da 20 layer totali, come illustrato in Figura 2.2, aventi la seguente organizzazione:

- un layer convoluzionale di input
- 18 layer nascosti convoluzionali e di attivazione ReLU, alternati uno con l'altro
- un layer convoluzionale di output

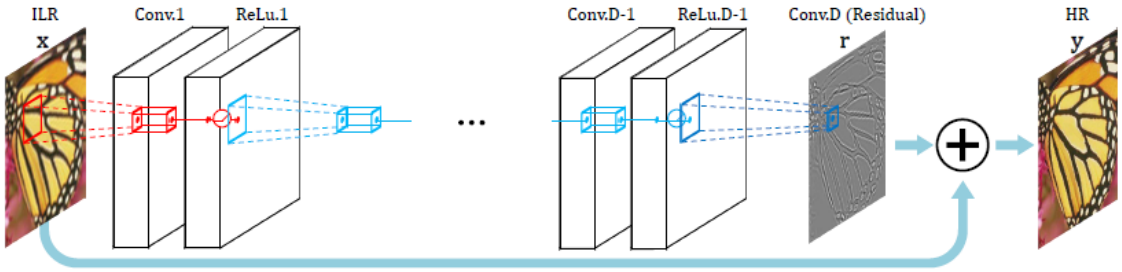


Figura 2.2: Struttura della rete VDSR

L'input della rete consiste in immagini interpolate mediante il metodo bicubico, le quali vengono poi trasmesse in avanti nella rete fino allo strato di output. Ogni layer convoluzionale, eccetto l'ultimo, presenta 64 filtri di dimensione 3x3 che scorrono sull'immagine determinando 64 feature map per ogni passo in avanti nella rete. Ogni filtro viene spostato con uno stride pari a 1, perciò tutti i pixel vengono considerati almeno una volta. Per evitare che le dimensioni della feature map diminuiscano eccessivamente per via della profondità della rete (le dimensioni vengono ridotte ad ogni operazione di

convoluzione), rischiando di perdere informazioni rilevanti sui dettagli, viene eseguita un'operazione di padding per ogni layer convoluzionale. Ricordiamo che il padding consiste nell'aggiungere una cornice di pixel con valore pari a zero intorno all'immagine: applicando questo aggiustamento ad ogni passo della rete le dimensioni della feature map restano invariate tra lo stato di input e quello di output. Il padding, inoltre, rende la rete VDSR più efficiente rispetto ad altre nel ricostruire i dettagli situati ai bordi dell'immagine, in quanto porta tutti i pixel verso il centro in modo che siano il più possibile sottoposti a tutti i filtri.

Veniamo ora alla fase di training. La loss function scelta per l'addestramento dei parametri della rete è l'errore quadratico medio (MSE) fra l'immagine di input e quella di output. Ricordiamo che formalmente, indicata con x l'immagine LR e con y quella HR, e dato un training set di immagini, il problema di super-risoluzione consiste nel determinare la funzione f tale che:

$$y = f(x)$$

Per raggiungere l'approssimazione migliore di y, in fase di backpropagation viene calcolato il MSE tra il risultato ottenuto e quello desiderato:

$$\frac{1}{2} \|y - f(x)\|^2$$

La particolarità della rete VDSR è che applica un apprendimento basato sui residui. Poichè infatti si presume che l'immagine di partenza e quella di arrivo siano simili fra loro, viene definita un'immagine detta "residua" data dalla differenza tra le due: $r = y - x$. I valori dei pixel dell'immagine r saranno in gran parte pari a zero, o assumeranno valori molto piccoli. Questo permetterà di alleggerire notevolmente i costi computazionali della rete nel calcolo dell'errore MSE, che può essere riscritto come:

$$\frac{1}{2} \|r - f(x)\|^2$$

Questo tipo di approccio si riflette nella struttura della rete VDSR. L'ultimo layer, chiamato anche loss layer, prende in input l'immagine LR interpolata originale e l'immagine residua restituita dalla rete, quindi ricostruisce l'output finale sommandole. La funzione di errore viene calcolata come la differenza tra l'immagine HR e quella ricostruita.

Per quanto riguarda l'algoritmo di ottimizzazione impiegato per la discesa del gradiente, gli autori del modello suggeriscono il mini-batch gradient descent con momentum = 0.9, weight decay = 0.0001, batch size = 64 e numero di epoche = 10. Tuttavia come verrà evidenziato in seguito, sostituendo questo metodo con Adam è possibile ottenere risultati migliori a parità di learning rate.

2.5.2 Punti di forza del modello

Le caratteristiche distintive che hanno reso la rete VDSR innovativa sono sostanzialmente tre: la struttura di tipo "very deep", l'implementazione del residual learning e l'apprendimento multi-scala. Nel tempo molte altre reti costruite per operare sulla super risoluzione hanno adottato uno o più di questi approcci, ma il modello VDSR è stato sicuramente pionieristico nel campo. I risultati numerici mostrati nei grafici e tabelle di questa sezione fanno riferimento a immagini contenute nel dataset "Set5".

Modello very deep

Sappiamo già che la particolarità delle reti neurali convoluzionali è quella di considerare come input di ogni layer solo un'area ristretta di neuroni appartenenti allo strato precedente, e che tale area è chiamata receptive field. Con questo accorgimento è possibile sfruttare la correlazione a livello locale dei pixel disposti su layer adiacenti per generare feature maps il più possibile dettagliate. Procedendo da un layer a quello successivo le dimensioni dei receptive field aumentano, in quanto i valori dei pixel vengono moltiplicati e sommati tra loro. Nel nostro caso, la rete VDSR imposta filtri iniziali di dimensioni 3x3. Tali dimensioni raddoppiano in altezza e larghezza ogni

volta che si passa ad un layer successivo, ne consegue che per una rete di profondità N, le dimensioni del receptive field alla fine della rete saranno (2N+1)x(2N+1): numero di layer e dimensioni del receptive field sono direttamente proporzionali. Un receptive field esteso permette di raccogliere un maggior numero di informazioni relative all'immagine. Per problemi mal posti, come quello di super risoluzione, questo è essenziale in quanto permette di avere maggiori presupposti sui quali ricostruire dettagli ad alta frequenza, ottenendo un risultato finale di qualità superiore. Il grafico in Figura 2.3 mostra il variare del valore di PSNR (parametro che esprime in termini numerici la bontà dell'approssimazione basandosi sul valore del MSE tra l'output ottenuto e quello previsto)a seconda del numero di layer (depth) della rete, per diversi valori di scale factor. Come previsto, in tutti i casi la performance ha un andamento crescente all'aumentare della profondità.

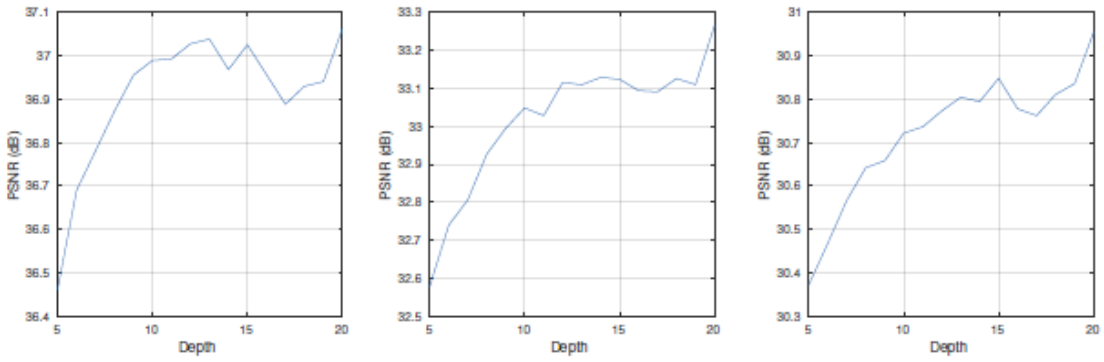


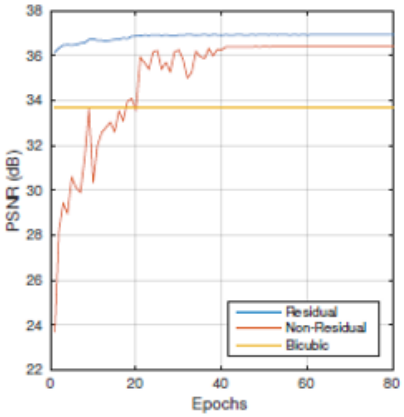
Figura 2.3: Andamento del PSNR al variare del numero di layer. I fattori di scala considerati sono 2, 3, 4 procedendo da sinistra verso destra.

