Alma Mater Studiorum · Università di Bologna

Scuola di Scienze

Dipartimento di Fisica e Astronomia

Corso di Laurea in Fisica

Architetture di deep learning per l'imaging medico del tumore alla prostata

Relatore:

Prof. Gastone Castellani

Presentata da:

Matteo Campanini

Anno Accademico 2017/2018

Sommario

Questa tesi vuole essere un'esposizione sintetica delle architetture di deep learning applicate all'imaging medico, con particolare attenzione all'analisi di immagini TRUS e MRI multiparametriche per la diagnosi automatica del tumore alla prostata. Il lavoro è diviso in due parti. Nella prima parte vengono prima presentati i principali concetti teorici del machine learning per poi analizzare nel dettaglio tre delle architetture di deep learning più utilizzate: la rete neurale profonda, la rete neurale convoluzionale e vari tipi di autoencoder. Nella seconda parte viene mostrato come le tecnologie di deep learning vengono utilizzate nell'analisi delle immagini mediche, per poi analizzare le architetture di deep learning utilizzate nei sistemi automatici di diagnosi del tumore alla prostata da analisi di immagini TRUS e MRI multiparametriche.

Indice

Capitolo 1

Deep Learning

Con Machine Learning si intende un insieme di tecniche e approcci di natura statistico-computazionale di risoluzione di determinati tipi di problemi che prevedono che l'algoritmo impari da un certo set di dati. Lo studio degli algoritmi di machine learning è un campo di ricerca oggi molto attivo che ha ottenuto molti successi nel risolvere problemi difficili da formulare esattamente e risolti intuitivamente da un essere umano come il riconoscimento delle immagini o la comprensione del linguaggio parlato. Tali compiti infatti sono molto difficili da risolvere con programmi "hard coded" il cui funzionamento non prevede un addestramento. In questo approccio i programmi esposti all'addestramento imparano una gerarchia di concetti dai dati. L'apprendimento di concetti si può anche chiamare apprendimento di rappresentazioni dei dati. L'apprendimento di rappresentazioni è un sottoinsieme delle tecniche di machine learning, e l'autoencoder è il tipo di architettura principe di queste tecniche. Se poi si apprendono rappresentazioni da altre rappresentazioni, avremo una stratificazione profonda dell'apprendimento: il deep learning. In Figura 1.1 è mostrato un diagramma che mostra la relazione concettuale tra queste diverse tecniche.

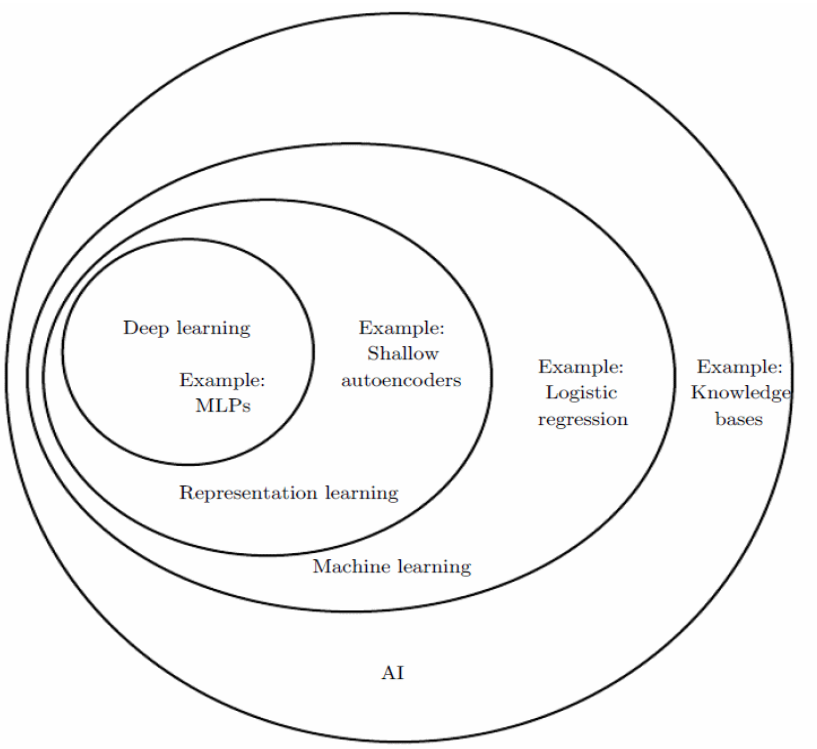

Figura 1.1: Un diagramma di Venn che rappresenta la relazione concettuale tra le tecniche descritte. Come si vede il deep learning è un caso particolare di apprendimento di rappresentazione, che a sua volta è un sottoinsieme del machine learning e il tutto è una sottodivisione delle tecnologie di intelligenza artificiale. Da [1]

1.1 Machine learning

Che un algoritmo o un programma impari significa che, con le parole di Mitchell(1997): "Un programma impara dall'esperienza E rispetto ad un compito C con una misura di performance P, se la sua performance, misurata da P, rispetto a C, migliora con l'esperienza E".

1.1.1 Il compito

Alcuni *compiti* che gli algoritmi di machine learning possono essere per esempio:

Classificazione In un compito di classificazione si chiede all'algoritmo di collocare un input in una delle categorie fornite. L'operazione è formalmente equivalente al produrre ("imparare") una funzione $f : \mathbb{R}^n \rightarrow \{0, 1, ..., k\}$ dove n è la dimensione dell'input e l'insieme $\{0, 1, ..., k\}$ rappresenta l'insieme delle possibili classi a cui un certo input può appartenere. Un esempio di classificazione è una delle prime reti neurali convoluzionali implementate con successo: LeNet5, il cui compito è riconoscere delle lettere o cifre scritte a mano. Uno dei set di dati più utilizzati per il testing degli algoritmi di deep learning è il MNIST dataset, mostrato in Figura 1.2, che è formato da immagini raffiguranti cifre da 0-9 scritte a mano. Il compito dell'algoritmo che viene addestrato su questo set di dati è di riconoscere la cifra scritta a mano nell'immagine (prendere come input un'immagine e assegnarle un elemento dell'insieme 0,1,2,3,4,5,6,7,8,9).

Classificazione con input mancanti In un problema di classificazione può presentarsi il problema di non avere disponibili tutte le misurazioni necesssarie per la produzione della funzione richiesta dal problema di classificazione. In generale il problema di classificazione con input mancanti comprende due fasi: la gestione dei dati mancanti e l'effettiva classificazione. In base a come si risolvono i due problemi si può raggruppare la maggior parte delle techniche in quattro tipi:

1. Si ignorano gli input con dati mancanti, ovvero si procede alla classificazione con gli *input completi* a disposizione

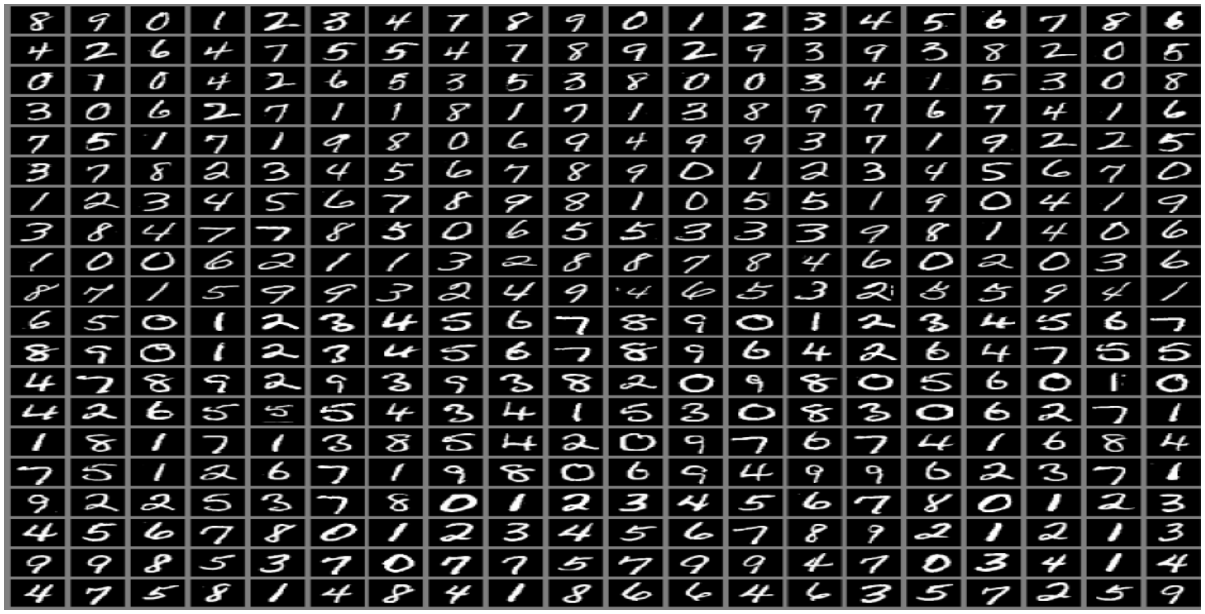

Figura 1.2: Alcuni esempi del MNIST data set, formato da immagini raffiguranti cifre scritte a mano. Da [1]

2. Imputazione (stima e rimpiazzamento) dei valori mancanti in input, seguito dalla normale classificazione con input completi

3. Utilizzo dell'approcio ML (Maximum Likelihood), dove la distribuzione di dati in input è modellata dall'algoritmo EM (Expectation-Maximization), seguito da classificazione basata sulla regola di Bayes

4. Utilizzo di una procedura di machine learning che gestisce gli input mancanti senza imputazione esplicita.

Il problema della classificazione con input mancanti è di cruciale interesse in quanto si hanno input incompleti in molti campi di applicazione. Le cause della mancanza di dati in input possono essere di diversa natura: per esempio nelle applicazioni mediche alcuni test non possono essere effettuati sul paziente perchè troppo invasivi o dispendiosi.

Gestione di input incompleti Come detto nella parte precedente, un algoritmo di apprendimento può stimare i valori mancanti nei vettori di input. Il problema della gestione dei valori mancanti in input è di grande importanza in molte applicazioni in cui non si ha a disponibilità una grande mole di dati, ciò è quasi sempre il caso nelle applicazioni mediche. In [2] vengono valutate le performance di alcune tecniche di imputazione su un dataset per la previsione della probabilità di sopravvivenza dopo un intervento per la rimozione del tumore al seno. Nell'articolo vengono illustrati tre algoritmi usati per l'imputazione di questo dataset: un multi-layer-perceptron, una Self-organizing map e un K-nearest neighbours.

Regressione La regressione può essere visto come una classificazione con output continuo: l'algoritmo deve produrre dall'input una funzione $f : \mathbb{R}^n \to \mathbb{R}$. L'esempio più semplice di regressione è la **regressione lineare**. In questo tipo di problema si fornisce all'algoritmo un certo input x e si richiede di associare tale input con un numero reale: in questo caso l'output $\hat{y}$ è una funzione lineare dell'input:

$$\hat{y} = \mathbf{w}^T\mathbf{x} + \mathbf{b}$$

dove $\mathbf{w}$ è un vettore di parametri, detti anche pesi, della stessa dimensione del vettore input e $\mathbf{b}$ è il vettore bias, sempre della stessa dimensione dell'input. L'esempio della regressione lineare è approfondito più avanti.

1.1.2 L'esperienza

Come esperienza di un algoritmo di apprendimento si intende un l'addestramento dell'algoritmo su un determinato dataset. Il dataset è una collezione di vettori dati in input, in cui ogni componente del vettore è un valore assegnato ad una certa proprietà del campione. Uno dei primi dataset studiati con tecniche di machine learning è l'**Iris dataset**, un dataset contenente le misure di diverse parti di 150 piante di Iris, come la

lunghezza dei petali e del gambo, e in cui viene anche specificato la specie della pianta. Si dividono in generale i tipi di addestramento degli algoritmi di apprendimento in due macrocategorie:

1. **Algoritmi ad apprendimento supervisionato**
   In questo caso ad ogni esempio del dataset è associato un certo target che l'algoritmo dovrà poi predire nel testing e nell'applicazione. Nell'Iris dataset all'algoritmo è chiesto di predire la specie della pianta dalle misure fornite. Il termine "supervisionato" deriva dal fatto che l'atto di etichettare i vettori input veniva eseguito da un operatore umano che facesse da "insegnante". L'apprendimento supervisionato consiste nell'osservare un certo numero di vettori casuali x, associati ad un certo valore y, e stimare $p(y|x)$.

2. **Algoritmi ad apprendimento non supervisionato**
   A questi algoritmi è richiesto di trovare da solo delle proprietà utili dalla struttura del dataset fornito. Uno dei compiti che in cui il training è non supervisionato è il **clustering**, in cui l'algoritmo deve dividere il dataset in base alle proprietà degli esempi. L'apprendimento non supervisionato consiste nell'osservare un certo numero di vettori casuali x e determinare la distribuzione $p(x)$, o qualche sua proprietà.

I due tipi di apprendimento non sono formalmente definiti, molti algoritmi di apprendimento sono in grado di risolvere entrambi i tipi di problemi. Per la regola della catena della probabilità abbiamo che per un vettore $x \in \mathbb{R}^n$ la probabilità congiunta $p(x)$ può essere decomposta come:

$$p(x) = \prod_{i=1}^{N} p(x_i|x_1, ..., x_{i-1})$$

che equivale a dire che un problema il cui training è non supervisionato può essere scomposto in N problemi il cui training è supervisionato e viceversa si può scomporre

un problema di apprendimento supervisionato di $p(y, x)$ sapendo che:

$$p(y, x) = \frac{p(x, y)}{\sum_{y'} p(x, y')}$$

Esistono anche altre varianti di apprendimento, come l'apprendimento semisupervisionato, dove solo alcuni esempi sono provvisti di etichetta. La maggior parte degli algoritmi di apprendimento è addestrato comunque su un dataset, che è una collezione di esempi che a loro volta sono collezioni di features. Un modo comune per descrivere un dataset è con una **matrice design**, che è semplicemente una matrice le cui righe sono i vettori esempio in input, e ogni colonna rappresenta una feature. L'Iris dataset, per esempio, contiene 150 esempi con 4 features per esempio, e quindi è rappresentato da una matrice design $\boldsymbol{X} \in \mathbb{R}^{150\times4}$.

1.1.3 La misura della performance

Per valutare la performance dell'algoritmo su un compito specifico serve una misura quantitativa delle sue abilità. Nel caso dei compiti di classificazione si parla di *accuratezza* che è semplicemente la frazione di esempi per cui l'output dell'algoritmo è corretto. Certe volte la scelta della misura della performance non è ovvia, per esempio in un compito di trascrizione si può scegliere di valutare l'abilità di trascrizione di intere frasi o di parti della frase.

1.1.4 Un esempio: la regressione lineare

Uno degli algoritmi fondamentali è la regrassione lineare. Il modello risolve un problema di regressione in cui l'output è una funzione lineare dell'input. Detto meglio: l'algoritmo prendere in input un vettore $\boldsymbol{x} \in \mathbb{R}$ e predice il valore di uno scalare y che è il valore di output. Chiamiamo $\hat{y}$ il valore corretto che l'algoritmo deve predire (ovvero le "etichette" del set di training supervisionato). Definiamo l'output come:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$$

dove $\boldsymbol{w}$ è un vettore di parametri della stessa dimensione del vettore di input $\boldsymbol{w} \in \mathbb{R}$. Il vettore $\boldsymbol{w}$ è detto anche dei pesi. Esso controlla il comportamento dell'algoritmo di apprendimento. In questo caso l'elemento w_i di $\boldsymbol{w}$ viene moltiplicato per l'elemento x_i di $\boldsymbol{x}$. Ci serve ora una misura della performance dell'algoritmo. Avremo bisogno di un certo numero di esempi dalla matrice in input per formare il test set e di un vettore di target con i risultati ottimali che l'algoritmo dovrebbe prevedere. Chiamiamo rispettivamente $\boldsymbol{X}^{test}$ la matrice di esempi per il testing e $\boldsymbol{y}^{test}$ il vettore target. Un modo per calcolare la performance dell'algoritmo è valutare l'errore qudratico medio tra il target corretto e la previsione dell'algoritmo $\hat{\boldsymbol{y}}^{test}$

$$MSE_{test} = \frac{1}{m} \sum_i \left(\hat{\boldsymbol{y}}^{test} - \boldsymbol{y}^{test} \right)_i^2$$

che corrisponde anche alla distanza euclidea tra la predizione del programma e il target. L' algoritmo di apprendimento dunque modifica i pesi $\boldsymbol{w}$ in modo tale da ridurre l'errore MSE_{test} durante la fase di apprendimento, ovvero quando l'algoritmo osserva il set di training, formato da $\boldsymbol{X} * train$ e $\boldsymbol{y}^{train}$ che sono rispettivamente la matrice degli esempi e il vettore di target per l'addestramento. Un semplice metdodo per minimizzare l'errore quadratico medio per il training è quello di risolvere per il gradiente uguale a zero rispetto ai pesi:

$$\nabla_{\boldsymbol{w}} MSE_{train} = 0$$

$$\rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \left|\left| \hat{\boldsymbol{y}}^{train} - \boldsymbol{y}^{train} \right|\right|_2^2 = 0$$

$$\rightarrow \left(\boldsymbol{X}^{train} \boldsymbol{w} - \boldsymbol{y}^{train} \right)^T \left(\boldsymbol{X}^{train} \boldsymbol{w} - \boldsymbol{y}^{train} \right) = 0$$

$$\rightarrow 2\boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)} \boldsymbol{w} - 2\boldsymbol{X}^{(train)T} \boldsymbol{y}^{(train)} = 0$$

$$\rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(train)T} \boldsymbol{X}^{(train)} \right)^{-1} \boldsymbol{X}^{(train)T} \boldsymbol{y}^{(train)}$$

Il sistema di equazioni risolto dall'ultima relazione è detto **equazioi normali**. In generale il termine regressione lineare si riferisce al modello in cui vi è anche un termine di bias b:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b$$

1.1.5 Capacità, Underfitting e overfitting

Ciò che si vuole da un algoritmo di apprendimento non è solo di massimizzare la performance su un certo set di dati, ma anche di funzionare bene su nuovi input al di fuori del set di dati di addestramento. Nell'esempio della regressione lineare, ciò che si è minimizzato è il *training error*, ovvero l'errore sul *training set* cioè il set di dati per l'addestramento.

$$\frac{1}{m^{train}} ||\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}||_2^2$$

Tuttavia noi siamo interessati a minimizzare il *test error* su un opportuno *test set*.

$$\frac{1}{m^{test}} ||\mathbf{X}^{test}\mathbf{w} - \mathbf{y}^{test}||_2^2$$

La capacità di un algoritmo di funzionare bene su nuovi input è detta **generalizzazione** e il test error è anche detto **generalization error**. Per fare in modo di minimizzare l'errore di generalizzazione potendo solo osservare il training set bisogna necessariamente sapere qualcosa su come formare il set di training e il set di testing dai dati disponibili. Si fanno un certo numero di assunti chiamati **i.i.d. assumption**, che seignifica **independent identically distributed**, cioè che gli esempi in ogni dataset sono **indipendenti**

gli uni dagli altri, e che il train set e test set sono **distribuiti identicamente**, ovvero presi dalla stessa distribuzione di probabilità. Un algoritmo di apprendimento ha test error maggiore o uguale al training error, e ha una buona performance quando:

1. Il training error è piccolo
2. Il gap tra il training error e il test error è piccolo

questi due fattori corrispondono ai due problemi tra i più comuni nel machine learning: **l'underfitting** e **l'overfitting**. L'underfitting corrisponde alla situazione in cui l'errore di training non è sufficientemente piccolo , l'overfitting a quella in cui il gap tra l'errore di training e di generalizzazione non è stato minimizzato sufficientemente. Il fatto che una delle due situazioni si presenti può essere controllato dalla **capacità** del modello. Come capacità si intende informalmente il numero funzioni che il modello può fittare. Se prendiamo in considerazione l'esempio della regressione lineare, il modello è teoricamente capace di riconoscere tutte le funzioni lineari che gli vengono date in input. Si può anche dire che il suo spazio delle ipotesi è quello delle funzioni lineari.

$$\hat{y} = b + wx$$

Ora se aggiungiamo un x^2 al modello lineare avremo un modello capace di fittare anche funzioni quadratiche, abbiamo esteso il suo spazio delle ipotesi:

$$\hat{y} = b + w_1 x + w_2 x^2$$

Possiamo da qui aggiungere potenze di x. Gli algoritmi di apprendimento funzionano al meglio generalmente quando la loro capacità è adatta alla complessità del problema da risolvere. In generale una capacità troppo bassa potrebbe non raggiungere il livello di complessità desiderato e causare underfitting, mentre una troppo alta può risolvere problemi più complessi ma potrebbe portare all'overfitting. In Figura 1.3 è mostrato un semplice esempio di cosa potrebbe significare avere a che fare con un problema di underfitting o overfitting. In questo esempio la distribuzione di esempi da fittare è di tipo quadratico, e proviamo a fittare i punti con un modello lineare, uno quadratico

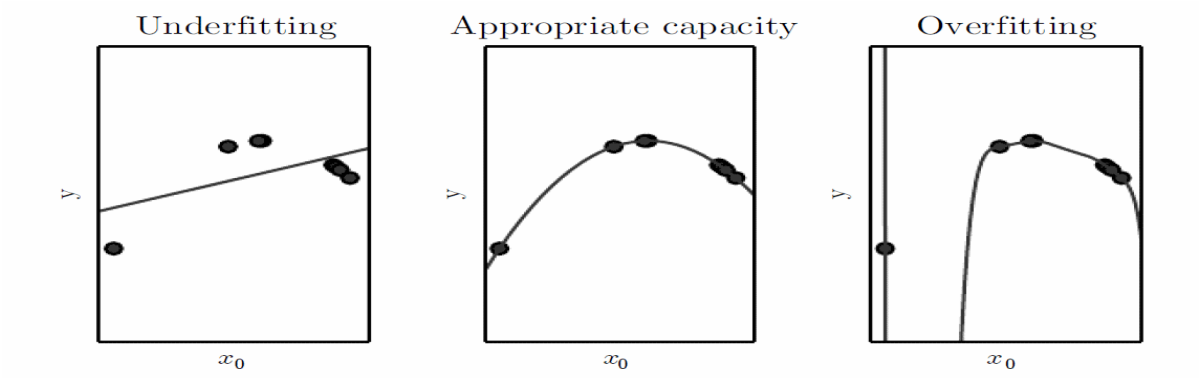

Figura 1.3: Tre risultati di una regressione su un dataset preso da una distibuzione quadratica effettuata con algoritmi con capacità diversa. *Sinistra*) La regressione effettuata da un modello lineare. La capacità è troppo bassa per la distribuzione da imparare. Questo è un tipo caso di underfitting. *Centro*) La regressione effettuata un modello quadratico. Si nota come la capacità del modello è quella adatta al problema. *Destra*) La regressione effettuata da un modello polinomiale di grado 9. Si nota come il modello ha imparato delle features troppo specifiche del training set. Questo è un esempio di overfitting. Da [1].

e uno di grado 9. Come si vede il primo modello non riesce a riconoscere la funzione quadratica poichè le funzioni quadratiche sono al di fuori del suo spazio delle ipotesi, causando underfitting, mentre l'ultimo riconosce un grado di complessità troppo elevato e troppo specifico per il training set, situazione corrispondente all'overfitting. Quando il modello è quadratico il dataset è fittato correttamente: la capacità è adatta al problema.

1.1.6 Iperparametri e sets di validazione

Gli algoritmi di apprendimento hanno alcuni parametri che possono variare il loro comportamento e non sono modificati dal training. Essi sono detti **iperparametri**. Il grado di un modello polinomiale è un esempio di iperparametro. In generale la capacità di un modello è un iperparametro cruciale. Se tentassimo di ottimizzare la capacità di un modello sul training set, l'algoritmo la massimizzerebbe, risultando in overfitting. Per

risolvere questo problema ci serve un set di validazione che l'algoritmo non osserva come training. I set di validazione devono essere formati da esempi dal training set, poichè è importante che gli esempi del test set non siano utilizzati per prendere decisioni sul modello da utilizzare. In particolare si dividono i dati di training in due gruppi, e uno di essi è usato come set di validazione usato per stimare l'errore di generalizzazione per poi modificare gli iperparametri. Generalmente si utilizza il 20% dei dati di training per formare il set di validazione. Dopo l'ottimizzazione degli iperparametri si può procedere alla stima dell'errore di generalizzazione con il test set.

1.1.7 Discesa del gradiente e discesa del gradiente stocastica

La discesa del gradiente è una tecnica di ottimizzazione dei parametri che, presa la loss function del modello, ottimizza i parametri dell'algoritmo per minimizzare tale funzione usando come guida il gradiente rispetto ai parametri e bias, trovando un minimo della funzione. Se abbiamo una certa loss function **L** (per ogni punto del training set) e il vettore di parametri da ottimizzare $\boldsymbol{\theta}$, allora si calcola

$$\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

e si aggiorna il valore di θ nel seguente modo:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Dove ϵ è detto passo e definisce di quanto l'algoritmo debba "spostare" il valore di θ nella direzione opposta a quella del gradiente. Un algoritmo di importanza cruciale per lo sviluppo di algoritmi di deep learning è la discesa stocastica del gradiente, che è una estensione della normale discesa del gradiente. Il problema dell'approccio normale è che nel caso si abbia una gran quantità di dati esso richieda un calcolo molto lungo. La discesa stocastica del gradiente invece di calcolare il gradiente utilizzando tutti i dati a disposizione, ne fa una stima prendendo in considerazione solo un sottoinsieme di esempi,

detto **minibatch** preso uniformemente dal training set di dimensione m' piccola rispetto a quella del set totale m. La stima del gradiente è

$$\hat{\boldsymbol{g}} = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

La discesa del gradiente stocastica dunque sposta il valore di $\boldsymbol{\theta}$ nella direzione opposta al gradiente di una quantità opportuna ϵ

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$$

1.2 Reti neurali feedforward

Le reti neurali feedforward, o multilayer perceptrons, sono i moodelli di deep learning di riferimento. Lo scopo della rete neurale è di approssimare una certa funzione $f^*((x))$. La rete definisce una funzione $f(\boldsymbol{x}, \boldsymbol{\theta})$ e impara i parametri θ per approssimare meglio la funzione. I modelli sono detti feedforward poichè l'informazione fluisce dal livello di input a quello di output passando per tutti i layer intermedi, senza feedback. Una rete in cui è presente un feedback dell'informazione è detta rete ricorrente (Recurrent neural network). Il modello è detto **rete** poichè compone diverse funzioni. Per esempio il modello potrebbe combinare tra diverse funzioni $f^{(3)}$, $f^{(2)}$ e $f^{(1)}$ per formare $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. Il numero di funzioni combinate è detta **profondità** della rete e chiamiamo le funzioni combinate **layers** della rete. In questo caso $f^{(1)}$ è il primo layer, $f^{(2)}$ è il secondo e $f^{(3)}$ il terzo. Da qui viene il termine deep learning. Il modo di utilizzare gli hidden layer è decisione dell'algoritmo: il loro utilizzo non è specificato nè dal programmatore e neanche dal tipo di dato in input. Gli hidden layers sono i layers il cui output non è specificato dai dati in input. Il numero di nodi in ogni layer è detto larghezza del layer. Le reti neurali sono dette neurali perchè la loro struttura è ispirata al modello neuroscientifico del funzionamento del cervello. Infatti i nodi della rete sono anche detti neuroni, poichè ricevono input da molte unità e ognuno calcola un suo valore in output. Tuttavia lo studio delle reti neurali è di area prettamente matematico-ingengeristica e lo scopo di tali reti non è di modellare il funzionamento del cervello.

1.2.1 Un esempio: imparare XOR

Introduciamo un semplice esempio di funzionamento di una rete neurale feed forward con il compito di imparare la funzione XOR. La funzione XOR è una funzione logica a due variabili che restituisce 1 quando solo una delle due variabili è uno, e restutisce 0 negli altri casi. Chiamiamo la funzione che vogliamo far apprendere dal programma $y = f^*(\boldsymbol{x})$, il nostro modello fornirà una funzione $y = f(\boldsymbol{x}; \boldsymbol{\theta})$ e cercherà di modificare i parametri $\boldsymbol{\theta}$ in modo da rendere f il più simile possibile a f^*. In questo compito non ci interessa la generalizzazione statistica, vogliamo che il modello funzioni bene solo sul training set $\mathbb{X} = [0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T$. Trattiamo il problema come un problema di regressione e usiamo l'errore quadratico medio come funzione loss.

$$\boldsymbol{J}(\boldsymbol{\theta}) = \frac{1}{4} \sum_{x \in \mathbb{X}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta}))^2$$

Definiamo quindi il nostro modello come

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{x}^T \boldsymbol{w} + b$$

Da qui si minimizza $\boldsymbol{J}(\boldsymbol{\theta})$ per $\boldsymbol{w}$ e b. Risolvendo le equazioni di minimizzazione si ottiene $\boldsymbol{w} = \boldsymbol{0}$ e $b = \frac{1}{2}$. La funzione da' in output sempre 0.5. Il modello lineare non è quindi in grado di imparare la funzione XOR, come mostrato nella Figura 1.4.

Ora introduciamo una semplice rete neurale con un hidden layer con due unità per la risoluzione del problema. Sostanzialmente volgliamo usare un modello che impari un diverso spazio delle features in cui un modello lineare possa imparare la funzione. In Figura 1.5 è mostrato uno schema del modello.

La rete comprende un vettore di unità nascoste $\boldsymbol{h}$ che sono clacolate dalla funzione $f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$, il loro valore poi è dato come input al secondo layer, il cui output è l'output

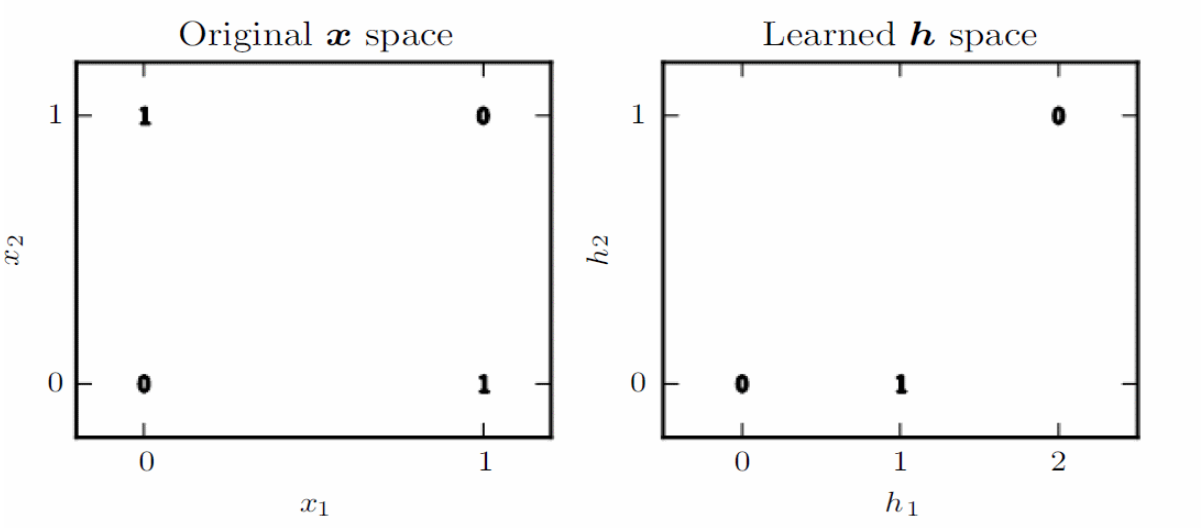

Figura 1.4: Apprendimento della funzione XOR. *Sinistra*) Un modello lineare applicato direttamente all'apprendimento della funzione XOR *Destra*) La rappresentazione dei dati imprarate dall'hidden layer **h**. Da [1].