Disporre di una rete very deep è vantaggioso anche dal punto di vista della semplicità di calcolo: si ricordi infatti che gli strati nascosti comprendono numerosi layer di attivazione ReLU, i quali apportano non linearità al modello e permettono di risolvere funzioni complesse con un numero limitato di neuroni, inibendo quelli che assumono valori negativi.

Residual learning

Nella rete SRCNN l'immagine di partenza viene considerata solo nel primo strato di input, per poi essere scartata. Per questo motivo, la rete deve tenere in memoria ad ogni passo tutti i dettagli dell'immagine, e l'output sarà ricostruito solamente a partire dalle feature apprese. In reti con un consistente numero di layer, questo processo richiede un grande dispendio di memoria e può impiegare molte epoche per convergere alla performance ottimale.

La rete VDSR permette di conservare l'immagine così come viene data in input e di predire solamente i contenuti ad alta frequenza (l'immagine residua). Al termine del processo le due immagini vengono sommate per ottenere quella ad alta risoluzione. Questo sistema permette alla rete di convergere molto più velocemente, raggiungendo lo stato dell'arte nel giro di qualche epoca.



Epoche	10	20	40	80
Residui	36.74	36.74	36.91	36.93
No residui	30.33	33.59	36.36	36.42
Differenza	6.41	3.28	0.65	0.52

Figura 2.4: Miglioramento della perfromance dovuto all'uso dei residui

Inoltre si ha un incremento anche per quanto riguarda l'accuratezza dei risultati, ottenendo valori di PSNR maggiori indipendentemente dal learning rate scelto. Nella Figura 2.4 sono messi a confronto alcuni risultati otenuti da due reti, una che lavora con e l'altra senza l'uso di residui, con un learning

rate = 0.01. Si noti come la rete che implementa l'apprendimento con i residui impiega solamente 10 epoche per ottenere un risultato migliore di quella senza residui dopo 80 epoche.

Apprendimento multi-scala

Il fattore di scala (o scale factor) indica il valore per il quale le dimensioni dell'immagine LR sono state moltiplicate nel processo di ricampionamento. Ad esempio, se il fattore di scala è 2, le dimensioni dell'immagine sono state raddoppiate.

Tipicamente viene creata una rete diversa per ogni fattore di scala per essere pronti a qualsiasi scenario, ma questo è sconveniente se disponiamo di reti costituite da molti layer, in quanto addestrare molteplici modelli separati per ogni dataset dilata notevolmente i tempi di training. Inoltre una rete addestrata per un solo fattore di scala, non permette di gestire dataset costruiti con scale diverse.

La rete VDSR al contrario, permette di eseguire un addestramento definito "multi-scala", condividendo i parametri per diversi valori di scale factor. Questo permette di allenare in una sola volta dati con fattori di scala differenti, invece che costruire un dataset separato per ognuno. Durante il training è quindi possibile che un batch contenga immagini con diversi valori di scale factor. Test sperimentali eseguiti su reti a scala sia singola che multipla hanno permesso di ricavare due osservazioni interessanti, riportati anche nella tabella di Figura 2.5:

- a parità di fattore di scala, i risultati conseguiti da una rete multi-scala sono comparabili a quelli di una rete costruita ad hoc su quell'unico fattore di scala

- per grandi fattori di scala (es. 3, 4) la rete multi-scala è migliore di quella singola, in quanto l'addestramento su valori di scala inferiori (es. 2) permette di trarre informazioni utili per gli addestramenti successivi.

È infatti pratica molto comune avviare il training di una rete a partire da input preaddestrati.

Test/Train	x2	x3	x4	x2,3	x2,4	x3,4	x2,3,4	Bicubica
x2	37.10	30.05	28.13	37.09	37.03	32.43	37.06	33.66
x3	30.42	32.89	30.50	33.22	31.20	33.24	33.27	30.39
x4	28.43	28.73	30.84	28.70	30.86	30.94	30.95	28.42

Tabella 2.1: Valori PSNR per modelli addestrati con diversi fattori di scala. I valori in rosso indicano che il corrispondente scale factor è stato utilizzato per l'addestramento.

2.5.3 Proposte di modifiche al modello VDSR

Nella fase sperimentale di questa tesi, si è cercato di testare alternative al modello originale proposto nell'articolo di riferimento. Di seguito verrà fatta una panoramica dei principali tentativi eseguiti, mentre nel capitolo 3 saranno analizzati e confrontati nel dettaglio i risultati ottenuti dal punto di vista numerico-statistico.

Una prima modifica testata ha riguardato la fase di preparazione delle immagini a bassa risoluzione da fornire in input alla rete. Nel modello originale viene specificato che la rete VDSR accetta immagini LR ricampionate mediante interpolazione bicubica. Il tentativo eseguito è stato quello di modificare il metodo di interpolazione, sostituendolo con la nearest neighbor, per valutare la variazione della performance della rete. Come ci si potrebbe aspettare, il risultato è stato notevolmente peggiore: l'interpolazione nearest neighbor applicata ad immagini complesse, contenenti molti dettagli, risulta essere troppo approssimativa e introduce fattori di disturbo (aliasing, blurring) che compromettono la precisione della rete. Questa strada è stata quindi del tutto scartata.

Un secondo tentativo, del quale verranno mostrati alcuni risultati, è stato modificare il layer di attivazione ReLU nella struttura interna della rete.

Le funzioni di attivazione alternative che sono state scelte sono la LeakyReLU, con un fattore aplha=0.3, e la ELU. Come vedremo, questa modifica ha apportato qualche miglioramento dal punto di vista della percezione dell'immagine (ovvero un incremento del valore SSIM che misura la qualità visiva dell'approssimazione) ma nessuna particolare variazione del PSNR.

Per quanto riguarda l'algoritmo di ottimizzazione per la fase di training, come anticipato è stato sostituito il metodo di discesa del gradiente con mini-batch suggerito dall'articolo con l'algoritmo Adam, che si è rivelato assolutamente migliore a parità di valore del learning rate.

La proposta che tuttavia ha restituito i risultati più significativi è stata quella di introdurre un fattore di densità nella rete mediante l'utilizzo di layer fully connected. La struttura base della rete è stata modificata inserendo tra il layer convoluzionale e quello di attivazione un layer dense, contenente 64 filtri, lungo tutta la rete. Come funzione di attivazione è stata mantenuta la ReLU, mentre nei layer convoluzionali è stata modificata la dimensione dei filtri, incrementandola da 3x3 a 7x7, secondo la logica che prevede risultati migliori per ampi receptive field. L'addestramento è stato effettuato sfruttando l'algoritmo di ottimizzazione Adam. Questa nuova architettura ha permesso di ottenere una performance migliore a livello globale, sia nel valore del PSNR sia in quello del SSIM, richiedendo un tempo di addestramento superiore ma ancora compatibile con le possibilità di una macchina comune.