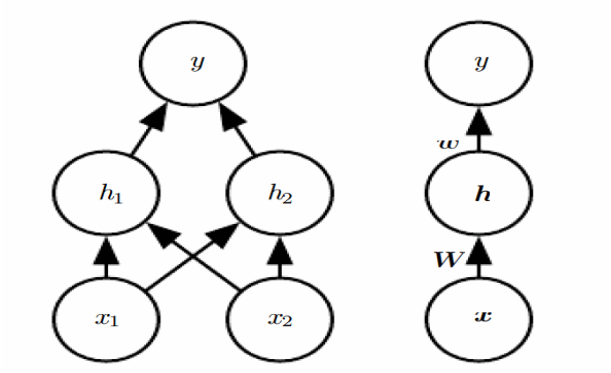

Figura 1.5: Schema della semplice rete neurale utilizzata per imparare la funzione XOR. Da [1].

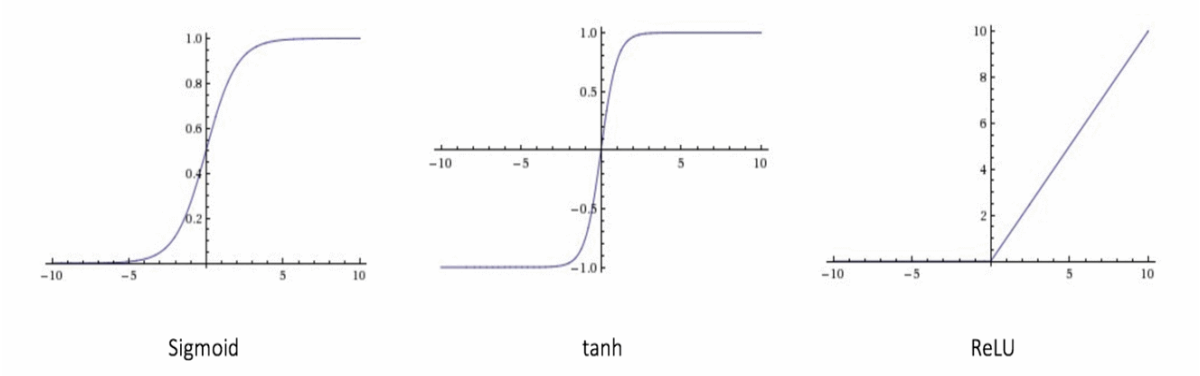

Figura 1.6: I grafici di alcune tra le funzioni di attivazione più utilizzate *Sinistra)* Grafico della funzione sigmoide *Centro)* Grafico della funzione tangente iperbolica *Destra)* Grafico della funzione unità lineare rettificata(ReLU). Da [5].

totale della rete neurale. Il secondo layer applica come prima una funzione lineare, ma stavolta sull'output del primo layer $\boldsymbol{h}$. Il modello completo ora è formato da due funzioni concatenate $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = f^{(2)}(f^{(1)}(\boldsymbol{x}))$. $f^{(1)}$ in questo caso non può essere lineare in quanto se lo fosse anche l'intero modello rimarrebbe lineare, e non potrebbe risolvere il problema. Se $f^{(1)}$ fosse lineare, ovvero avesse la forma $f^{(1)}(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{x}$ e anche $f^{(2)}$ avesse la forma $f^{(2)}(\boldsymbol{h}) = \boldsymbol{h}^T\boldsymbol{w}$ avremmo $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{W}^T\boldsymbol{x}$ e potremmo rappresentarla come $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w'}$ con $\boldsymbol{w'} = \boldsymbol{W}\boldsymbol{w}$. Per rendere quindi la rappresentazione non lineare utilizziamo una funzione fissata g che introduca non linearità al modello, ovvero definiamo $\boldsymbol{h} = g(\boldsymbol{W^T x} + \boldsymbol{c})$. Tale funzione è chiamata **funzione di attivazione**. Questo metodo è utilizzato nella maggior parte delle reti neurali. La funzione di attivazione utilizzata di default è la funzione ReLU definita come $f_{ReLU}(x) = max0, x$. Altre funzioni di attivazione possono essere la funzione sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$ e la tangente iperbolica $tanh = 2\sigma(2x) - 1$. I grafici di queste funzioni sono mostrati in figura 1.6.

Il nostro modello ora ha la forma:

$$f(\boldsymbol{x; W, c, w}, b) = \boldsymbol{w}^T max0, \boldsymbol{W^T x} + \boldsymbol{c} + b$$

Consideriamo ora una soluzione del problema:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

e b = 0.

Abbiamo la design matrix contenente i quattro punti del set

$$\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Il primo passo consiste nel moltiplicare la matrice input $\mathbf{X}$ per la matrice dei pesi del primo layer $\mathbf{W}$

$$\mathbf{XW} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

e aggiungendo il vettore bias **c**:

$$\mathbf{XW+c} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

In questa rappresentazione tutti i punti appartengono ad una retta con coefficiente angolare 1. Muovendoci sulla linea, l'output dovrebbe cominciare a 0, andare a 1 e tornare nuovamente a 0. Un modello lineare non può imparare una funzione del genere. Per questo applichiamo la funzione di attivazione ReLu ad ogni punto per calcolare **h**

$$\mathbf{h} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

completiamo moltiplicando h per il vettore dei pesi **w**:

$$\mathbf{hw} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Abbiamo trovato una soluzione per l'apprendimento della funzione XOR. Naturalmente in un problema reale i parametri e i punti in input possono essere di un numero assai più grande, in generale quindi non si può semplicemente indovinare una soluzione

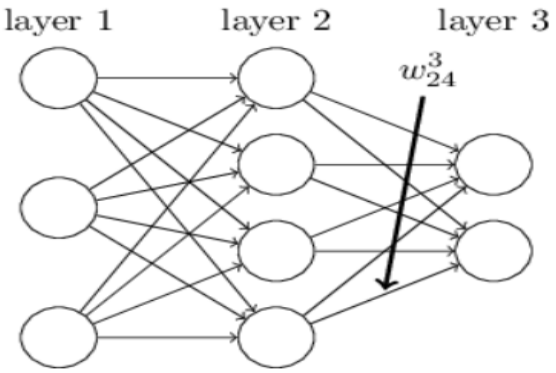

Figura 1.7: Schema di una semplice rete neurale fully connected. In figura è evidenziato uno dei parametri per dare un esempio della notazione usata. Da [5].

come in questo caso. Nei casi reali uno dei metodi più utilizzati sono gli algoritmi di apprendimento guidati dal gradiente, in un certo senso simili alla discesa del gradiente già discussa.

1.2.2 Backpropagation

Una rete neurale è definita feedforward se l'informazione in input $\mathbf{x}$ fluisce nella rete fino all'output, dove poi la funzione loss viene calcolata. Tale concetto è detto forward propagation. L'algoritmo backpropagation invece serve a calcolare il gradiente della cost function dall'informazione presa dal valore della cost function, infatti calcolare il gradiente analiticamente può essere concettualmente semplice, ma computazionalmente dispendioso. La backpropagation è un metodo di calcolo del gradiente poco dispendioso in termini computazionali. In generale l'algoritmo non è valido solo nel campo delle reti neurali, ma può essere usato per calcolare la derivata o gradiente di ogni funzione.
Prendiamo come esempio una rete neurale feedforward fully connected. Definiamo allora w^l_{jk} l'elemento della matrice dei pesi associato alla connessione tra il k-esimo neurone del (l-1)-esimo layer e il j-esimo neurone del l-esimo layer. In Figura 1.7 è mostrato uno schema della situazione.

Inoltre definiamo b_j^l il bias, $a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$ l'output, $z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$ l'input pesato e δ_j^l l'errore del j-esimo neurone nel l-esimo layer. Se allora la cost function è scrivibile come media su tutti gli esempi in input ovvero se èp possibile scrivere $C = \frac{1}{n}\sum_x C_x$ dove C_x è il valore della cost function calcolata per un singolo esempio e n è il numero di esempi e se è anche scrivibile come funzione degli output della rete allora possiamo applicare l'algoritmo backpropagation. L'algoritmo backpropagation è basato su quattro equazioni fondamentali[1]:

Equazione dell'errore nell'output layer:

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

Equazione dell'errore $\boldsymbol{\delta^l}$ in funzione dell'errore nel layer successivo $\boldsymbol{\delta^{l+1}}$:

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l)$$

Equazione della derivata della cost function rispetto ad ogni bias nella rete:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

Equazione della derivata della cost function rispetto ad ogni peso della rete:

[1]Nella notazione possiamo omettere il termine in basso per indicare un vettore piuttosto che un valore scalare. Le funzioni, come la funzione di attivazione σ sono applicate ad ogni elemento del vettore.

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Dove L indica l'ultimo layer della rete, $\nabla_a C$ è il vettore le cui componenti sono le drivate parziali rispetto ai valori in output dell'output layer $\frac{\partial C}{\partial a_j^L}$ e il prodotto $\odot$ è detto prodotto di Hadamard che opera la moltiplicazione tra gli elementi di un vettore:

$$(\boldsymbol{s} \odot \boldsymbol{t})_j = s_j t_j$$

Con queste equazioni è possibile calcolare il gradiente della cost function per poi aggiornare i pesi e bias.

1.3 Reti neurali convoluzionali

Una rete neurale convoluzionale (CNNs) è una rete che fa uso dell'operazione di convoluzione in almeno uno dei suoi strati invece della normale moltiplicazione di matrici. Le CNNs si sono dimostrate specialmente performanti nelle applicazioni pratiche, in particolare nell'image recognition, e in generale sono specializzate nel processare dati che hanno struttura "a griglia" (grid-like). Come si vede dalla Figura 1.8 le CNNs sono in grado di riconoscere oggetti, persone e animali nelle fotografie. Le CNNs si sono dimostrate il modello più efficace per l'analisi di immagini[10].

Una delle prime CNNs ad esser state implementate con efficacia è LeNet5 mostrata nel 1994 nel lavoro di Yann LeCun [7]. LeNet5 è una semplice CNN in grado di riconoscere e classificare lettere e numeri scritti a mano e sebbene esistano architetture più avanzate, tutte hanno bene o male la stessa stessa struttura. La struttura delle reti neurali convoluzionali si divide in:

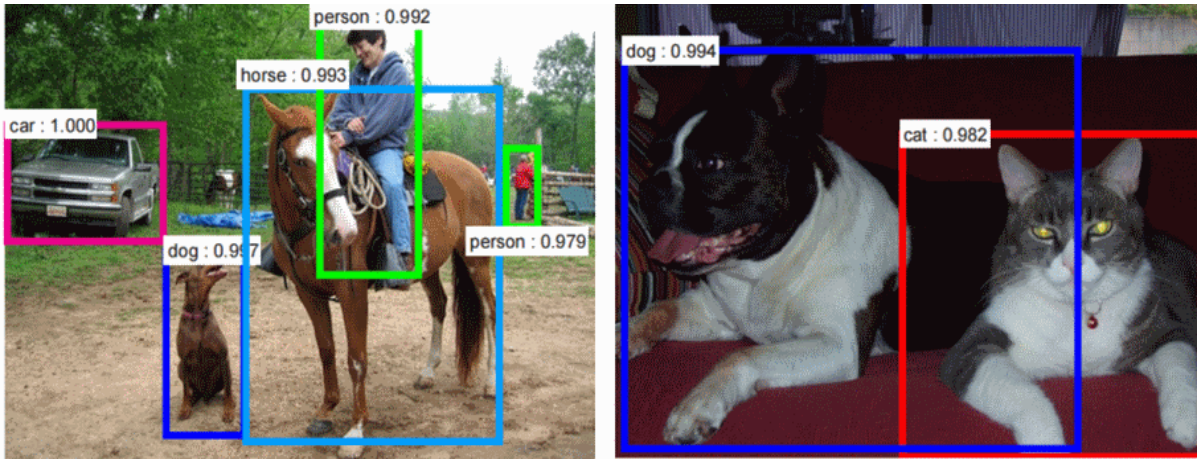

Figura 1.8: Due esempi di riconoscimento di immagini con una CNN. Il programma è in grado di riconoscere i vari oggetti nelle immagini. Immagine da [6]

1. Layer convoluzionali
2. Layer di pooling o subsampling
3. Layer fully connected

I vari tipi di layer possono ripetersi nella struttura della CNN: in genere blocchi formati da layers convoluzionali seguiti da layers di pooling si alternano prima di terminare in uno o più layer fully connected. Nella Figura 1.9 è mostrata la struttura di LeNet5 formata da due layer convoluzionali alternati a layers di pooling e tre layers fully connected.

1.3.1 La convoluzione

In generale l'operazione di convoluzione è un'operazione su due funzioni di una variabile reale definita come:

$$s(t) = \int x(a)w(t-a)da$$

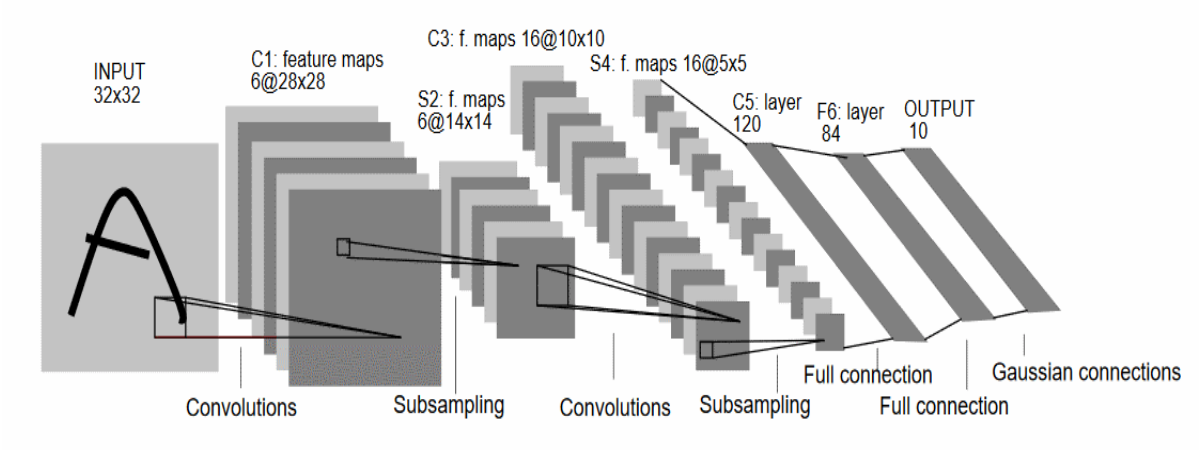

Figura 1.9: Struttura di LeNet5, qui usata per riconoscere una lettera scritta a mano.Ogni piano nell'immagine rappresenta una feature map. Si nota dall'immagine come i layer di pooling sono alternati ai layer di subsampling e come il le feature maps finali vengono analizzate dai layers fully connected. Immagine da [7]

Ed è indicata dal simbolo $*$:

$$s(t) = (x * w)(t)$$

In realtà l'operazione di convoluzione che ci interessa è la convoluzione discreta, dove l'integrale nella formula diventa una sommatoria:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a)$$

Dove x è detto *input*, w *kernel* ed s *output* o *feature map*. L'operazione di convoluzione che utilizzano le CNNs è discreta e la sommatoria è naturalmente finita, e nel caso dell'analisi delle immagini l'input diventa un array bidimensionale I e dunque anche il kernel sarà bidimensionale:

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

La convoluzione discreta può essere vista come una moltiplicazione di matrici, come si vede in Figura 1.10. In pratica, un'immagine è effettivamente una matrice WxH,

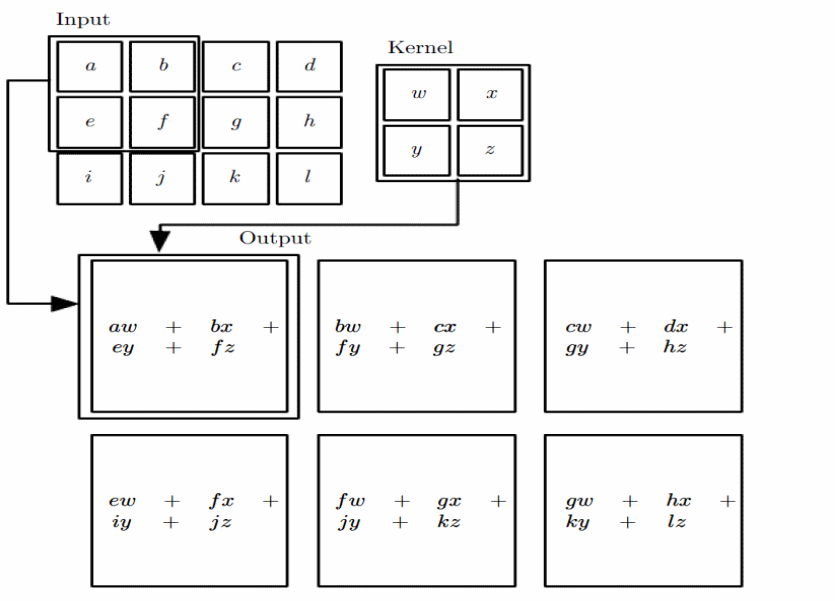

Figura 1.10: L'operazione di convoluzione utilizzata dalle CNNs è equivalente ad una moltiplicazione di matrici. Da [1]

dove W è la larghezza in pixel e H è l'altezza, di numeri che definiscono il colore che ha quell'immagine (per semplicità consideriamo immagini in bianco e nero, dove il valore in ogni pixel determina quanto "bianco" o "nero" sarà quel pixel). L'operazione di convoluzione quindi è quella di scannerizzare l'intera immagine con un filtro, che è una matrice di dimensione più piccola della matrice immagine. A seconda di quale filtro si utilizza per la convoluzione si ottengono diverse immagini in output, possiamo scegliere il filtro per individuare i bordi, effettuare un blur o altri effetti. Nella Figura 1.11 vengono mostrati degli esempi di convoluzione su un'immagine con i rispettivi filtri.