Capitolo 3

Risultati numerici

In questo ultimo capitolo saranno esposti i risultati numerici e grafici ottenuti applicando la rete VDSR e le alternative ad essa proposte. In fase sperimentale sono stati fatti numerosi tentativi di modifiche ai parametri e alla struttura originale della rete, i più importanti dei quali sono stati introdotti nella Sezione 2.5.3. I risultati qui riportati faranno leva principalmente sul variare dei risultati per diversi fattori di scala e sul confronto fra i modelli VDSR e VDSR di tipo dense.

3.1 Composizione del dataset

La rete neurale VDSR costruita per risolvere il problema di super risoluzione prevede un processo di apprendimento di tipo supervisionato, quindi necessita di un dataset costituito da coppie di input e output su cui basare il suo addestramento. Esso è suddiviso in: training set e validation set. Il training set contiene le immagini da sottoporre alla rete in fase di addestramento, quindi costituisce il punto di partenza dell'algoritmo di training. Il validation set, invece, viene utilizzato per la validazione del modello ottenuto, eseguita al termine di ogni epoca dell'addestramento. Entrambi i set sono stati generati mediante lo stesso procedimento, a partire però da campioni di immagini diverse. Per generare il training set si è scelto un dataset ottenuto

sommando le 91 immagini del dataset T91[22] e le 200 immagini del Berkeley Segmentation Dataset[24], mentre per il validation set è stato utilizzato il dataset "Set5"[23]. Il procedimento per ottenere le immagini input della rete è il medesimo sia per la fase di addestramento sia per quella di validazione, e verrà descritto di seguito.

Ad ogni immagine iniziale ad alta risoluzione viene applicato un sottocampionamento con tre diversi fattori di scala (2, 3, 4) e rotazioni di 90, 180 e 270 gradi, per simulare l'acquisizione iniziale di un dato a bassa risoluzione. Succesivamente ogni dato viene sovracampionato con fattori di scala 2, 3, 4 mediante interpolazione bicubica. Le immagini del training set sono patch di dimensioni 41x41 pixel delle immagini ad alta risoluzione ottenute con interpolazione bicubica. Come descritto nel capitolo precedente, la rete VDSR ricostruisce per ogni input il residuo tra l'immagine iniziale ad alta risoluzione e l'immagine ad alta risoluzione ottenuta con la bicubica (Figura 2.2). Il training set, dunque, contiene i patch estratti dalle immagini bicubiche ad alta risoluzione e i corrispondenti patch residui, chiamati anche "labels", che corrispondono all'output fornito dal modello.

3.2 Criteri di valutazione

Quando un'immagine subisce delle modifiche, è molto probabile che subisca una perdita di informazioni che ne comprometta la qualità. Lo scopo della super risoluzione è quello di ripristinare le feature di immagini a bassa risoluzione ed ottenere immagini ad alta risoluzione con un più alto numero di pixel rispetto a quello di partenza. Si rende quindi necessario stabilire dei criteri per valutare la bontà dell'approssimazione ottenuta con gli algoritmi che risolvono il problema di SISR. In questo senso è possibile distinguere due classi di metodi di valutazione:

- Metodi soggettivi: si basano sulla percezione soggettiva di chi analizza l'immaginI e non prevedono nessun riferimento numerico prestabilito.

- Metodi oggettivi: utilizzano formule matematiche e statistiche per ottenere valori numerici che esprimano le caratteristiche fisiche dell'immagine, in modo da poter eseguire confronti tra grandezze della stessa unità di misura ed ottenere così risultati universalmente comparabili.

In questa sezione saranno definiti i due parametri su cui si baserà il confronto tra i risultati: il peak signal to noise ratio (PSNR) e lo structural similarity index (SSIM). Entrambi sono comunemente utilizzati nei problemi legati all'imaging e rientrano nella categoria dei metodi oggettivi.

3.2.1 PSNR

Il peak signal to noise ratio (PSNR) esprime il rapporto tra il massimo valore assunto dai pixel di un'immagine e la distorsione, il rumore, ad essa applicato. Date due funzioni, f e g, che nel nostro caso rappresentano rispettivamente l'immagine ad alta risoluzione originale e quella ottenuta dalla rete VDSR, è possibile calcolare il valore PSNR applicando la formula:

$$PSNR(f,g) = 10log_{10}\frac{max(f)}{MSE(f,g)}$$

dove $MSE(f,g)$ è l'errore quadratico medio fra le due funzioni, ed ha la forma:

$$MSE(f,g) = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}(f_ij - g_ij)^2$$

con W e H rispettivamente la larghezza e l'altezza delle immagini. Se il MSE tende a zero, il PSNR tende all'infinto, perciò maggiore sarà il valore PSNR di un'immagine, migliore sarà la sua qualità.

3.2.2 SSIM

Lo structural similarity index (SSIM) misura la somiglianza fra due immagini tenendo in considerazione, a differenza del PSNR, i termini di percezione della qualità del sistema visivo umano. Perchè questo sia possibile, il SSIM

considera la distorsione totale di un'immagine come il prodotto di tre fattori variabili: luminosità (l), contrasto (c) e struttura (s). Formalmente:

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g)$$

con f, g che nel nostro caso sono rispettivamente l'immagine HR e quella restituita dalla rete. I valori di l, c ed s sono definiti come segue:

$$l(f,g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C1}$$
$$c(f,g) = \frac{2\sigma_f\sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C2}$$
$$s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f\sigma_g + C3}$$

dove μ è la luminosità media, σ è la deviazione standard che misura il contrasto, e C_1, C_2, C_3 sono costanti aggiunte per evitare l'annullamento del denominatore. Il range di valori per il SSIM è [0, 1], più ci si avvicina allo zero, minore sarà la correlazione fra le due immagini.

3.3 Composizione del test set

Per testare la performance della rete, è stato costruito un test set contenente immagini provenienti da diversi dataset ("Set5", "Set14"[25], "Urban100" [26]), al fine di considerare tipologie di immagini il più possibile diverse fra loro. In particolare, un dataset rilevante per testare la rete VDSR è "Urban100". Esso contiene 100 immagini di ambienti urbani accomunate dalla presenza di numerosi dettagli come linee e bordi, soggetti al problema di aliasing. Le reti presenti in letteratura[4] che risolvono il problema di SISR non hanno ottenuto performance ottimali lavorando su questo dataset. Come verrà dimostrato dai risultati riportati, la rete VDSR, e in particolare quella modificata con l'aggiunta di layer fully connected, raggiunge performance ottimali soprattutto su questa tipologia di immagini. La Figura 3.1 mostra alcune delle 20 immagini scelte per la fase di training.

La preparazione delle immagini da passare alla rete per il test si articola in diverse fasi, di seguito riportate.

Innanzitutto i valori dei pixel dell'immagine vengono trasformati in valori double e successivamente normalizzati nell'intervallo [0,1] dividendo ciascun valore per il massimo.

Quindi, poichè la rete opera su input bidimensionali, è necessario separare i canali RGB delle immagini a colori, tenendone in considerazione solamente uno ed eliminando gli altri. In fase di elaborazione dei risultati sarà possibile ricostruire l'immagine con i colori originali semplicemente sovrapponendo i tre output ottenuti considerando ogni volta un canale diverso.

Prima di procedere con il ricampionamento, deve essere scelto il fattore di scala da utilizzare: esso è un valore numerico intero che indica di quanto ingrandire o rimpicciolire un'immagine. Una delle caratteristiche principali della rete VDSR è che permette di implementare un apprendimento multi-scala, e una volta addestrata può essere utilizzata per operare su diversi fattori di scala. I valori qui prescelti per questo parametro sono 2, 3 e 4.

Il passo successivo è quello di ridimensionare l'immagine in modo tale che le sue dimensioni siano divisibili esattamente per il fattore di scala selezionato. Questa operazione viene eseguita mediante la definizione della funzione $modcrop(img, scalefactor)$, la quale assegna all'immagine passata nel parametro img nuove dimensioni che corrispondono al primo intero inferiore o uguale al valore delle dimensioni originali esattamente divisibile per il fattore di scala.

L'ultimo passaggio si divide due fasi distinte:

- fase di downsampling: ogni immagine reale ad alta risoluzione viene sottocampionata con fattore di scala 2, 3 o 4, al fine di ottenere un dato a bassa risoluzione;

- fase di upsampling: ai dati LR viene applicata un'interpolazione bicubica con fattore di scala 2, 3 o 4.

Applicando questi passi a tutto il dataset si ottengono i campioni richiesti come input dalla rete. Il compito della rete sarà quello di ricostruire i dettagli persi per ottenere un'immagine il più possibile simile a quella originale in termini di risoluzione.

Una nota finale riguarda il formato di salvataggio dei dati: per preservare i valori numerici dei pixel delle immagini si è scelto di salvare i campioni a bassa risoluzione come griglie di valori in formato csv. In questo modo sarà anche più semplice in fase di testing e valutazione dei risultati prelevare i valori numerici per il calcolo degli errori.

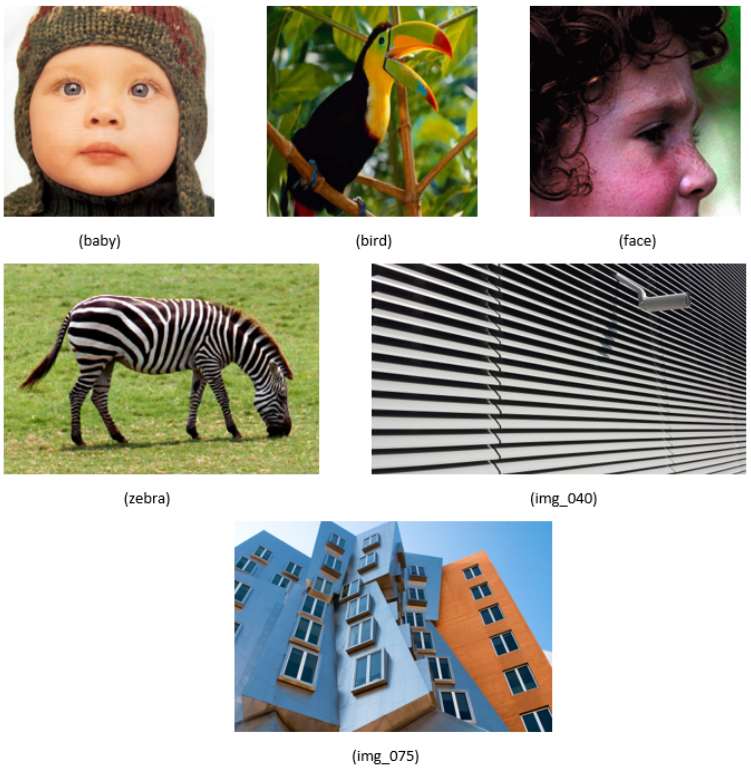


Figura 3.1: Alcune delle immagini che compongono il dataset: "baby" e "bird" ("Set5"), "zebra" e "face" ("Set14") e "img040" e "img075" ("Urban100")

3.4 Analisi dei risultati

Prima di presentare i risultati relativi alle reti costruite, vengono qui elencati i tentativi eseguiti in fase sperimentale che hanno determinato scelte comuni a tutti i modelli, riguardanti parametri e metodi utilizzati:

- Come algoritmo di interpolazione da applicare nella fase di preparazione delle immagini è stato mantenuto quello bicubico. Infatti un tentativo iniziale è stato quello di addestrare e testare la rete su immagini LR interpolate mediante il metodo nearest neighbor, ma i risultati ottenuti sono stati decisamente peggiori.

- Come learning rate è stato mantenuto il valore 0.0001 già proposto dagli autori del modello. Altri valori sono stati presi in considerazione ma non hanno apportato particolari modifiche ai risultati.

- L'algoritmo di ottimizzazione proposto dall'articolo[16], basato sul mini-batch gradient descent è stato modificato con Adam, mantenendo inalterato il valore del learning rate. La Tabella 2.1 mostra alcuni risultati di PSNR ottenuti applicando la rete VDSR su tre diverse immagini per i fattori di scala 2, 3 e 4. Come si può notare, il metodo di ottimizzazione Adam permette di ottenere risultati migliori in termini di PSNR, e questa tendenza è stata riscontrata su tutti i campioni testati.

Nell'analisi finale dei risultati è stato scelto di focalizzarsi principalmente sulle variazioni dovute all'impiego di diversi fattori di scala in fase di testing e sul diverso comportamento delle reti considerate. A tal proposito, le architetture prese in esame saranno:

- la rete VDSR originale, come proposta dagli autori e descritta nella Sezione 2.5.1, con l'unica modifica dell'algorimto di ottimizzazione;

- due reti VDSR ottenute sostituendo i layer di attivazione ReLU con LeakyReLU ed ELU;

	Mini-batch Gradient Descent	ADAM
Scala x2	76.4712	82.4771
Scala x3	72.4000	75.0719
Scala x4	68.1923	70.2369

baby

	Mini-batch Gradient Descent	ADAM
Scala x2	58.3733	64.4102
Scala x3	52.1979	55.9897
Scala x4	47.4099	49.9756

zebra

	Mini-batch Gradient Descent	ADAM
Scala x2	52.7284	54.5638
Scala x3	46.8994	48.3979
Scala x4	42.9464	44.2135

img_040

Tabella 3.1: Confronto di valori PSNR per modelli addestrati con ottimizzazione mini-batch e Adam sulle immagini “baby”, “zebra” e “img040”.

- la rete VDSR fully connected, ottenuta inserendo layer dense tra quelli convoluzionali e di attivazione, nella quale sono state modificate anche le dimensioni dei filtri portandole a 7x7.

Tutte e quattro le reti elencate sono state addestrate mediante apprendimento multi-scala per scale factor pari a 2, 3, 4. Il tempo richiesto per il training è stato di circa un giorno per la rete VDSR originale (ReLU) e quelle con i layer di attivazione modificati (LeakyReLU ed ELU), mentre è raddoppiato (due giorni) per la rete VDSR dense. I risultati riportati nelle sezioni successive fanno riferimento a un test set composto da 20 immagini selezionate fra i dataset “Set5”, “Set14” e “Urban100”. Poichè i valori di PSNR e di SSIM sono molto variabili a seconda dell’immagine che viene di volta in volta considerata, è stato deciso di calcolare le differenze fra i valori ottenuti mediante interpolazione bicubica e quelli ottenuti da ogni rete, in modo tale da confrontare l’incremento percentuale apportato da ciascun

modello sui due parametri di valutazione. Le piattaforme e i linguaggi utilizzati per l'implementazione delle reti e per la raccolta dei dati relativi ai risultati sono state: Python e le sue librerie Tensorflow e Keras per la parte di creazione del modello; Matlab per l'elaborazione delle immagini in fase di preprocessing e testing; R per la raccolta e l'analisi statistica dei risultati.

3.4.1 Ingrandimento x2

In questa sezione verranno riportati i risultati ottenuti dalle reti sopra elencate per il problema di SISR con fattore di scala pari a 2. La Tabella 3.2 riporta i valori di PSNR a confronto per le immagini del test set riportate in Figura 3.1. Si noti come tutte le reti raggiungano risultati migliori rispetto alla bicubica, fatta eccezione della ELU per le immagini "baby", "bird" e "face".