Ci sono inoltre altri parametri che influenzano la convoluzione:

1. **Profondità**

   Potendo scegliere diversi filtri nello stesso layer di convoluzione per estrarre diverse features dell'immagine, definiamo tale numero D di filtri come *profondità*.

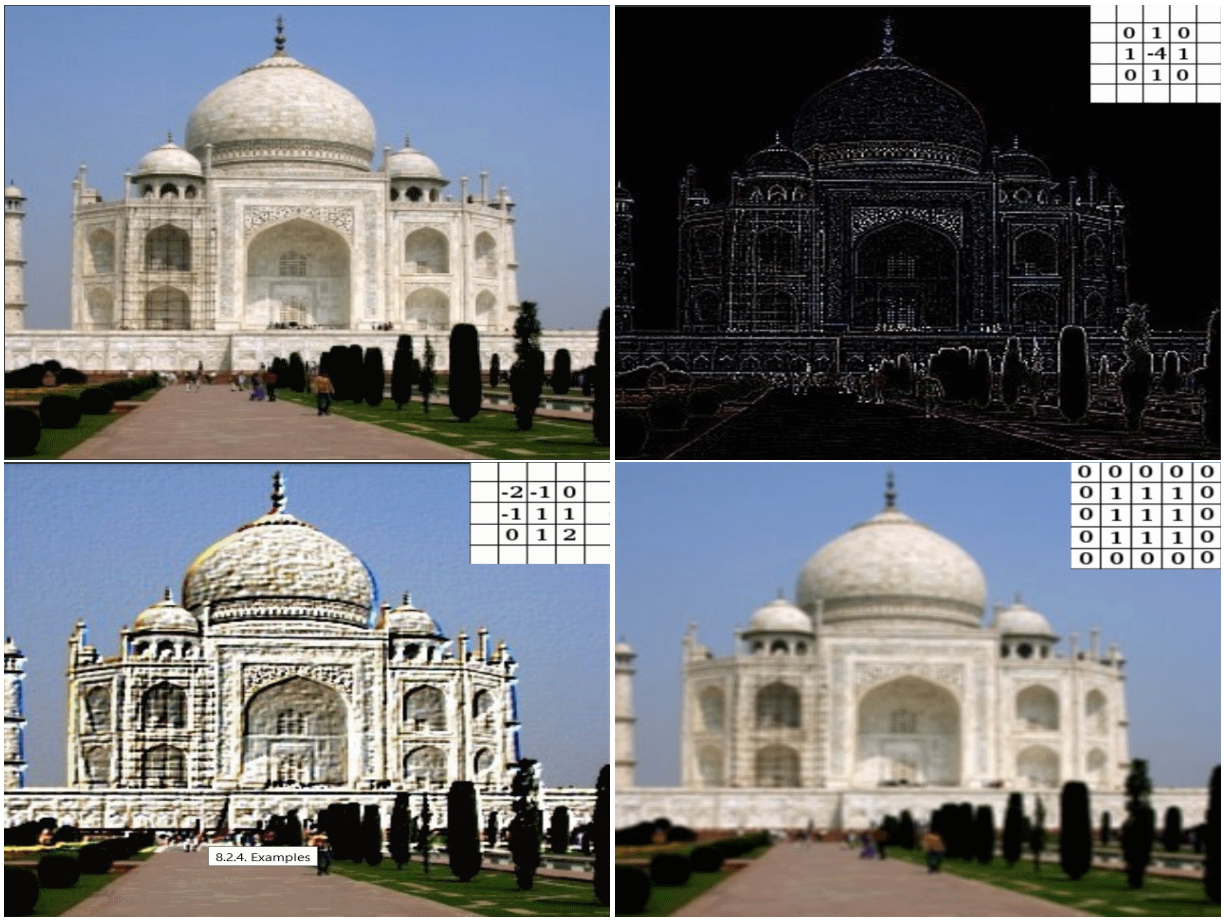

Figura 1.11: Diversi filtri applicati ad una immagine. *Alto a sinitra*) Immagine originale. *Alto a destra*) Rilevazione dei bordi. *In basso a sinistra*) Filtro emboss *In basso a destra*) Filtro blurr. Da [8].

2. **Passo**

   Possiamo scegliere di quanti pixel spostare il filtro di volta in volta, chiamiamo tale numero *passo*. Di solito si sceglie un passo di un pixel (in questo caso la dimensione dell'output è uguale a quella dell'input), due o, più raramente, tre.

3. **Zero padding** Se la dimensione dell'immagine non è un multiplo del passo, la dimensione dell'output della convoluzione sarà minore di quella dell'input. L'operazione di *zero padding* consiste nell'aggiungere dei pixel di valore zero ai brdi dell'immagine, così da evitare tale restringimento.

1.3.2 Funzione di Attivazione ReLU

Una volta completato il processo di convoluzione, vi è un layer che introduce *non linearità* al modello, si fa questo perchè il nostro modello vuole individuare delle features non lineari, introducendo questa funzione di attivazione dunque aumentiamo la complessità del modello. La funzione di attivazione più utilizzata nelle CNNs è la ReLU (Rectified Linear Unit):

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

La funzione semplicemente rimpiazza ogni pixel con valore negativo con il valore 0. Nella Figura 1.12 si vede l'effetto della funzione ReLU sulla feature map.

1.3.3 Il pooling

L'ultimo passaggio che effettua un layer convoluzionale è l'operazione detta di *Pooling*. In questo step modifichiamo l'output degli step precedenti atteraverso una funzione di pooling. Una funzione di pooling assegna un certo valore ad un punto della mappa in output in base al valore di un certo numero di pixel nell'intorno del punto. Per esempio, dividendo l'input della funzione di pooling in quadrati 3X3, la funzione *max pooling* restituisce il pixel col valore più alto, la funzione *average pooling* restituisce la media di tutti i valori nell'input. In Figura 1.13 è mostrato l'effetto del max pooling ccon filtro 2x2.

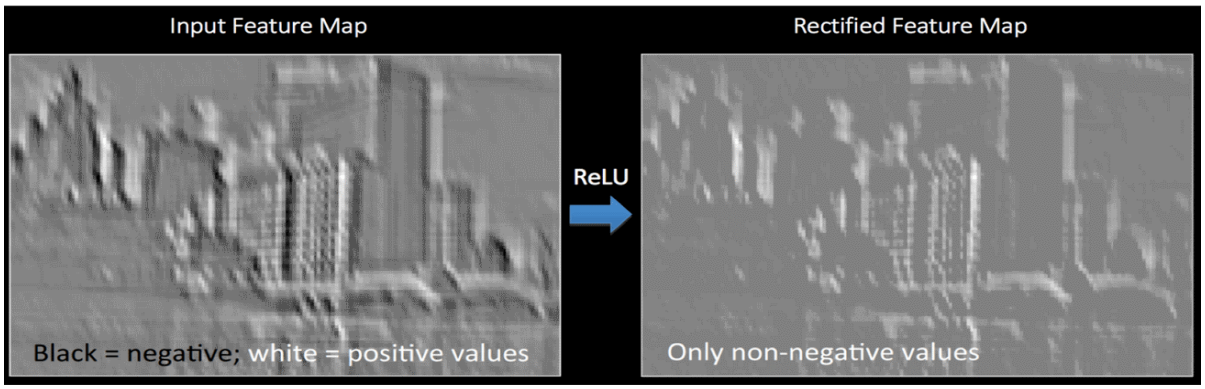

Figura 1.12: Effetto della funzione ReLu su una feature map. Nella feature map data in input i valori in nero sono valori negativi, quelli bianchi sono positivi e quelli grigi sono intermedi. La funzione ReLU rimuove i punti negativi. Da [6].

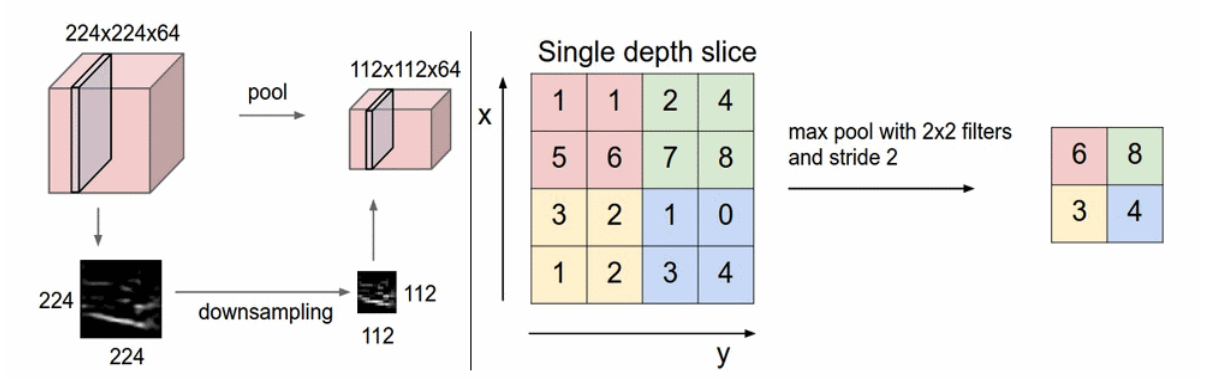

Figura 1.13: Max pooling con filtro 2x2 nelle reti neurali convoluzionali. A sinistra è mostrata la riduzione della feature map. A destra l'effetto del max pooling per un'immagine 2D con filtro 2x2. Da [9].

L'operazione di pooling dunque:

1. rende la rappresentazione dell'input più piccola così da essere più facilmente processabile
2. riduce i parametri e i calcoli da fare quindi controllando l'overfitting
3. rende la rete invariante rispetto a piccole trasformazioni, distorsioni e traslazioni
4. aiuta a rendere equivariante la rappresentazione di un'immagine, ovvero rende un oggetto riconoscibile indipendentemente dalla sua posizione nella figura

1.3.4 Vantaggi delle CNNs

L'utilizzo dei layer convoluzionali in una rete neurale apporta dei vantaggi introducendo tre novità:

Interazione sparsa Il fatto che il filtro sia più piccolo dell'input permette alla rete neurale apprendere features salienti senza che ogni unità di output interagisca con ogni unità di input. Se abbiamo una immagine di m pixel in input e una in output di n pixel, in una rete neurale tradizionale avremo $m * n$ connessioni e quindi $m * n$ parametri. Se invece permettiamo a ciascun neurone di un certo layer di interagire con solo k punti in input (naturalmente con k più piccolo di m anche di ordini di grandezza) avremo molti meno parametri, alleggerendo così il calcolo. La situazione è mostrata in Figura 1.14.

Condivisione dei parametri Oltre alla sparsità delle interazioni tra i nodi della rete neurale, un'altra conseguenza dell'utilizzo della convoluzione è il fatto di utilizzare un parametro per tutti i punti di input, invece di avere un peso diverso per ogni signola connessione. Ogni neurone ad in un dato layer cerca di individuare la stessa feature nell'input. In questo modo si accelera ulteriormente il tempo di calcolo e la memoria necessaria. La situazione è mostrata in Figura 1.15.

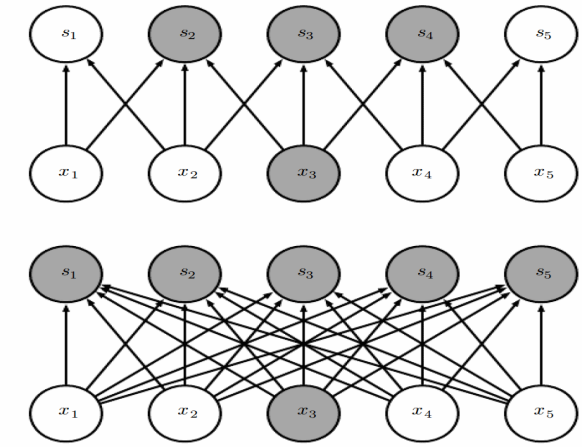

Figura 1.14: La connettività sparsa in una rete neurale convoluzionale. Nell'immagine in altro abbiamo la connettività sparsa permettendo ad ogni neurone di interagire con un nnumero limitato di punti in input. Nell'immagine sotto una connessione tipica di una rete neurale, detta fully connected. Da [1].

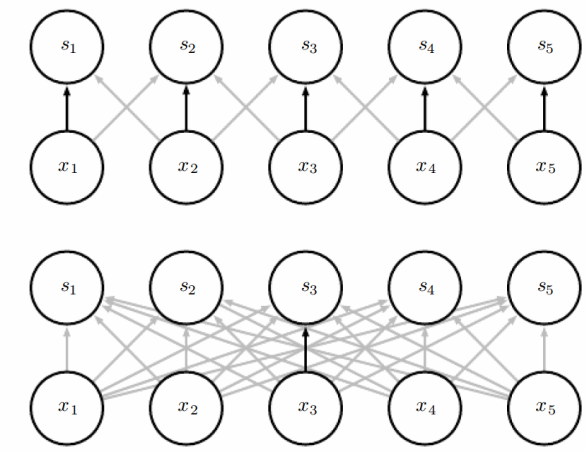

Figura 1.15: La condivisione dei parametri in una rete convoluzionale. Nell'immagine in alto un layer convoluzionale: la freccia nera indica l'elemento centrale di un kernel di tre elementi. Grazie alla condivisione dei parametri, questo parametro è utilizzato su tutti i punti di input. In basso la situazione in una rete neurale tradizionale: la freccia nera indica l'elemento centrale della matrice di pesi che è utilizzato una sola volta. Da [1].

Equivarianza della rappresentazione rispetto alla traslazione Come già introdotto, come equivarianza della rappresentazione si intende che dato un certo input, se esso cambia, l'output cambierà nello stesso modo. Una funzione $f(x)$ è equivariante rispetto alla funzione g se $f(g(x)) = g(f(x))$. Nel caso della convoluzione, se g è una funzione che trasla l'input, allora l'operazione di convoluzione è equivariante a g. L'equivarianza rispetto alla traslazione è una conseguenza della condivisione dei parametri. La convoluzione non è di per sè equivariante ad altre alterazione dell'input, come il cambiamento della scala o la rotazione: serviranno altri meccanismi per renderla equivariante rispetto anche a tali trasformazioni.

1.4 Autoencoders

L'autoencoder è una rete neurale addestrata a training non supervisionato a ricostruire l'input che gli è fornito. Esso è formato da un encoder ed un decoder, che rispettivamente applicano una funzione encode $\mathbf{h} = f(\mathbf{x})$ e una funzione di ricostruzione $\mathbf{r} = g(\mathbf{h})$ Se l'autoencoder è reso in grado di imparare esattamente $g(f(\mathbf{x}))$ esso non è particolarmente utile. Infatti l'utilità dell'autoencoder sta nel metterlo in condizione di non poter restituire esattamente l'output, in questo modo è costretto ad imparare le informazioni più utili del set di dati. Per questo l'autoencoder è solitamente da un unico hidden layer $\mathbf{h}$ che descrive un certo codice usato per rappresentare l'output. L'autoencoder è utilizzato di solito per l'estrazione di features o la riduzione della dimensionalità, recentemente sono utilizzati anche come architettura generativa.

1.4.1 Autoencoder sottocompleti, sovracompleti e regolarizzati

In generale, utilizzando un autoencoder, non si è interessati all'output ma piuttosto all'hidden layer che dovrebbe identificare alcune proprietà salienti dell'input. Un modo per evitare che l'autoencoder impari la funzione identità (il caso in cui sarebbe inutile) è semplicemente rendere l'hidden layer $\mathbf{h}$ più piccolo dell'input. Si parla in questo caso di autoencoder incompleto. Il processo si riduce semplicemente a minimizzare la loss

function L che penalizza $g(f(\mathbf{x}))$ se è troppo diversa da $\mathbf{x}$:

$$L(\mathbf{x}, g(f(\mathbf{x})))$$

Se la loss function L è l'errore quadratico medio e il decoder è lineare, l'autoencoder si effettua una PCA(Principal Component Analysis). Se invece le funzioni encode $\mathbf{h}$ e decode $\mathbf{r}$ sono non lineari, l'autoencoder impara una generalizzazione non lineare della PCA. Tuttavia se sia l'encoder e il decoder hanno capacità troppo grandi, verrà effettuata una copia dell'input senza l'estrazione di features importanti. Se invece si lascia che il codice abbia la dimensione uguale o maggiore dell'input (autoencoder sovracompleto) con funzioni encode e decode lineari il programma riproduce l'input senza imparare features utili. Tuttavia, in generale, si può scegliere di addestrare ogni architettura scegliendo la dimensione della capacità dell'encoder e del decoder in base alla complessità del problema che si vuole risolvere. Gli **autoencoder regolarizzati**, invece di limitare la dimensione del codice, utilizzano una determinata loss function che fa in modo che il modello ad avere altre proprietà oltre al copiare l'input. Le altre proprietà che l'autoencoder regolarizzato può avere sono per esempio la sparsità, la derivata della rappresentazione ridotta o la stabilità rispetto al rumore o ad input mancanti.

1.4.2 Autoencoder sparsi

Un autoencoder sparso è un autoencoder in cui è aggiunto un termine di penalità $\Omega(\mathbf{h})$ all'errore di ricostruzione:

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h})$$

Gli autoencoder sparsi sono utilizzati di solito per estrarre le features per un altro compito come la classificazione. A differenza del normale autoencoder, all'autoencoder sparso è richiesto di estrarre delle features specifiche dal dataset: la scelta del termine di penalità $\Omega(\mathbf{h})$ (può essere visto come il termine regolarizzatore) determina quali features si vuole estrarre con l'algoritmo.

1.4.3 Denoising autoencoder

Il denoising autoencoder è un altro tipo di autoencoder, dove invece di aggiungere un termine di penalità alla cost function L si introduce un certo rumore all'input

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$$

dove $\tilde{\boldsymbol{x}}$ è una copia del vettore input a cui è applicato un qualche tipo di perturbazione.

1.4.4 Addestramento avversario

In molti casi è utile sondare il vero livello di apprendimento di un algoritmo rispetto ad un compito oltre ad un test set. Un modo per farlo è di ricercare specificatamente degli esempi tali per cui si ottiene una classificazione erronea. L'addestramento che comprende una perturbazione volontaria degli esempi dal set di training è detto addestramento avversario. Tale tipo di addestramento si è rilevato utile nel contesto della regolarizzazione delle reti neurali. Come tecniche di **regolarizzazione** si intendono tutte quelle tecniche usate per ridurre solamente l'errore di testing ma non quello di training. Si è mostrato che anche delle reti neurali che ottengono una accuratezza vicina o pari al 100% sono attaccabili da esempi avversari. Un esempio di classificazione erronea da parte di una rete neurale convoluzionale per la classificazione di immagini è mostrata in Figura 1.16.

La sensibilità agli esempi avversari è dovuta alla eccessiva linearità del modello. Molte volte infatti succede che il modello sia costituito da blocchi lineari e sebbene sia molto facile da ottimizzare, l'output dell'algoritmo può cambiare molto rapidamente se vi sono molti valori in input. In fatti se cambiamo ogni punto del vettore input di un certo numero ϵ, allora una funzione lineare con pesi $\boldsymbol{w}$ di tale input con pesi $\boldsymbol{w}$ varia come

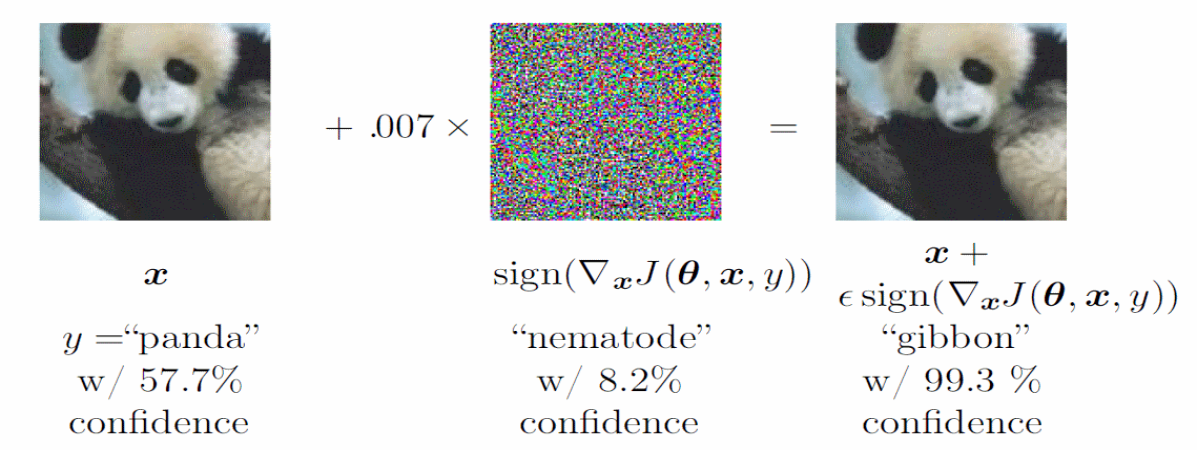

Figura 1.16: Esempio avversario che applicato a GoogLeNet su ImageNet. All'immagine originale è aggiunta una minuscola perturbazione impercettibile ad un osservatore umano, e l'algoritmo di classificazione riconosce come "gibbone"(una scimmia) con una confidenza del 99.3% quello che era un panda. Da [1]

$\epsilon \, ||\boldsymbol{w}||_1$ che può avere un valore molto grande se $\boldsymbol{w}$ è di grande dimensione. Il training avversario permette di evitare l'alta variabilità lineare locale. Inoltre il training avversario può essere utilizzato per ottenere un addestramento semisupervisionato.

1.4.5 Representation learning

In generale la scelta di un certo tipo di rappresentazione dei dati può influire sull'efficienza di un algoritmo di apprendimento. L'apprendimento supervisionato di una rete neurale feedforward può semplicemente essere un classificatore lineare (l'ultimo layer) a cui viene passata una rappresentazione dei dati generata da tutti i layer prima di esso. Nel caso delle reti feedforward con addestramento supervisionato gli hidden layer non hanno restirizioni o istruzioni specifiche sulle rappresentazioni da imparare. Si può tuttavia in altri casi scegliere specificamente che tipo di rappresentazione estrarre dai dati. Una rappresentazione estratta in un qualunque modo può essere utilizzata per un altro compito. Tale metodo è molto utile nella pratica quando ci si trova con dei grossi datasets in cui la maggior parte dei dati è senza target. Se si prova ad addestrare un programma

sui pochi dati etichettati il modello spesso overfitta sui dati. Possiamo in questi casi imparare una buona rappresentazione dei dati non etichettati e usarle per facilitare il training supervisionato. Il **Greedy layer-wise unsupervised pretraining** è un esempio canonico di apprendimento non supervisionato che impara una rappresentazione che aiuta poi nell'apprendimento supervisionato per un altro compito. Questo tipo di pretraining è eseguito di solito da architetture formate da un layer, come le RBM(restricted Boltzmann machine), un encoder o un qualsiasi modello che impari una rappresentazione latente dei dati. Data una struttura a più layer, ogni layer sarà preaddestrato con la rappresentazione del layer precedente. Questo approccio è stato utile nel trovare un buono stato di inizializzazione per le reti neurali. Successivamente al preaddestramento la rete viene calibrata sul compito da risolvere. Il **transfer learning** è un'altra tecnica di regolarizzazione in cui il modello è addestrato su due o più compiti diversi dove però l'apprendimento per uno di essi può essere significativo anche per gli altri. Il transfer learning è molto utilizzato nel riconoscimento delle immagini. Se per esempio addestriamo una rete neurale convoluzionale su un dataset per insegnarle a riconoscere delle immagini naturali (normali foto in cui è chiesto alla rete di riconoscere persone, animali, oggetti etc.) possiamo usare tale rete come rete preaddestrata su compiti in cui il volume di dati è scarso.

Capitolo 2

Applicazioni delle reti neurali nell'imaging medico del tumore alla prostata

2.1 Applicazioni delle reti neurali nell'imaging medico

Di seguito vengono elencati i maggiori utilizzi degli algoritmi di deep learning nell'imaging medico.

2.1.1 Classificazione

Classificazione di immagini o esami La classificazione delle immagini mediche è stato uno delle primi utilizzi degli algoritmi di deep learning nell'imaging medico. Di solito si ha un certo numero di immagini come input e un singola singola variabile diagnostica in output (presenza/assenza della malattia). I dataset nell'imaging medico sono di solito molto ristretti rispetto alla computer vision generale (centinaia/migliaia di campioni contro milioni). Si utilizza allora la tecnica del transfer learning. Il transfer learning consiste sostanzialmente nel pre-addestrare la rete neurale per aggirare il fab-

bisogno di dati per il training del programma. Vi sono due strategie di transfer learning maggiormente utilizzate:

1. Utilizzare una rete pre-addestrata come estrattrice di features
2. Ricalibrare (eseguire un "fine tuning") un programma pre-addestrato sui dati medici

Entrambe le strategie sono state utilizzate con successo in molti casi, sebbene la seconda si stia dimostrando migliore. È stato mostrato che la ricalibrazione di una versione preaddestrata dell'architettura Inception V3 di Google può raggiungere la performance di un esperto umano[]. Tale risultato non risulta essere stato raggiunto utilizzando la prima strategia. Le architetture maggiormente utilizzate nella classificazione sono state all'inizio principalmente SAEs (stacked autoencoders), e RBMs (restricted Boltzmann machine) per il preaddestramento non supervisionato. Recentemente, tuttavia, si nota una forte preferenza per le reti neurali convoluzionali. Le aree di applicazione di queste tecniche è molto vario, sono utilizzate dall'analisi di immagini MRI (risonanza magnetica) del cervello all'imaging della retina. In breve, nella classificazione degli esami clinici le CNNs sono il metodo standard utilizzato, sopratutto quelle preaddestrate su immagini naturali che raggiungono quasi l'accuratezza di un medico esperto. Infine degli esperti hanno mostrato che le CNNs possono essere adattate specificatamente per sfruttare al meglio le caratteristiche delle immagini mediche.

Classificazione di un oggetto o di una lesione La classificazione di una lesione consiste nella classificazione di una piccola parte dell'immagine medica, come un tumore in uno specifico organo. Per questo tipo di compito le informazioni locali della lesione sia le informazioni globali sulla posizione della lesione sono importanti. Di solito degli algoritimi di deep learning generici non sono adatti per questo tipo di compito. Un approccio utilizzato in molti casi è quello multistream. Alcuni studi hanno utilizzato delle reti convoluzionali multistream concatenate e anche una combinazione di reti convoluzionali e ricorrenti. L'utilizzo di componenti ricorrenti aiuta a rendere l'analisi indipendente dalla dimensione dell'immagine. Per l'analisi di immagini 3D le reti devono essere riadattate.

Oltre alle reti neurali convoluzionali anche altri tipi di architetture sono state utilizzate per lo scopo, tra cui RBM(Restricted Boltzmann Machine), SAEs(Stacked Autoencoders) e CSAE(Convolutional Stacked Autoencoders), la cui differenza di quest'ultimo da una rete neurale convoluzionale è il preaddestramento non supervisionato con degli autoencoders sparsi. Un altro approccio interessante è anche il multiple instance learning (MIL) combinato con metodi deep learning. Questa procedura ottiene ottimi risultati in confronto all'estrazione di features a mano. Nella classificazione di lesioni vi è meno uso del preaddestramento della rete.

2.1.2 Rilevazione

Rilevazione di organi e regioni La rilevazione anatomica di oggetti è uno step importante nel preprocesso delle analisi di immagini mediche e per la pianificazione della terapia o intervento. Nella rilevazione di oggetti in 2D le architetture più utilizzate sono le reti convoluzionali, dati i loro buoni risultati. Sembra che anche le RNNs possano raggiungere ottimi risultati in questo compito.

Rilevazione di lesioni La rilevazione di lesioni nelle immagini mediche consiste nell'individuare lesioni nell'immagine. Questo tipo di problema è cruciale per la diagnosi e una delle più problematiche. Anche qui la maggior parte degli studi effettuati per questo scopo hanno fatto uso di reti neurali convoluzionali per analisi per pixel o voxel. Ci sono alcune differenze tecniche nella rlevazione e la classificazione di lesioni con reti neurali e, a parte queste la rilevazione e la classificazione sono risolti in modo piuttosto simile.

2.1.3 Segmentazione

Segmentazione di organi o di sottostrutture La segmentazione degli organi è una parte importante nei sistemi di diagnosi automatici e permette analisi quantitative di volume e forma delle regioni di interesse. Il compito consiste nell'identificare i pixel o voxel che formano il contorno, volume o superfice dell'oggetto desiderato. La maggior parte degli articoli sull'uso di algoritmi di deep learning applicati all'imaging medico si occupano

di segmentazione. Per questo compito sono state implementate molte architetture basate sulla rete convoluzionale e la RNN. La più conosciuta delle reti convoluzionali implementate per la segmentazione è la rete neurale U-Net da [3]. Oltre alle reti convoluzionali, anche le reti RNNs sono state utilizzate recentemente per la segmentazione.

Segmentazione di lesioni Nella segmentazione delle lesioni è un approccio misto tra la rilevazione di oggetti e la segmentazione. In questo compito sono state utilizzate architetture U-Net e simili per avere un'analisi di tipo globale e locale dell'immagine.

2.1.4 Registrazione

La registrazione delle immagini mediche consiste nel calcolo di una trasformazione di coordinate da un'immagine medica ad un'altra. Due metodi sono più comunemente utilizzati:

1. utilizzare reti di deep learning per stimare una misura di similarità tra due immagini per una ottimizzazione iterativa

2. predire direttamente i parametri della trasformazione con modelli di regressione

Anche qui le architetture usate più spesso sono di tipo convoluzionale per ora. Per il compito della registrazione delle immagini sono state utilizzate diverse architetture tra cui SAE e CNN,tuttavia non si è ancora ben compreso quale metodo sia il migliore.