Modello	Baby	Bird	Zebra	Face	Img040	Img075
Bicubica	81.4281	78.5323	61.3523	69.2182	49.1485	54.9322
VDSR ReLU	82.4771	82.8619	64.4102	70.3119	54.5638	54.7716
VDSR Dense	83.8176	84.5970	66.3010	70.9183	59.2649	57.5065
VDSR LeakyReLU	81.8743	82.8220	65.0103	70.4293	55.5094	55.4498
VDSR ELU	80.5986	76.3006	61.6655	68.2349	52.3218	56.7288

Tabella 3.2: Valori di PSNR a confronto per interpolazione bicubica e reti implementate con fattore di scala 2.

La Tabella 3.3, invece, riporta i valori statistici (media, deviazione standard e intervallo di confidenza) relativi agli incrementi apportati a PSNR e SSIM della bicubica da parte delle quattro reti considerate. Per l'intevallo di confidenza si è scelto un livello di confidenza α pari a 0.05, anche per gli altri fattori di scala (3, 4). In questo caso i risultati sono riferiti alle 20 immagini dell'intero test set. Si noti come per entrambi i parametri la rete che permette di ottenere risultati migliori sia la VDSR con layer dense. In particolare essa apporta in media un miglioramento del 7.16% sull'approssimazione ottenuta dall'interpolazione bicubica. Per questo fattore di scala la seconda migliore

rete risulta essere la LeakyReLU, anche se i valori ottenuti da questo modello e dai restanti due si mantengono abbastanza simili fra loro.

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.04190508	0.04534658	0.02852320	0.07160493
SSIM	0.01095375	0.01626154	0.01318587	0.02376355

(a) - Media

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.02773355	0.03091188	0.03222773	0.04413263
SSIM	0.01295783	0.01263745	0.01326084	0.01538863

(b) – Deviazione standard

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	[0.02975054, 0.05405962]	[0.0317991, 0.05889406]	[0.01439904, 0.04264737]	[0.05226331, 0.09094656]
SSIM	[0.00527484, 0.01663267]	[0.01072304, 0.02180004]	[0.007374159, 0.01899758]	[0.01701931, 0.03050779]

(c) – Intervallo di confidenza

Tabella 3.3: Media (a), deviazione standard (b) e intervallo di confidenza (c) delle differenze percentuali fra i risultati di PSNR e SSIM con le reti VDSR e i risultati ottenuti mediante interpolazione bicubica, per fattore di scala 2. I valori in rosso evidenziano i risultati migliori.

Nei grafici della Figura 3.2 sono riportati i risultati degli incrementi di PSNR(a) e SSIM(b) ottenuti su tutte le 20 immagini per le quattro reti. È possibile notare come i punti verdi che rappresentano i valori ottenuti con la rete fully connected si trovino al di sopra di quelli relativi alle altre reti per la maggior parte dei casi, sia per il PSNR, sia per il SSIM. Inoltre con tale rete è stato possibile ottenere picchi di incrementi fino al 20,58% sul PSNR rispetto all'immagine interpolata. Al contrario, le variazioni di SSIM sono state in generale più contenute, con un massimo del 5.35% ottenuto dalla rete dense.

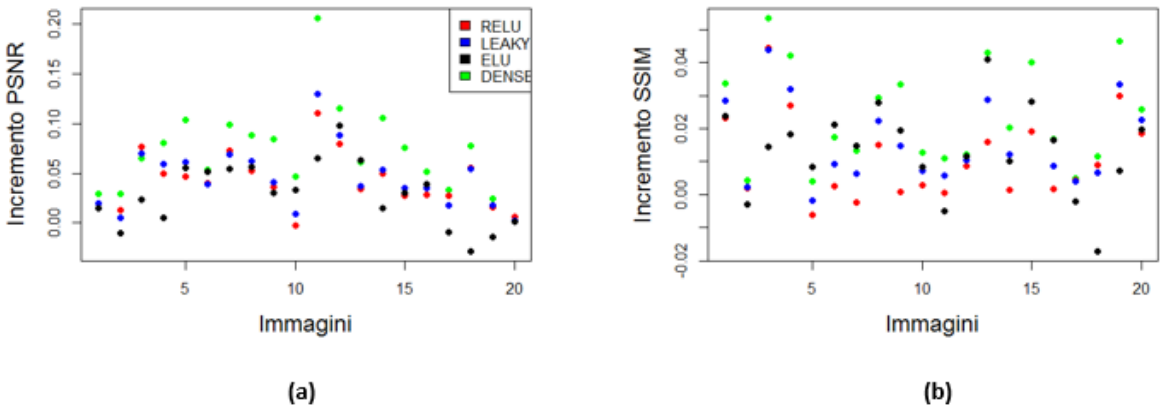


Figura 3.2: Confronto fra i valori di incremento per PSNR(a) ed SSIM(b) per le diverse reti con fattore di scala 2.

3.4.2 Ingrandimento x3

Verranno ora analizzati i risultati ottenuti dalle reti per un fattore di scala pari a 3. Il procedimento per il calcolo dei valori PSNR e SSIM è il medesimo della precedente sezione. In Tabella 3.4 sono mostrati i valori di PSNR ottenuti con i modelli VDSR e con interpolazione bicubica. Anche in questo caso le reti da noi considerate ottengono risultati migliori rispetto alla bicubica, con l'unica eccezione della rete ELU per l'immagine "face".

Modello	Baby	Bird	Zebra	Face	Img040	Img075
Bicubica	73.5359	68.1716	53.1945	65.1170	43.8332	51.2405
VDSR ReLU	75.0719	71.5747	55.9897	66.4099	48.3979	51.2728
VDSR Dense	76.2181	73.7502	58.4998	66.7328	53.9508	53.9108
VDSR LeakyReLU	75.2852	72.6249	57.1004	66.3249	49.4321	51.9332
VDSR ELU	74.6268	70.4230	56.3097	64.9742	48.2020	52.7821

Tabella 3.4: Valori di PSNR a confronto per interpolazione bicubica e reti implementate con fattore di scala 3.

La Tabella 3.5 mette a confronto media, deviazione standard e intervallo di confidenza ottenuti per gli incrementi dei due parametri da parte delle

quattro reti che accettano in input immagini ricampionate con scale factor 3. I risultati sono riferiti all'intero test set. Come si può notare, i valori sono aumentati rispetto al caso in cui il fattore di scala era pari a 2, raggiungendo un miglioramento medio del 8.37% per il PSNR e del 5.75% per il SSIM con la rete dense.

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.04316726	0.05008502	0.03388566	0.08373460
SSIM	0.03242621	0.03882216	0.02978311	0.05757531

(a) – Media

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.0254883	0.03160145	0.02487361	0.05367148
SSIM	0.01715273	0.01811799	0.01603208	0.02675765

(b) – Deviazione standard

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	[0.03199672, 0.05433779]	[0.03623533, 0.06393471]	[0.02298452, 0.0447868]	[0.06021247, 0.1072567]
SSIM	[0.02490883, 0.03994359]	[0.03088175, 0.04676257]	[0.02275687, 0.03680935]	[0.04584846, 0.06930215]

(c) – Intervallo di confidenza

Tabella 3.5: Media (a), deviazione standard (b) e intervallo di confidenza (c) delle differenze percentuali fra i risultati di PSNR e SSIM con le reti VDSR e i risultati ottenuti mediante interpolazione bicubica, per fattore di scala 3. I valori in rosso evidenziano i risultati migliori.