2.2 Applicazioni delle reti neurali nell'imaging del tumore alla prostata

2.2.1 L'imaging del tumore alla prostata

Il tumore alla prostata è il quarto più comune cancro nei due sessi, e il secondo più comune tra gli uomini. Nel 2012 sono stati stimati 1.1 milioni di casi, il cui 70% nelle regioni più sviluppate del pianeta. Ad approsimativamente il 16% degli uomini verrà

diagnosticato [vedi quote da current] e sebbene la percentuale di sopravvivenza a 5 anni dalla diagnosi sia cresciuta dal 66.0%(1975) al 99.6% (2005) questo tipo di tumore rimane un problema medico di prima importanza. Nonostante l'attiva ricerca nel determinare le cause del tumore alla prostata, si sono trovate alcuni incerti fattori di rischio tra cui fattori genetici, di razza o etnicità, dieta e obesità. Tale lista da sola non può essere utilizzata per una diagnosi certa, quindi l'analisi di immagini mediche è fondamentale per l'identificazione della patologia. Lo sviluppo del tumore alla prostata può avvenire in due modi: lenta e veloce. La crescita lenta si presenta nell'85% dei casi, e in questo caso il tumore resta confinato nella ghiandola, e il trattamento può essere sostituito da una sorveglianza attiva. Nel secondo caso il tumore si sviluppa velocemente e crea metastasi con gli organi confinanti, nella maggior parte dei casi le ossa, e dato che la metastasi alle ossa è una malattia incurabile, tale situazione incide in modo sostanziale sulla mortalità della malattia. Il tumore si sviluppa di solito in aree specifiche della prostata: circa il 70-80% si sviluppa nella zona periferica(PZ), 10-20% nella zona di transizione(TZ) e circa il 5% nella zona centrale. Lo screening del tumore alla prostata si compone comunemente di tre step:

1. Un controllo PSA (Prostatic-Specific Antigen) per distinguere tra un caso di tumore a basso o alto rischio

2. Per conferma viene eseguita una biopsia transrettale ad ultrasuoni (TRUS)

3. Alla fine si analizzano i risultati per valutare la prognosi e lo stadio del tumore

Sebbene lo screening PSA si sia rivelato utile nell'identificazione del tumore nelle fasi iniziali esso è relativamente inaffidabile. Inoltre anche la TRUS può dare risultati incerti data la casualità con cui vengono prelevati i campioni di tessuto da analizzare: è possibile infatti "mancare" del tutto il tessuto malato e ottenere un falso negativo. Per ora, inoltre, la media di sovradiagnosi di falsi positivi si stima essere del 35% mentre il non riconoscimento di casi clinicamente rilevanti del 30%. Per questo vi è una ricerca molto attiva nei sistemi di diagnosi automatica utilizzando immagini a risonanza magnetica (MRI).

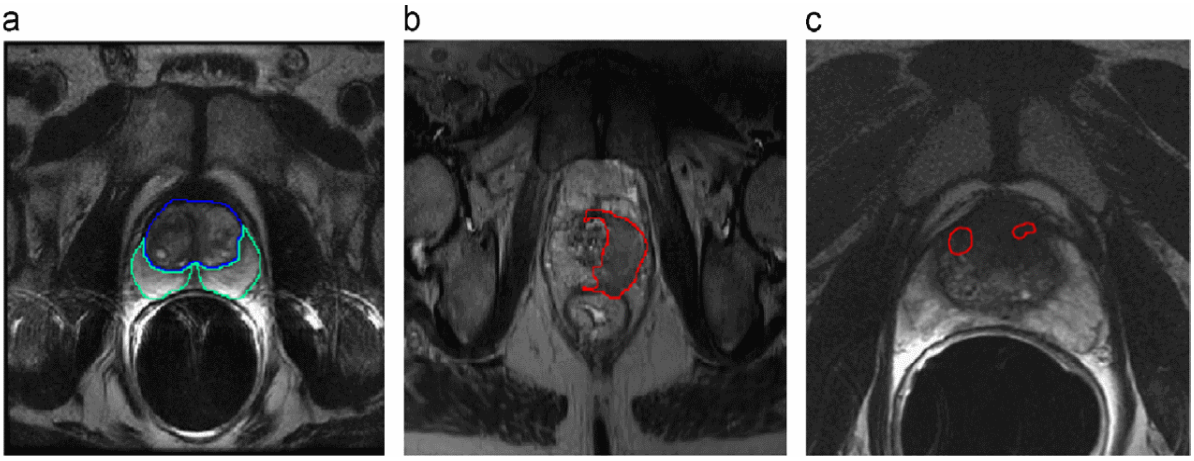

Figura 2.1: Immagini acquisite con sistema di imaging $T_2 - W$ MRI con gli scanner MRI a 1.5 e 3.0 Tesla . a) Sezione di un immagine $T_2 - W$ MRI acquisita con lo scanner MRI a 1.5 Tesla. L'area contornata in blu rappresenta la zona centrale (ZC) mentre quella in verde la zona periferica(PZ) b) Sezione di un immagine $T_2 - W$ MRI acquisita con lo scanner MRI a 3 Tesla in cui in rosso è evidenziato il tumore nella zona periferica(PZ). c)Sezione di un immagine $T_2 - W$ MRI acquisita con lo scanner MRI 3.0 Tesla in cui in rosso è evidenziato il tumore nella zona centrale(CZ).

Tecniche di imaging MRI

Rispetto alle tecniche elencate prima, un imaging a MRI della prostata è molto meno invasivo, e con un adeguato sistema di analisi, anche più accurato. Di seguito vengono elencati i metodi più utilizzati di imaging MRI.

$T_2 - W$ MRI L'acquisizione di immagini $T_2 - W$ MRI è stata la prima tecnica ad essere stata utilizzata per la diagnosi del tumore alla prostata[5]. Delle immagini acquisite con questa tecnica sono mostrate nell'immagine2.1, in cui quella a sinistra (a) raffigura una prostata sana, mentre le altre due, (b) e (c), raffigurano tumori rispettivamente nella zona centrale e nella zona periferica.

Nella tecnica $T_2 - W$ MRI è difficile riconoscere un eventuale tumore nella zona centrale(CZ) per il fatto che sia il tumore che la zona centrale stessa hanno bassa intensità

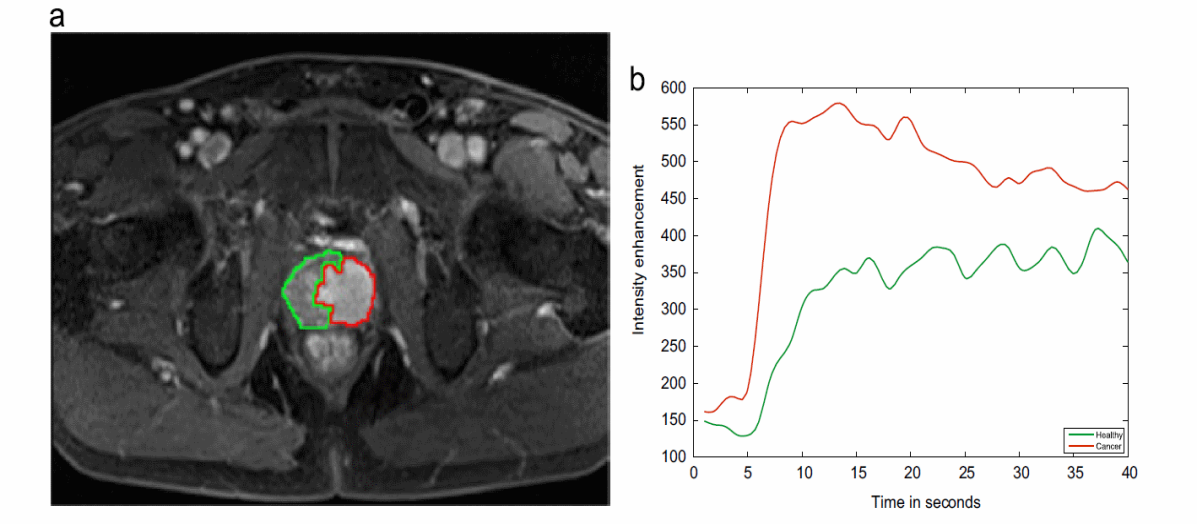

Figura 2.2: a) immagine $T_1 - W$ MRI ottenuta con lo scanner MRI a 3.0 Tesla b) Tipico andamento dei segnali di enhancement in un imaging DCE MRI per un tessuto sano (verde) e di un tumore (rosso)

di segnale. Nella zona periferica invece le zone con bassa intensità di segnale segnalano la presenza di un tumore. Si è anche mostrato che l'aggressività del tumore è inversamente proporzionale all'intensità del segnale della zona che occupa. Nonostante l'utilità della tecnica, essa manca come detto di sensibilità nella zona centrale.

DCE MRI La tecnica di imaging DCE MRI (Dynamic Contrast-Enhanced) sfrutta la vascularità caratteristica dei tessuti. Un mezzo di contrasto è iniettato nel paziente che estravasa poi dai vasi nello spazio extravasculare-extracellulare e rilasciato poi di nuovo nei vasi prima di essere espulso dai reni. La velocità di diffusione dell'agente può variare secondo diversi parametri: la permeabilità dei microvasi, la loro superficie ed il flusso di sangue. La tecnica si basa sull'acquisizione di un certo numero di immagini $T_1 - W$ MRI nel tempo. Da qui si analizza il comportamento farmacocinetico della sostanza nei tessuti della prostata. In Figura 2.2 è mostrato il comportamento al variare del tempo di una parte di tessuto sano e in cui è presente il tumore con questa tecnica.

Come si vede dal grafico, l'andamento della curva di enhancement per un tumore è

diversa da quella del tessuto sano. Esistono vari metodi di analisi quantitativi e semi-quantitativi di tali curve che aiutano nella localizzazione del tumore. Una combinazione dell'imaging DCEMRI con $T_2 - W$ MRI ha portato a risultati molto migliori rispetto al $T_2 - W$ MRI da solo. Tuttavia, data la natura "dinamica" della tecnica, un minimo movimento del paziente durante l'acquisizione dati può portare a errori di analisi. Inoltre in queste immagini è difficile distinguere un tumore maligno da una prostatite (infiammazzione della prostata) nella zona periferica o da una iperplasia prostatica benigna (un aumento del volume della prostata) nella zona centrale.

DWMRI La DWMRI (Diffusion-weighted MRI) sfrutta lo spostamento delle molecole d'acqua, ed è la tecnica più recente utilizzata nella diagnosi del tumore alla prostata. Il metodo è basato sul fatto che l'intensità del segnale è inversamente proporzionale al grado di moto casuale delle molecole d'acqua. Un maggiore grado di casualità del moto porta un maggiore perdita di segnale e un minore grado di casualità porta ad una minore perdita di segnale. Utilizzando questa tecnica combinata con immagini $T_2 - W$ MRI si hanno prestazioni migliori rispetto a immagini $T_2 - W$ MRI da sole. Tuttavia questa modalità ha una basssa risoluzione spaziale e specificità. Per ovviare a questi problemi si possono estrarre delle mappe quantitative dalle immagini DWMRI dette mappe ADC, che hanno colori invertiti. Anche in questo caso, in combinazione con immagini $T_2 - W$ MRI la performance è significativamente migliorata. In questo caso però ci sono alcuni tessuti nella ghiandola centrale (CG) che possono essere scambiati per un tumore per il loro basso livello di segnale ed è stato anche mostrato che vi è una grande variabilità delle mappe da paziente a paziente, rendendo più complicato riconoscere la lesione. In Figura 2.3 è mostrata un'immagine acquisita con la tecnica DWMRI e la sua mappa ADC.

La maggior parte degli studi condotti nella diagnosi automatica del tumore alla prostata hanno utilizzato le tecniche di imaging T_2W, MRS o DCEMRI poichè il contrasto tra i tessuti adiacenti alla prostata e quello della prostata è abbastanza marcato, mentre l'utilizzo di immagini DWMRI è più difficoltoso in quanto il contrasto è molto basso, sopratutto ad alti valori di b. La tecnica di imaging TRUS, come già detto, ha diversi

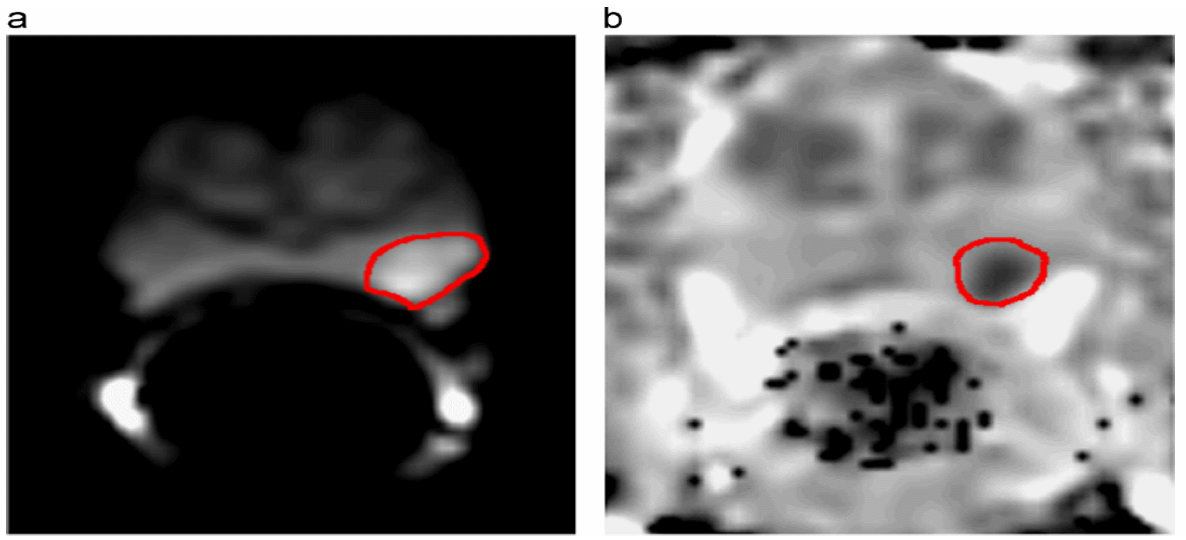

Figura 2.3: a) Immagine ottenuta con imaging DWMRI b) una ADC map. L'intensità di segnale corrispondente alla posizione del tumore è correlata inversamente in questi due tipi di tecniche di imaging. In rosso è cerchiato il tumore.

svantaggi ripetto all'imaging MRI multiparametrico. Alcuni di questi sono l'invasività del test e il fatto che in queste immagini è presente un certo livello di rumore che rende l'analisi più complicata. Vi sono comunque studi sui sistemi CAD che utilizzano tali immagini per la segmentazione della prostata.

2.2.2 I sistemi CAD

L'interpretazione delle immagini della prostata ottenute con la risonanza magnetica multiparametrica (MP-MRI) è spesso molto complicata, lunga e richiede una certa esperienza da parte del radiologo. Per questo si vogliono trovare sistemi di diagnosi automatici, i sistemi CADx (computer-aided diagnosis systems), che possano ridurre il tempo di analisi, che riducano l'esperienza richiesta dal radiologo e che eventualmente migliorino le prestazioni in termini di sensitività e specificità. Tipicamente ad un sistema CAD per la diagnosi del tumore alla prostata vengono fornite delle immagini MP-MRI, ed esso fornisce un risultato diagnostico, come una mappa di predizione che mostra le zone in

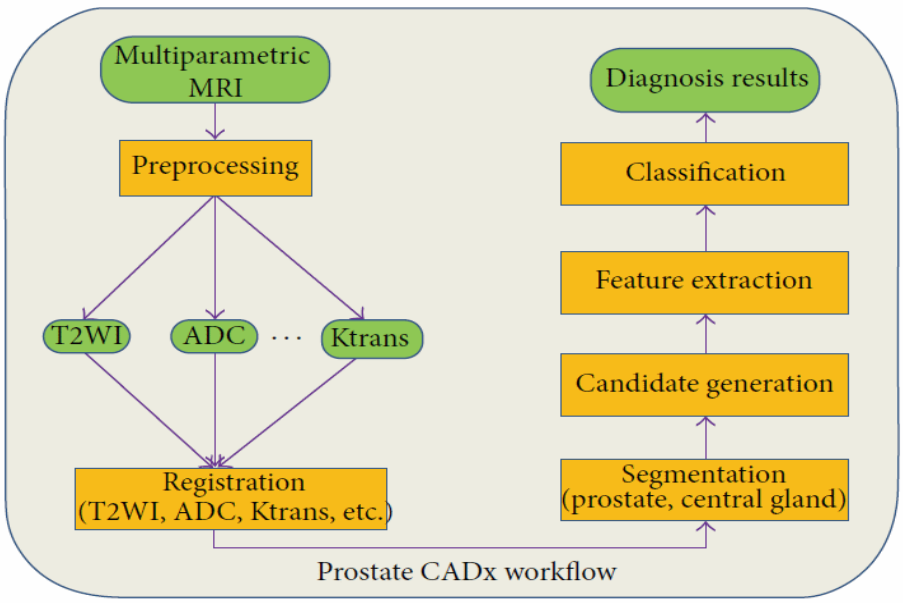

Figura 2.4: Schema procedurale di un tipico sistema CADx

cui è più probabile che vi si trovi il tumore. Ci sono alcuni step che i sistemi CADx hanno in comune, come la segmentazione o la classificazione. In Figura 2.4 è mostrato lo schema di un tipico sistema CAD.