Dai grafici in Figura 3.3 si riscontra lo stesso comportamento del caso precedente: i valori ottenuti con la rete dense risultano migliori per tutte le immagini appartenenti al test set, con un picco del 23,08% per il PSNR e del 10,51% per il SSIM. Le performance ottenute dalle reti ReLU, LeakyReLU ed ELU risultano essere piuttosto simili fra loro nel numero di volte in cui una risulta migliore delle altre, in particolare per quanto riguarda il valore di SSIM. La seconda media migliore resta comunque quella della LeakyReLU.

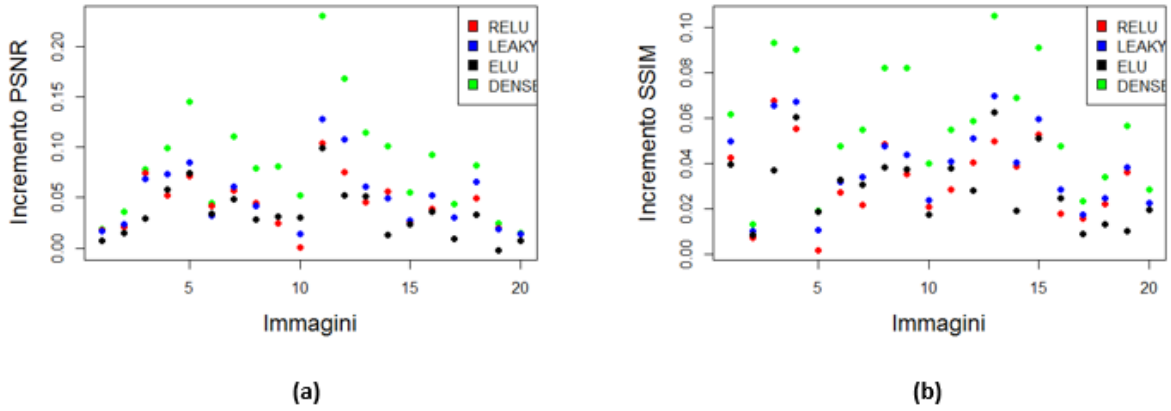


Figura 3.3: Confronto fra i valori di incremento per PSNR(a) ed SSIM(b) per le diverse reti con fattore di scala 3.

3.4.3 Ingrandimento x4

L'ultimo fattore di scala considerato è 4. Ricordiamo che maggiore è il fattore di scala, maggiore è la variazione nelle dimesioni delle immagini in fase di ricampionamento e peggiore è l'approssimazione ottenuta mediante interpolazione bicubica (ci sono più pixel incogniti da ricostruire). La Tabella 3.6 riporta i valori di PSNR delle immagini in Figura 3.1 ottenuti con la bicubica e le quattro reti per questo fattore di scala. In questo caso tutte le reti apportano un incremento positivo al PSNR relativo al metodo di interpolazione bicubica.

Modello	Baby	Bird	Zebra	Face	Img040	Img075
Bicubica	68.2989	61.4871	48.1408	62.3471	40.2577	51.0029
VDSR ReLU	70.2369	64.0484	49.9258	63.5863	44.2135	51.2058
VDSR Dense	71.3166	66.3362	53.8873	63.9726	50.1627	53.9952
VDSR LeakyReLU	71.0156	65.2793	51.2164	63.6142	44.9279	51.8718
VDSR ELU	69.2097	63.1419	49.9756	62.4237	42.2707	52.8718

Tabella 3.6: Valori di PSNR a confronto per interpolazione bicubica e reti implementate con fattore di scala 4.

La Tabella 3.7 mostra i miglioramenti medi ottenuti dalle nostre reti tenendo in considerazione l'intero test set. La rete dense mantiene i risultati migliori anche in questo caso, con un incremento di PSNR medio del 8.55%.

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.04084017	0.04763071	0.022927750	0.08551932
SSIM	0.04837780	0.05669718	0.030881250	0.08958741

(a) -Media

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	0.02404596	0.02806095	0.01571367	0.05896256
SSIM	0.02333038	0.02619088	0.01488203	0.04402315

(b) – Deviazione standard

	VDSR ReLU (originale)	VDSR LeakyReLU	VDSR ELU	VDSR Dense
PSNR	[0.03030176, 0.05137858]	[0.03533268, 0.05992874]	[0.01604106, 0.02981445]	[0.05967832, 0.1113603]
SSIM	[0.038153, 0.0586026]	[0.04521874, 0.06817563]	[0.02435903, 0.03740347]	[0.07029377, 0.1088811]

(c) – Intervallo di confidenza

Tabella 3.7: Media (a), deviazione standard (b) e intervallo di confidenza (c) delle differenze percentuali fra i risultati di PSNR e SSIM con le reti VDSR e i risultati ottenuti mediante interpolazione bicubica, per fattore di scala 4. I valori in rosso evidenziano i risultati migliori.

I grafici di Figura 3.4 mostrano i risultati di PSNR e SSIM relativi alle diverse reti a confronto. La rete VDSR originale e quella con attivazione LeakyReLU hanno performance simili fra loro, mentre con la ELU si ottengono i risultati peggiori.

3.4.4 Considerazioni finali sui risultati

Dai risultati ottenuti da questo la voro di tesi è possibile trarre alcune osservazioni fondamentali.

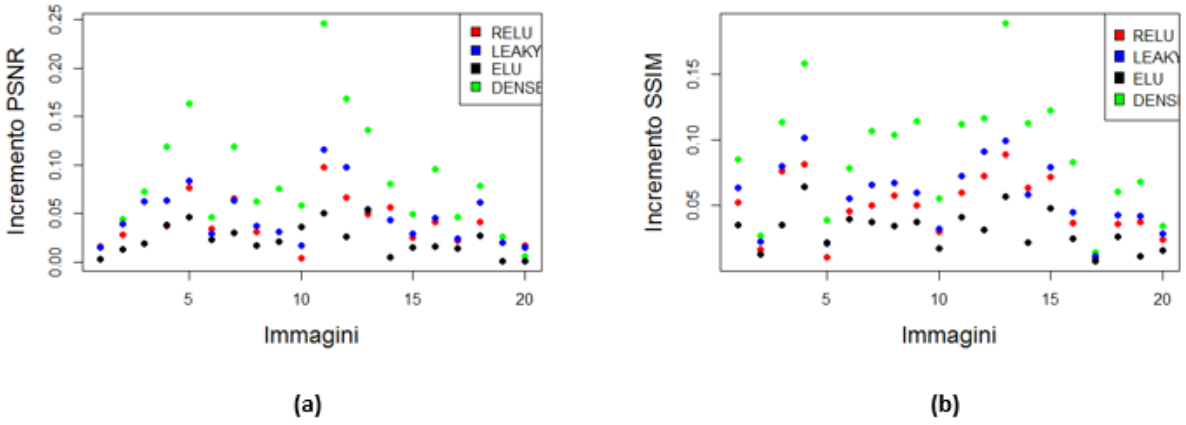


Figura 3.4: Confronto fra i valori di incremento per PSNR(a) ed SSIM(b) per le diverse reti con fattore di scala 4.

Innanzitutto è stata confermata la superiorità dei metodi di SISR basati sul deep-learning nei confronti del metodo bicubico. Tutte le reti implementate hanno infatti apportato variazioni positive ai risultati ottenuti mediante interpolazione bicubica, con rare eccezioni per la rete con layer di attivazione ELU.

Tra le quattro reti proposte, la migliore si è rivelata essere la VDSR modificata con l'inserimento di layer fully connected: essa ha riportato performance medie superiori alle altre di almeno tre punti percentuali per ogni fattore di scala.

A tal proposito, i miglioramenti visivamente e numericamente più consistenti sono stati ottenuti per il fattore di scala 4: per questo motivo le immagini qui riportate come esempi (Figure 3.5, 3.6, 3.7) sono state generate tenendo in considerazione questo parametro di ricampionamento, in modo da rendere il più possibile riconoscibili gli effetti apportati dalla super-risoluzione. Dalla tabella 3.7 notiamo infatti come rispetto ai fattori di scala 2 e 3, ci sia stato uno spostamento degli intervalli in senso positivo, con incrementi massimi raggiunti del 24,60% per il PSNR e del 18,8% per il SSIM,

entrambi riferiti alla rete VDSR dense.

Un'ultima importate considerazione riguarda la variazione della performance della rete VDSR dense a seconda della tipologia di immagini sottoposte alla rete. Come anticipato, sono state inserite nel test set immagini provenienti da diversi dataset, tra i quali "Urban100". Le immagini in esso contenute rappresentano soggetti urbani (palazzi, strade, finestre, ecc.) caratterizzati dalla presenza di numerose linee nette difficili da ricostruire in modo accurato per il problema dell'aliasing. La rete VDSR dense ha tuttavia ottenuto i suoi risultati migliori proprio per queste particolari immagini. La Tabella 3.8 mostra a confronto i valori medi degli incrementi di PSNR rispetto alla bicubica calcolati su 10 immagini appartenenti solamente a questo dataset.

	Test set completo	10 immagini da Urban100
x2	0.07160493	0.09447369
x3	0.08373460	0.10761260
x4	0.08551932	0.10961500

Tabella 3.8: Valori medi di incremento sul PSNR a confronto per immagini del dataset Urban100.

Di seguito saranno mostrati esempi di immagini appartenenti al test set nelle varie fasi del processo di super risoluzione. In particolare verranno confrontate l'immagine HR originale, l'immagine LR ricampionata con fattore di scala 4, l'immagine ricostruita mediante interpolazione bicubica e mediante le 4 reti considerate. È possibile notare come la rete dense risulti molto efficace per immagini contenenti dettagli lineari e contorni definiti, e ciò è particolarmente evidente per le immagini in Figura 3.5 e 3.7.

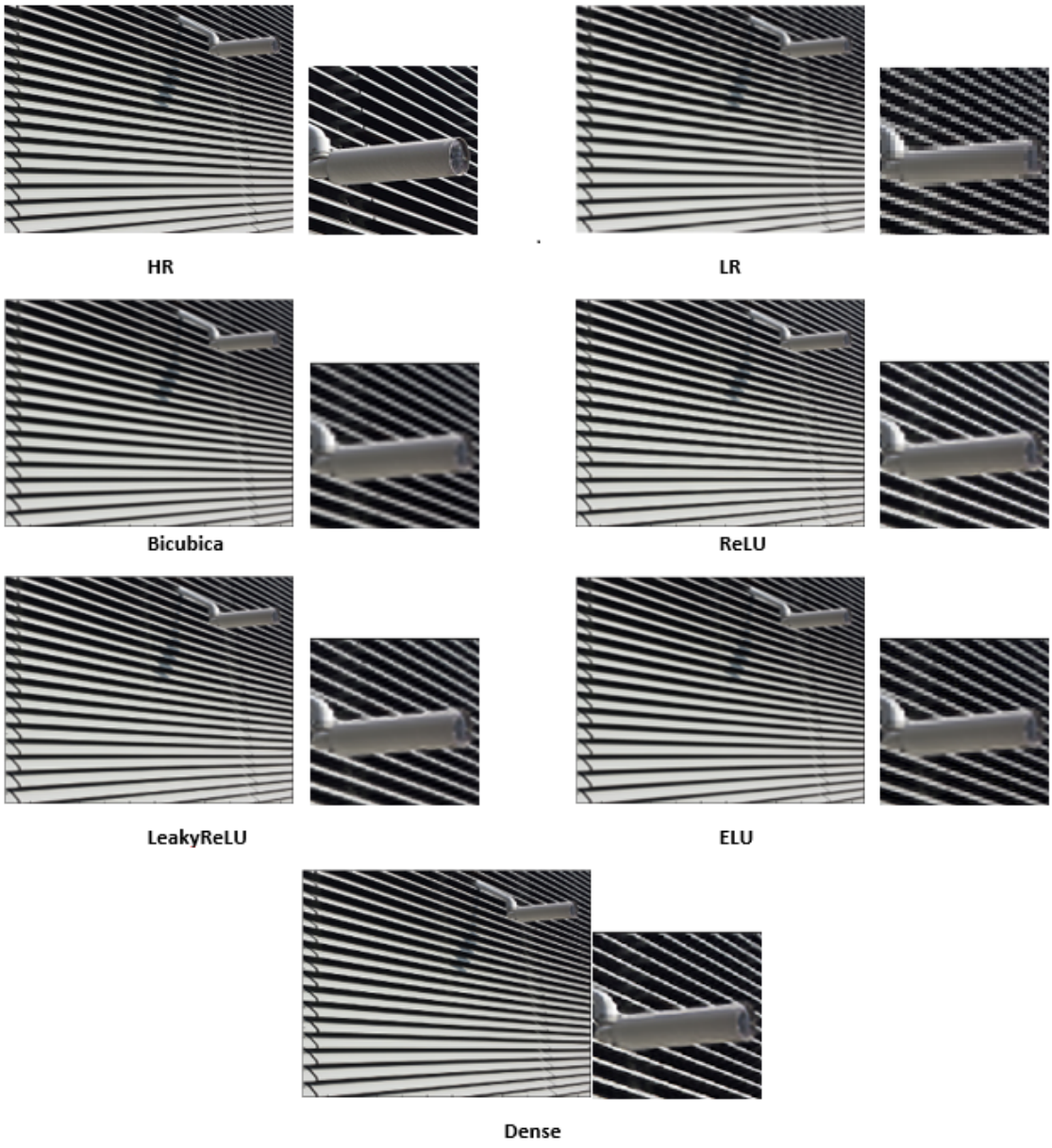


Figura 3.5: Confronto fra dettagli dell'immagine originale HR, ricampionamento LR, ricostruzione mediante bicubica, VDSR ReLu, LeakyReLU ed ELU e VDSR dense. Immagine "img40" dal dataset Urban100.

Figura 3.6: Confronto fra dettagli dell'immagine originale HR, ricampionamento LR, ricostruzione mediante bicubica, VDSR ReLu, LeakyReLU ed ELU e VDSR dense. Immagine "baby" dal dataset Set5.