Preprocesso dell'immagine Nel primo step, l'immagine viene normalizzata in modo tale da rendere il tumore individuabile con più facilità.

Segmentazione La segmentazione della prostata è necessaria per la classificazione successiva. Una segmentazione appropriata della prostata può anche aiutare nella radioterapia, biopsia e terapia focale oltre alle sue applicazioni nella diagnosi. La segmentazione della prostata è particolarmente complicata poichè i contorni della prostata sono facilmente confondibili con i tessuti adiacenti. Inoltre le immagini date in input possono essere state prese con protocolli di imaging diversi a seconda del centro medico od ospedale.

Registrazione Data la varietà di modalità di acquisizione immagini citata prima, è necessario comunemente una fase di registrazione delle immagini, in modo tale che tutte le immagini condividano lo stesso sistema di riferimento.

Classificazione Una volta completati gli step precedenti si può procedere alla rilevazione e classificazione dell'eventuale tumore. Come classificazione del tumore si può intendere sia il classificare i pixel(o voxel) dell'immagine medica come tumore presente o non presente (quindi effettivamente eseguendo una segmentazione della lesione) o la stima del grado di aggressività.

2.2.3 Segmentazione della prostata

La segmentazione automatica della prostata è un compito importante per la molte applicazioni cliniche come la radioterapia per la cura del tumore. Le maggiori difficoltà che si incontrano sono:

1. L'aspetto non omogeneo della prostata ai suoi bordi
2. La forma e dimensione della prostata significativamente variablie da paziente a paziente
3. La variabilità dell'intensità del segnale attorno al retto
4. Basso contrasto tra la ghiandola e le strutture adiacenti
5. I porotocolli di acquisizione delle immagini possono variare in base al centro medico o all'ospedale

Valutazione della performance degli algoritmi di segmentazione Nell'articolo [23] vengono elencati i metodi più utilizzati per valutare la precisione di un algoritmo di segmentazione della prostata(qui nel caso tridimensionale, ma valgono anche nel caso bidimensionale) :

1. **Il coefficiente Dice(DSC)**

   Il coefficiente Dice è calcolato come

   $$D(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|}$$

   dove $|X|$ è il numero di voxel nella segmentazione di riferimento e $|Y|$ è il numero di voxel della segmentazione dell'algoritmo.

2. **aRVD** aRVD è detta la differenza assoluta di volume. La differenza relativa di volume RVD è definita come

   $$RVD(X,Y) = 100 * (\frac{|X|}{|Y|} - 1)$$

   la differenza assoluta di volume è quindi

   $$aRVD = |RVD(X,Y)|$$

3. **La distanza di Hausdorff (HD o HDE)** Prima di poter calcolare la distanza di Hausdorff bisogna estrarre le superfici della segmentazione di riferimento e dell'algoritmo. Fatto ciò la distanza di Hausdorff è definita come

   $$HD(X_s, Y_s) = \max_{x \in X_s} \left(\min_{y \in Y_s} \mathbf{d}(x,y) \right)$$

   Dove X_s e X_s sono gli insiemi di punti superficie della segmentazione di riferimento e dell'algoritmo rispettivamente e $\mathbf{d}$ è la normale distanza euclidea.

In [18] è proposto un metodo di segmentazione della prostata in tempo reale durante la biopsia, migliorando un loro precedente lavoro sullo stesso problema [19]. Alla loro architettura originale di segmentazione della prostata basata su una struttura convoluzionale per l'acquisizione di features spaziali, è aggiunta una unità ricorrente per l'estrazione delle features temporali. La struttura base di segmentazione convoluzionale è basata sull'architettura U-net[20]. La U-net, illustrata in Figura 2.5, è un tipo di struttura di CNN particolarmente efficace nella segmentazione di immagini biomediche.

La struttura comprende un cammino contraente (parte sinistra) e uno espansivo (parte destra). Il cammino contraente è lo stesso di una CNN tradizionale che utilizza

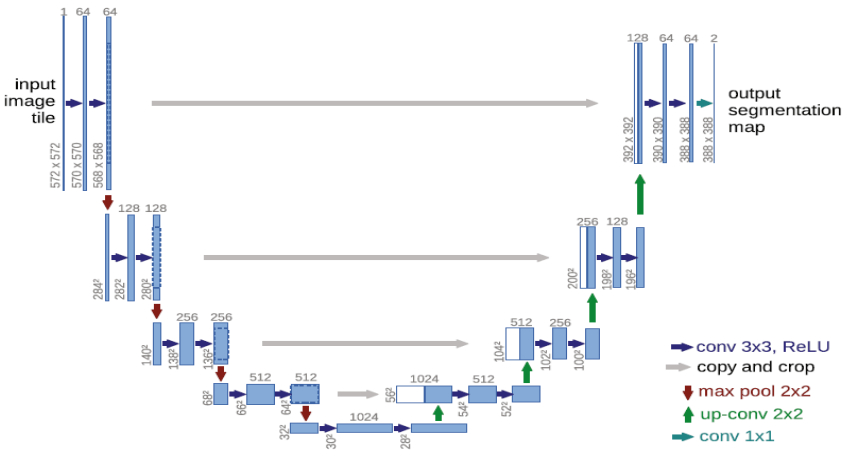

Figura 2.5: Schema della struttura della U-net. Da [3]

due convoluzioni 3x3, ognuna seguita da una funzione ReLU e un'operazione di max pooling 2x2 con passo 2. Il cammino inverso invece ricostruisce l'immagine segmentata con alta risoluzione. L'architettura proposta in questo articolo raggiunge un DSC di $(92.91 \pm 2.74)\%$, una mSDC (mean distance score) di $(1.12 \pm 0.79)mm$ e una distanza di Hausdorff HDE di $(2.97 \pm 1.96)mm$.

Nel recente lavoro [26] viene proposta una struttura di segmentazione della prostata utilizzando una Global Convolutional Adversarial Network (GCA-Net). La struttura comprende una struttura 3D fully convolutional Encoder-decoder, in cui l'encoder è una 3D ResNet che estrae le features dall'input e il decoder fully convolutional sfrutta la dettagliata estrazione delle features dell'encoder per dare una predizione di segmentazione dell'input. La ResNet è una struttura proposta da [27] che sostanzialmente sfrutta l'apprendimento del residuo della funzione da approssimare,un metodo che risolve il problema della *degradazione* dell'accuratezza del modello all'aumentare della profondità. In questo caso la ResNet è adattata per l'analisi di immagini 3D dato che il modello originale è stato creato per immagini 2D. Nel training avversario è richiesto alla struttura di individuare le inconsistenze tra il ground truth e il risultato della segmentazione, e la struttura è guidata a minimizzare tale errore. Le loss function rispettivamente per la rete generativa e quella discriminativa sono:

$$Loss_g(X,Y) = \lambda Loss_{cross_entropy}(G(X),Y) + Loss_{GAN}(D(G(X)+X), True)$$

$$Loss_D(G(X),Y) = Loss_{GAN}(D(G(X)+X), False) + Loss_{GAN}(D(X+Y), True)$$

Dove G, la rete di segmentazione, è addestrata a dare un risultato della segmentazione G(X) simile alla ground truth Y dall'immagine MRI X. Allo stesso tempo X è concatenato con Y e G(X) separatamente e passsato a D per la discriminazione. L'approccio ottiene degli ottimi risultati nella segmentazione della prostata con un DSC di 0.88 e un ABD di 2.152 mm.

Oltre alla segmentazione dell'intera prostata, è anche importante segmentare a sua volta le parti di essa poichè, come abbiamo visto, le teniche di imaging possono dare livelli di segnale diversi a seconda di che zona si analizza. Nell'articolo [21] viene implementato un metodo di segmentazione delle parti della prostata, in particolare la zona di transizione(TZ) e la zona periferica(PZ) utilizzando una rete neurale convoluzionale. Anche qui l'architettura usata è una U-net che analizza immagini MPMRI tridimensionali. L'architettura riesce a segmentare le zone della prostata con un dice score di 0.85 per la zona di transizione(TZ) e di 0.60 per la zona periferca(PZ).
Un altro studio, [25], si cerca di segmentare le stesse parti della prostata utilizzando sempre la struttura dell'articolo precedente aggiungendo un encoder per aggiungere un termine di loss per pixel usando come dataset immagini $T_2 - weighted$MRI della prostata. I risultati comparati delle due architetture (3D-Unet con e senza autoencoder) sono mostrate in Figura 2.6.

L'architettura ottiene un Dice score di 0.85 sulla zona di transizione(TZ) e 0.67 su quella periferca. L'utilizzo dell'encoder ha leggermente migliorato la precisione di segmentazione della zona periferica, mentre il risultato sulla zona di transizione rimane invariato.

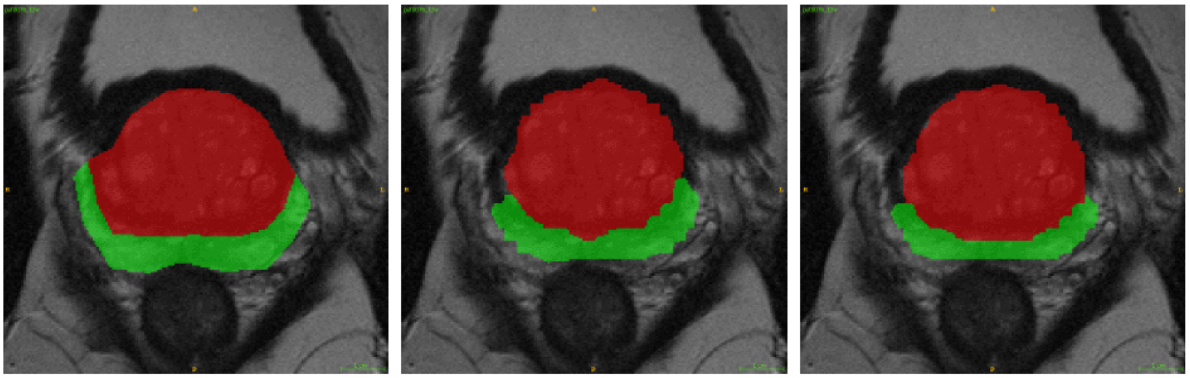

Figura 2.6: Segmentazione della prostata *sinistra*) Eseguita manualmente(ground truth con cui calcolare il DIcer score) *centro*) Ottenuta dalla segmentazione con 3DU-Net senza autoencoder *destra*) Ottenuta dalla segmentazione con 3DU-net con autencoder. Da [25]

2.2.4 Registrazione delle immagini

Un metodo di deep learning per la registrazione delle immagini mediche MRI e TRUS della prostata è stato proposto da *Haskins et al.*[28]. La registrazione delle immagini MR-TRUS permette l'analisi combinata dei de tipi di immagini, permettendo di integrare le due modalità nello stesso dataset. Il compito è difficile data la fondamentale differenza dei due metodi di imaging. Il lavoro [28] propone:

1. **Una metrica di similarità**
   La similarità tra le immagini TRUS e MRI è determinata utilizzando una rete neurale convoluzionale. La rete prende in input una immagine MRI e una TRUS e restituisce una stima dell'errore di registrazione. La rete è composta da 9 layer convoluzionali volumetrici con passo 1 in ogni direzione, e la funzione di attivazione ReLU è applicata tra ogni layer. Un layer di Batch Normalization (BN) è aggiunto dopo il secondo layer convoluzionale per rendere la rete più stabile rispetto alla differenza delle scale di intensità. Una skip connection è anche aggiunta per concatenare le feature maps ottenute a basso e alto livello. Il layer fully connected restituisce infine uno scalare che determina l'errore di registrazione tra le due immagini.

2. **Un metodo di ottimizzazione adatto allo scopo**
   In questo tipo di compito è necessario utilizzare una strategia di ottimizzazione efficace. Molti metodi di ottimizzazione comunemente usati possono dare cattivi risultati se l'inizializzazione dell'immagine mobile non è abbastanza vicina all'immagine fissa. Un metodo di ottimizzazione composta a due stadi: per primo un metodo differenziale con una certa inizializzazione e poi il risultato viene usato come soluzione iniziale stimata per la strategia di ottimizzazione Broyden-Fletcher-Goldfarb-Shanno (BFGS) di secondo ordine.

2.2.5 Diagnosi e classificazione istologica del tumore alla prostata

Metodo di valutazione dei sistemi di classificazione La valutazione della sensitività di un modello di diagnosi del tumore avviene solitamente calcolando l'area sotto la curva ROC(AUC), ottenuta mettendo a grafico la frazione di veri positivi in funzione dei falsi positivi. Tale area ha un valore che va da 0 a 1 ed indica la specificità ottenuta dal sistema di diagnosi.

Minh Hung Le et al. [10] investigano tre fattori nell'utilizzo delle reti neurali convoluzionali multimodali per la diagnosi del tumore alla prostata: per primo cercano un metodo di agumentazione dei dati, notando che le tecniche comunemente utilizzate nelle applicazioni di classificazione di immagini non mediche, come traslazioni, rotazioni casuali e riflessioni, non sono adatte per l'imaging medico. Vengono proposte invece delle tecniche di deformazione non rigide che potrebbero simulare casi di tumore e aumentare la varietà dei datasets assieme a trasformazioni rigide. Di seguito sono analizzati due metodi di fusione di informazioni da immagini ADC e $T2WMRI$ nelle CNNs, e viene proposto un nuovo metodo di fusione per la classificazione delle immagini, e viene comparato il risultato con altri approcci simili basati sulle reti neurali convoluzionali con estrazione delle features a mano, ottenendo un miglioramento dell'accuratezza del 2.5% e del 18.75%. In figura 2.7 sono mostrate le curve ROC dei tre metodi.