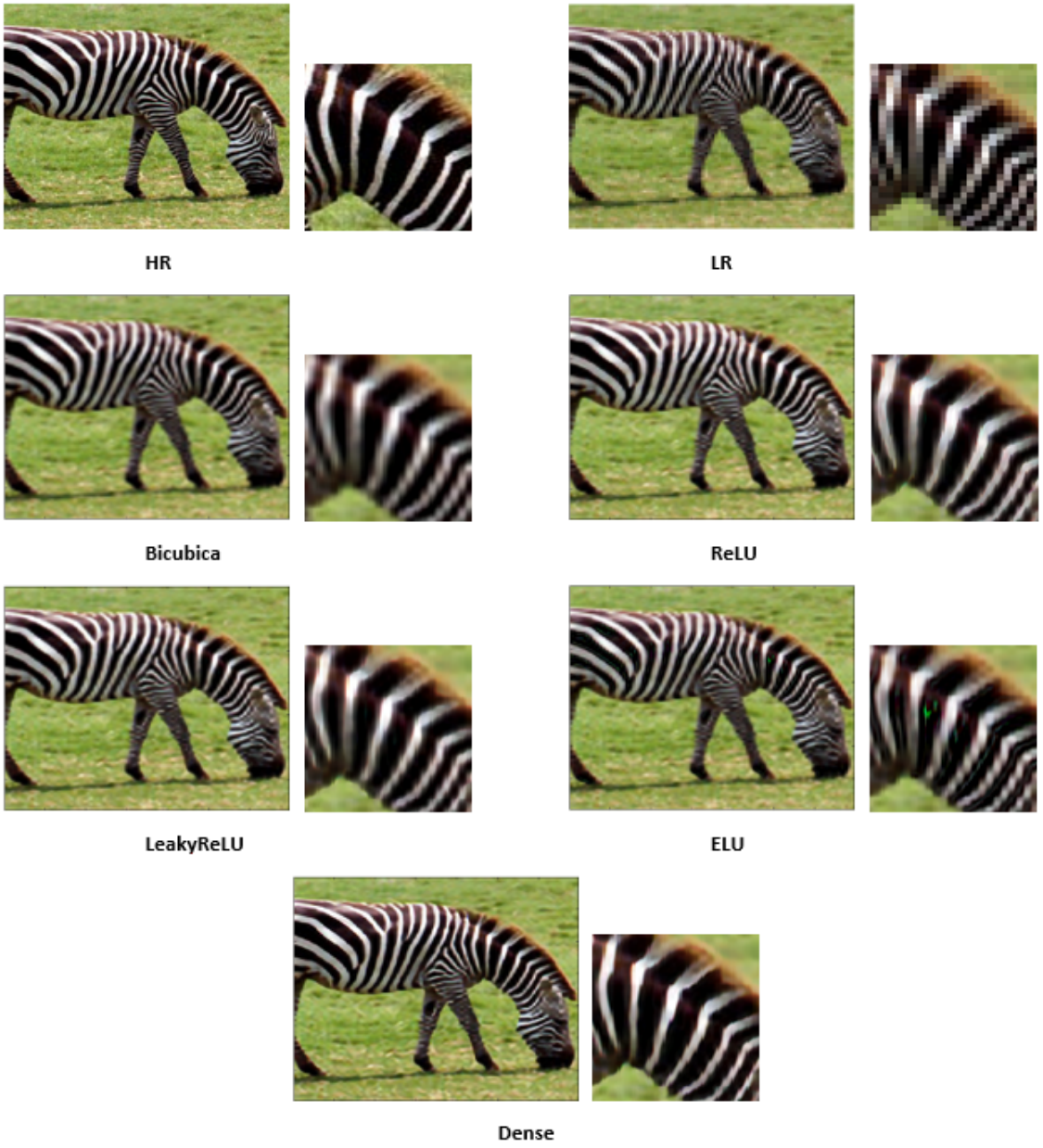


Figura 3.7: Confronto fra dettagli dell'immagine originale HR, ricampionamento LR, ricostruzione mediante bicubica, VDSR ReLu, LeakyReLU ed ELU e VDSR dense. Immagine "zebra" dal dataset Set14.

Conclusioni

In questo lavoro di tesi si è tentato di fornire un'accurata panoramica sul problema di super-risoluzione e sui modelli ad oggi più diffusi per risolverlo. L'intento finale, tuttavia, è stato quello di cimentarsi nella modellazione di nuove architetture di reti neurali a partire dalla rete VDSR selezionata come punto di partenza.

Sicuramente sono stati raggiunti risultati soddisfacenti, in particolare con la rete VDSR modificata inserendo un nuovo tipo di layer di attivazione (LeakyReLU ed ELU) e quella ottenuta aggiungendo layer fully connected (VDSR dense). Se alterando la funzione di attivazione si sono raggiunti risultati migliori in termini di SSIM, ma equiparabili alla rete originale in termini di PSNR, con la rete dense è stato possibile ottenere un incremento notevole per entrambi i criteri di valutazione. Questo miglioramento della performance ha richiesto una perdita in tempi di computazione nella fase di addestramento (due giorni a fronte di un solo giorno impiegato dalla rete VDSR originale), ma in misura decisamente trascurabile in confronto ai risultati ottenuti.

Sicuramente il modello VDSR potrà essere soggetto a ulteriori studi e perfezionamenti. Una direzione percorribile potrebbe essere quella di incrementare ulteriormente il numero di layer mantenendo la struttura fully connected, prestando attenzione tuttavia a eventuali rallentamenti dell'addestramento.

Un'altra alternativa potrebbe essere l'inserimento di layer che operano ricorsivamente ispirati al modello DRCN, per sfruttare i risultati dei livelli intermedi della rete.

Dal punto di vista del dominio delle immagini da sottoporre alla rete,

sarebbe interessante proseguire nella direzione accennata nella parte conclusiva, cercando altre categorie di immagini o dataset alternativi sui quali la rete VDSR dense risulti particolarmente effciente, per restringerne il campo applicativo e concentrarsi su suoi possibili miglioramenti in tale direzione.

Bibliografia

[1] C. Y. Yang, C. Ma, M. H. Yang, "Single-image super-resolution: A benchmark", Proc. Eur. Conf. Comput. Vis., pp. 372-386, 2014.

[2] J. Sun, Z. Xu, and H.-Y. Shum, "Image super-resolution using gradient profile prior", in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1-8.

[3] Farsiu, S., Robinson, D., Elad, M., Milanfar, P.: "Advances and Challenges in Super-Resolution", International Journal of Imaging Systems and Technology 14(2), 47-57 (2004); special issue on high-resolution image reconstruction

[4] W. Yang, X. Zhang, Y. Tian, W. Wang, J.H. Xue, "Deep Learning for Single Image Super-Resolution: A Brief Review", ArXiv2018

[5] J. Schmidhuber, "Deep learning in neural networks: An overview", Neural networks, vol. 61, pp. 85-117, 2015.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", nature, vol. 323, no. 6088, p. 533, 1986.

[7] I. Gogul, V. S. Kumar, "Flower species recognition system using convolution neural networks and transfer learning", 2017 Fourth International Conference on Signal Processing Communication and Networking (ICSCN), pp. 1-6, March 2017

[8] A.Hore, D. Ziou, "Image quality metrics: PSNR vs. SSIM", Proc. ICPR, pp. 2366-2369, 2010.

[9] Dabal Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task", Machine Learning (cs.LG), 2018, Vol. 1.

[10] Kingma, Diederik P. and Ba, Jimmy. "Adam: A Method for Stochastic Optimization", arXiv:1412.6980 [cs.LG], December 2014.

[11] R. Keys, "Cubic convolution interpolation for digital image processing", IEEE transactions on acoustics, speech, and signal processing, vol. 29, no. 6, pp. 1153-1160, 1981.

[12] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview", IEEE signal processing magazine, vol. 20, no. 3, pp. 21-36, 2003.

[13] C. Dong, C. C. Loy, K. He, and X. Tang. "Image super-resolution using deep convolutional networks", TPAMI, 2015.

[14] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution", in European Conference on Computer Vision. Springer, 2014, pp. 184-199.

[15] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network", in European Conference on Computer Vision. Springer, 2016.

[16] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[17] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution", in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016

[18] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution", in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, vol. 1, no. 2, 2017.

[19] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution", in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition", in ICLR, 2015.

[21] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult. Neural Net- works", IEEE Transactions on, 5(2):157-166, 1994.

[22] J. Yang, J. Wright, T. S. Huang, and Y. Ma. "Image superresolution via sparse representation", TIP, 2010.

[23] C. G. Marco Bevilacqua, Aline Roumy and M.-L. A. Morel. "Low-complexity single-image super-resolution based on nonnegative neighbor embedding", in BMVC, 2012.

[24] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics", in ICCV, 2001.

[25] R. Zeyde, M. Elad, and M. Protter. "On single image scale-up using sparse-representations", in Curves and Surfaces, pages 711-730. Springer, 2012.

[26] J.-B. Huang, A. Singh, and N. Ahuja. "Single image superresolution using transformed self-exemplars", in CVPR, 2015.