Indolent versus CS (%)	Handcrafted approach	ResNet Fusion-3	Combine-1	Combine-2	Combine-3
Accuracy	70.00	86.25	81.25	78.75	88.75
Sensitivity	95.12	100	95.12	90.24	100
Specificity	43.59	71.79	66.67	66.67	76.92

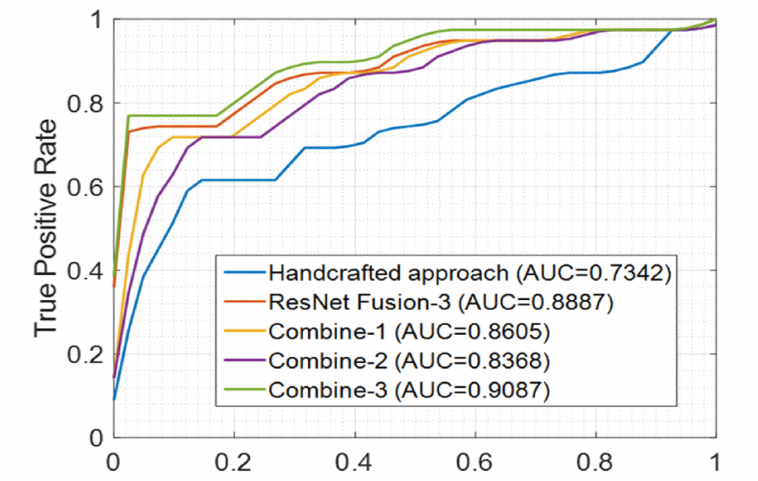

Figura 2.7: Curva ROC ottenuta dal modello SNCAE comparato ad altri metodi. Da [10]

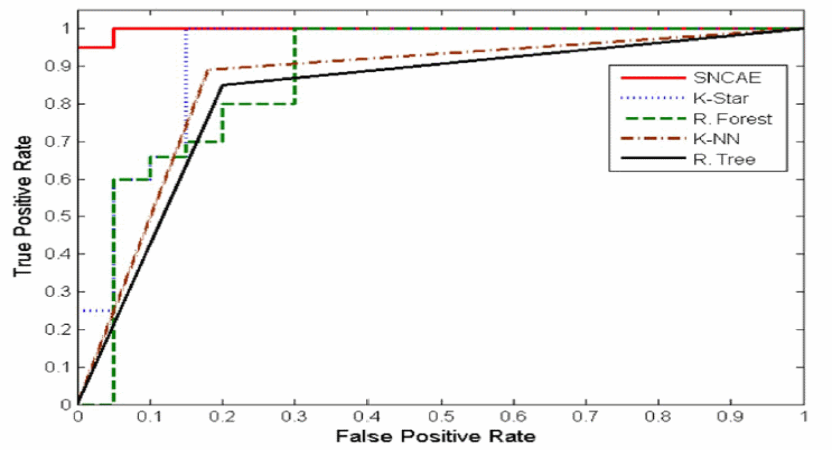

Figura 2.8: Curva ROC ottenuta dal modello SNCAE comparato ad altri metodi. Si nota l'efficacia sostanziale del modello. Da [32]

Nell'articolo [32] viene proposto un sistema CAD completo per la localizzazione e la segmentazione della prostata con un fattorizzazione non negativa di matrice NMF, costruire le CDF dalle ADC stimate dalle immagini DFMRI e la classificazione con un autoencoder per distinguere le immagini in cui è presente la lesione da quelle in cui non lo è. L'architettura di classificazione scelta è un Stacked Non-Negatively Constrained Auto-Encoders(SNCAE). Sostanzialmente si usano gli autoencoders per imparare una rappresentazione compressa dell'immagine per poi passare l'output ad un algoritmo che generalizza la regressione logistica con più categorie.I layers degli autoencoders sono preaddestrati col metodo greedy. Il valore di b nelle immagini DFMRI di b delle immagini DWMRI ottimale è di $700\frac{s}{mm^2}$. Il modello ottiene accuratezza, sensitività e specificità del 100%. In Figura 2.8 la curva ROC ottenuta dal modello comparata ad altri modelli.

Per assicurare che non vi sia overfitting nel modello, esso è stato valutato con una cross-validation, utilizzando 3 set per il training e uno per il testing, il tutto per quattro volte, ottenendo così un'accuratezza media del 95%. Gli autori sostengono che con datasets più grandi tale accuratezza tenderebbe a salire.

Il metodo radiomico per la rilevazione del tumore alla prostata Il metodo radiomico, a differenza dei sistemi CAD, è una tecnica di data mining, in cui si fa uso di una grande mole di dati per l'estrazione di features per l'analisi fenotipica del tumore, fornendo un metodo di diagnosi e prognosi. I processi radiomic-driven sono stati applicati con successo per la diagnosi e prognosi di molti tipi di lesione da immagini mediche MRI e TRUS e nella medicina di precisione.

Chung er al. [30] propongono un metodo radiomico per la diagnosi del tumore alla prostata utilizzando immagini MPMRI. La struttura utilizza immagini MPMRI standardizzate di pazienti con annotazioni e verifica della presenza della patologia da radiologo per generare un sequenziatore per estrarre le sequenze radiomiche specifiche dei casi di tumore alla prostata. Per un nuovo caso il sequenziatore radiomico è usato per generare una nuova sequenza radiomica dalle immagini MRI per l'analisi e classificazione fenotipica del tumore. Le sequenze radiomiche scoperte sono poi utilizzate da un classificatore per fornire supporto al processo di rilevazione e diagnosi. Una schematizzazione del processo è mostrato in figura2.9.

Nello studio è presentato un sequenziatore radiomico profondo, stocastico e convoluzionale, formato da 17 layers convoluzionali e due layers fully connected da 1000 e 500 nodi rispettivamente. Uno schema dell'architettura è mostrato in figura 2.10.

Uno dei problemi che sorgono nell'utilizzo di un modello di tale profondità è l'immensa quantità di parametri da far imparare al modello, richiedendo dunque una grande quantità di dati. Chung er al. approcciano il problema in due modi: primo, non considerano i parametri tra i nodi come da imparare singolarmente ma vengono determinati stocasticamente da una distribuzione. Con questo metodo i parametri da imparare vengono ridotti di molto. Secondo, si utilizza un metodo di data augmentation che consiste nella rotazione di ogni immagine di intervalli di 45° per aumentare il volume del dataset disponibile. Si ottiene con tale metodo una sensitività e una accuratezza maggiore rispetto ad altri modelli basati su estrazione di features a mano, rispettivamente del 64% (maggiore del 29%) e del 73% (maggiore del 16%), mentre la specificità dell'82% è peggiore del 10%.

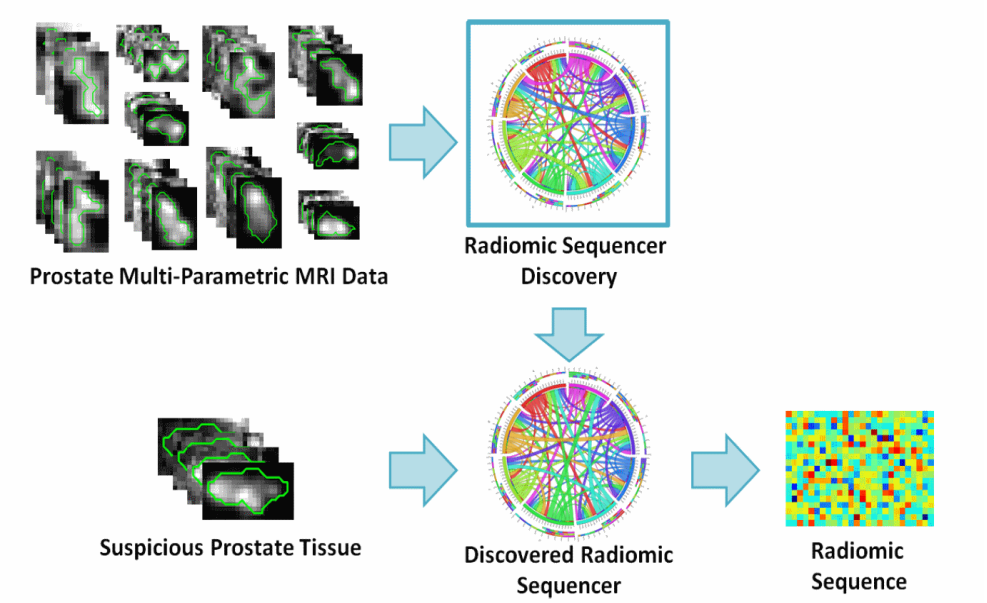

Figura 2.9: Schema del metodo radiomico per la diagnosi del tumore alla prostata. Da [30]

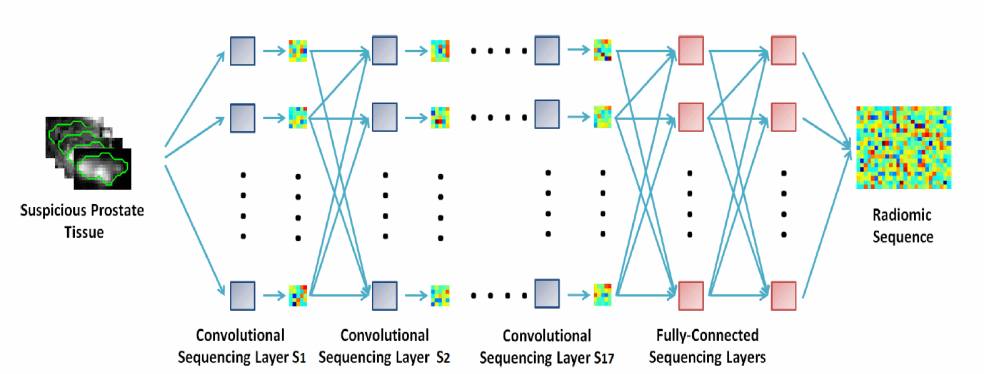

Figura 2.10: Architettura convoluzionale utilizzata per generare la sequenza radiomica di un caso clinico di tumore alla prostata. Da [30]

Classificazione del Gleason score Il Gleason score è un metodo per classificare i casi di tumore alla prostata per l'aggressività e il rischio per il paziente. Si tratta di un dato istologico, che riguardà cioè il tessuto tumorale. Il punteggio va da 2 a 10 e maggiore è lo score, più il tumore è aggressivo e rischioso. *J. Ren et al.*[29] propongono un modello di classificazione automatica del Gleason score del tumore alla prostata. Lo studio fa uso di un modello sia a training supervisionato che non supervisionato e avversario. Per prima cosa viene addestrato l'algoritmo sui dati etichettati disponibili, dove il Gleason score esatto è indicato. Dopodichè si addestra sul training set non etichettato eseguendo una ricalibrazione del classificatore preaddestrato. Il modello ottiene un'accuratezza dell' 83% del 10% superiore rispetto a studi precedenti.

2.3 Conclusioni

In questa tesi si sono introdotte le principali architetture di deep learning e il loro utilizzo nell'analisi di immagini mediche, dopodichè si sono analizzati i problemi presentati dai sistemi di diagnosi automatica del tumore alla prostata e le possibili soluzioni di essi che fanno uso di architetture di deep learning. Per prima cosa sono stati illustrati i concetti teorici principali dei metodi di machine learning, per poi analizzare nel dettaglio il funzionamento delle architetture di deep learning più utilizzate quali la rete neurale profonda, la rete neurale convoluzionale e vari tipi di autoencoder. Si sono poi mostrate le tecniche di imaging della prostata di tipo TRUS e MRI volte alla diagnosi del tumore in tale zona e come ogni signolo step dei sistemi di diagnosi automatica dall'analisi di immagini mediche TRUS e MRI multiparametriche rappresenti un problema unico e si sono presentati dei metodi efficaci di deep learning recentemente testati per la risoluzione di tali problemi. In particolare ci si è poi concentrati sulle tecnologie facenti uso di tecniche di deep learning impiegate nell'analisi delle immagini TRUS e MRI multiparametriche per la diagnosi del tumore alla prostata. I recenti studi analizzati in questo lavoro hanno raggiunto risultati notevoli impiegando diversi tipi di algoritmi di deep learning dimostrando l'efficacia di queste tecniche nei sistemi automatici di diagnosi del tumore alla prostata e ci si aspettano ulteriori migloramenti di performance di essi e un futuro impiego clinico diffuso ed efficace.

Bibliografia

[1] Goodfellow I., Bengio Y., Courville A., Deep Learning, MIT Press, 2016.

[2] J.M. Jerez et al., Missing data imputation using statistical and machine learning methods in a real breast cancer problem, Artificial Intelligence in Medicine 50 (2010) 105–115

[3] García Laencina, Pedro & Sancho-Gómez, José Luis & R Figueiras-Vidal, A. (2008). Machine Learning Techniques for Solving Classification Problems with Missing Input Data.

[4] Michael A. Nielsen, "Neural Network and Deep Learning", Determination Press, 2015

[5] Ujjwalkarn, A quick introduction to neural networks, The data science blog, 2016 `https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/`

[6] Ujjwalkarn, An Intuitive Explanation of Convolutional Neural Networks, The data science blog, 2016 `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`

[7] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE 1998;86:2278-2324

[8] `https://docs.gimp.org/en/plug-in-convmatrix.html`

[9] Andrej Karpathy, Appunti del corso CS231n: Convolutional Neural Networks for Visual Recognition , Stanford https://cs231n.github.io/convolutional-networks/#overview

[10] Minh Hung Le et al., Automated diagnosis of prostate cancer in multiparametric MRI based on multimodal convolutional neural networks, Phys. Med. Biol. 62 6497, 2017

[11] Lee J. et al., Deep Learning in medical imaging: General Overview, Korean J Radiol., 2017.

[12] S. Liao, Y. Gao, A. Oto, D. Shen, Representation Learning: A Unified Deep Learning Framework for Automatic Prostate MR Segmentation, Med Image Comput Comput Assist Interv. 2013 ; 16(0 2): 254–261.

[13] Shaoqing Ren, et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", 2015 https://arxiv.org/pdf/1506.01497v3.pdf

[14] Tom M. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997

[15] Jensen C., Korsager A.S., Boesen L., Østergaard L.R., Carl J. (2017) Computer Aided Detection of Prostate Cancer on Biparametric MRI Using a Quadratic Discriminant Model. In: Sharma P., Bianchi F. (eds) Image Analysis. SCIA 2017. Lecture Notes in Computer Science, vol 10269. Springer, Cham

[16] World Health Organization; stima dell' incidenza e della mortalità mondiale del tumore alla prostata nel 2012

[17] A Survey on Deep Learning in Medical Image Analysis, Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, Clara I. Sanchez, ArXiv, 2017

[18] Emran Mohammad Abu Anas, Pervin Mousavi, Purang Abolmaesumi, A deep learning approach for real time prostate segmentation in freehand ultrasound guided biopsy, Elsevier, Medical Image Analysis 48 (2018) 107–116

[19] Anas, E.M.A. , Nouranian, S. , Mahdavi, S.S. , Spadinger, I. , Morris, W.J. , Salcudean, S.E. , Mousavi, P. , Abolmaesumi, P. , 2017. Clinical target-volume delineation in prostate brachytherapy using residual neural networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 365–373 .

[20] Ronneberger, O. , Fischer, P. , Brox, T. , 2015. U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. Springer, pp. 234–241 .

[21] G. Mooij, I. Bagulho, H. Huisman, Automatic segmentation of prostate zones, arXiv, 2018

[22] G. Lemaitre, R. Marti, J. Freixenet, J.C. Vilanova, P.M. Wlaker, F. Meriaudeau, Computer-Aided Detection and Diagnosis for prostate cancer based on mono and multiparametric MRI: A Review, Elsevier, Computers inBiologyandMedicine60(2015)8–31

[23] Geert Litjens,Robert Toth,Wendy van de Ven,Caroline Hoeks,Sjoerd Kerkstra,Bram van Ginneken,Graham Vincent,Gwenael Guillard,Neil Birbeck,Jindang Zhang,Robin Strand,Filip Malmberg,Yangming Ou,Christos Davatzikos,Matthias Kirschner,Florian Jung et al. , Evaluation of prostate segmentation algorithms for MRI: The PROMISE12, Elsevier challenge, Medical Image Analysis (2014)

[24] Q. Liu, M. Fu, H. Jiang, X. Gong,Densely Dilated Spatial Pooling Convolutional Network using benign loss functions for imbalanced volumetric prostate segmentation, arXiv, 2018

[25] A. de Gelder, H. Huisman,Autoencoders for Multi-Label Prostate MR Segmentation, arXiv, 2018

[26] H. Jia, Y. Song, D. Zhang, H. Huang, D, Feng, M. Fulham, Y. Xia, W. Cai,3D Global Convolutional Adversarial Network for Prostate MR Volume Segmentation, arXiv, 2018

[27] O. Cickek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger, 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation, arXiv, 2016

[28] G. Haskins, J. Kruecker, U. Kruger, S. Xu, P.A. Pinto, B.J. Wood, P. Yan, Learning Deep Similarity Metric for Registration 3D MR-TRUS, arXiv, 2018

[29] Jian Ren, Iker Hacihaliloglu, Eric A. Singer, David J. Foran, Xin Qi,Adversarial Domain Adaptation for Classification of Prostate Histopathology Whole-Slide Images, ArXiv, 2018

[30] A.G. Chung, M.J Shafiee, D. Kumar, F.Khalvati, M.A. Haider, A. Wong, Discovery Radiomics for Multi-Parametric MRI Prostate Cancer Detection, arXiv, 2015

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, arXiv, 2015

[32] I. Reda, A. Shalaby, M. Elmogy, A.A ELfotouh, F. Khalifa, M.A. El-Ghar, E. Hosseini-Asl, G. Gimel'farb, N. Werghi, A. El-Baz,A comprehensive non-invasive framework for diagnosing prostate cancer, Elseiver, Computers in Biology and Medicine 81 (2017) 148–158