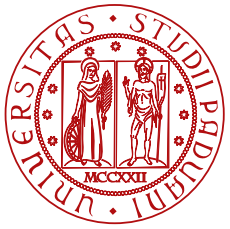

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Laurea Triennale in Ingegneria Informatica

Reti neurali convoluzionali per lo studio di varianti non codificanti in sequenze genomiche

Laureando

Alessandro Trigolo

Matricola 2043049

Relatore

Prof.ssa Cinzia Pizzi

Università degli Studi di Padova

Anno Accademico
2023/2024

Sommario

Questo elaborato mira ad approfondire il funzionamento delle reti neurali convoluzionali e di come questi modelli di deep learning siano in grado di estrarre significative informazioni da sequenze genomiche, analizzandone le zone non codificanti. In particolare, verranno comparati tre tool basati sulle CNN — DeepSEA, Basset e DeepSATA — e sarà fornita una revisione delle loro prestazioni.

Abstract

This thesis aims to deepen the understanding of the functioning of convolutional neural networks and how these deep learning models are able to extract significant information from genomic sequences, analyzing their non-coding regions. In particular, three CNN-based tools — DeepSEA, Basset and DeepSATA — will be compared and a review of their performance will be provided.

Indice

Indice delle Figure

Lista degli Acronimi

AI Intelligenza artificiale (*Artificial Intelligence*)

ANN Rete neurale artificiale (*Artificial Neural Network*)

ATP Adenosintrifosfato (*Adenosine TriPhosphate*)

CNN Rete neurale convoluzionale (*Convolutional Neural Network*)

ConvNet Rete neurale convoluzionale (*Convolutional Network*)

DHS *DNase I hypersensitive sites*

DL *Deep Learning*

DNA Acido desossiribonucleico (*DeoxyriboNucleic Acid*)

ENCODE *Encyclopedia of DNA Elements*

GD *Gradient Descent*

GPU Unità di elaborazione grafica (*Graphics Processing Unit*)

GWAS *Genome-Wide Association Study*

mRNA RNA messaggero

MSE Media del quadrato degli errori (*Mean Squared Errors*)

ncDNA DNA non codificante (*non coding DNA*)

NLL *Negative Log Likelihood*

NLP *Natural Language Processing*

NN Rete neurale (*Neural Network*)

PDB *Protein Data Bank*

PWM *Position Weight Matrix*

ReLU Funzione rettificatrice (*Rectified Linear Unit*)

RNA Acido ribonucleico (*RiboNucleic Acid*)

SGD *Stochastic Gradient Descent*

SVM *Support Vector Machines*

TAD *Topologically Associating Domains*

TF Fattori di trascrizione (*Transcription Factors*)

UTR Regione non tradotta (*UnTranslated Region*)

1

Introduzione

Ad oggi l'avanzamento della genomica — ramo della biologia molecolare che si occupa di studiare il genoma degli esseri viventi — si è rivelato notevolmente significativo al fine di approfondire e comprendere malattie legate alle mutazioni del genoma degli individui. Si stima che solamente una percentuale tra l'1% e il 2% del DNA contiene i *geni*, ovvero particolari regioni che contengono tutte le informazioni necessarie per la sintesi degli aminoacidi che poi comporranno le proteine [1], [2]. Ciò nonostante, la quasi totalità dei disturbi genomici è dovuta alle mutazioni nelle regioni non codificanti [3] — dette *varianti non codificanti*. Le mutazioni in queste zone del genoma, che apparentemente svolgono funzioni marginali, sono responsabili dello sviluppo di disturbi importanti, come le *malattie mendeliane*[1], l'epilessia, malattie cardiovascolari e soprattutto tumori — tra cui il cancro del colon-retto e il tumore al seno [3]–[11]. Risulta quindi vitale continuare a studiare gli effetti che le varianti non codificanti in sequenze genomiche hanno sugli individui.

Negli ultimi decenni, il progredire delle tecniche di *sequenziamento* [12] ha dato uno slancio rilevante allo sviluppo della *bioinformatica* — disciplina che unisce informatica e biologia. La bioinformatica si interessa a organizzare dati biologici in modo tale da facilitare l'accesso e l'inserimento di nuove informazioni (come il PDB [13]), sviluppare *tool* che permettono l'analisi dei dati e infine fornire una interpretazione significativa dei risultati ottenuti [14]. Più recentemente, l'accrescimento dei dati biologici e il costante avanzamento della potenza di calcolo hanno reso possibile l'applicazione di tecniche di *deep learning* (DL) anche nel campo della bioinformatica. Questo notevole progresso consente di scoprire e perfezionare soluzioni informatiche che permettano di delineare con sempre maggior precisione il ruolo che hanno le mutazioni nelle regioni non codificanti del DNA. Grazie a queste nuove tecnologie, la *genomica funzionale* — area della genomica che si interessa a descrivere le relazioni che ci sono tra i componenti di un sistema biologico, come geni e proteine [15] — ha avuto un forte impulso nell'approfondire le varianti non codificanti, tuttavia rimangono ancora significative lacune

[1] Le malattie mendeliane, causate dalla mutazione di un singolo gene, includono la fibrosi cistica e il morbo di Huntington.

nella comprensione della relazione tra mutazioni genetiche ed espressione genica. L'utilizzo di tecniche di deep learning risulta quindi cruciale per continuare la ricerca in questo ambito. L'obiettivo di questo elaborato è di discutere e confrontare tre tool che utilizzano le *reti neurali convoluzionali* per predire l'effetto delle varianti non codificanti su sequenze genomiche: DeepSEA [16], Basset [17] e DeepSATA [18].

Più precisamente, il Capitolo 2 introdurrà le basi della biologia molecolare, necessarie per comprendere interamente l'importanza delle varianti non codificanti. Successivamente, nel Capitolo 3 saranno approfonditi i principi fondamentali delle reti neurali e il modo in cui le reti convoluzionali possono essere utilizzate come ottimo strumento per predire l'effetto di sequenze genomiche. Il Capitolo 4 invece esaminerà i dettagli implementativi di ciascuno dei tre tool, indagando principalmente sugli aspetti legati alla codifica delle sequenze, alla struttura della rete e al *dataset* utilizzato per allenare il modello. Infine nel Capitolo 5 si riassumono le differenze analizzate nel capitolo precedente, offrendo una visione complessiva del confronto tra i tre tool.

2

Background biologico

La cellula è l'unità fondamentale della vita. La cellula è una piccola miscela acquosa con componenti chimici, racchiusi in una membrana, e possiede l'eccezionale capacità di replicarsi. Il primo elemento che permette di distinguere le cellule è la presenza di un nucleo. Vengono definite *procarioti* le cellule senza nucleo — che sono le più diffuse e compongono organismi unicellulari come i batteri e gli archei — mentre sono chiamate *eucarioti* le cellule che contengono un nucleo — le quali sono in genere più grandi e più complesse e costituiscono forme di vita multicellulari come animali, piante e funghi [19].

All'interno della cellula eucariote (Figura 2.1), immersi nel *citoplasma*, sono presenti diversi *organuli*, i quali svolgono una particolare funzione ciascuno. I *mitocondri* sono gli organuli più diffusi. Il loro compito è quello di generare energia chimica per la cellula: attraverso il processo di ossidazione di zuccheri e grassi, viene creata una sostanza che viene utilizzata nella

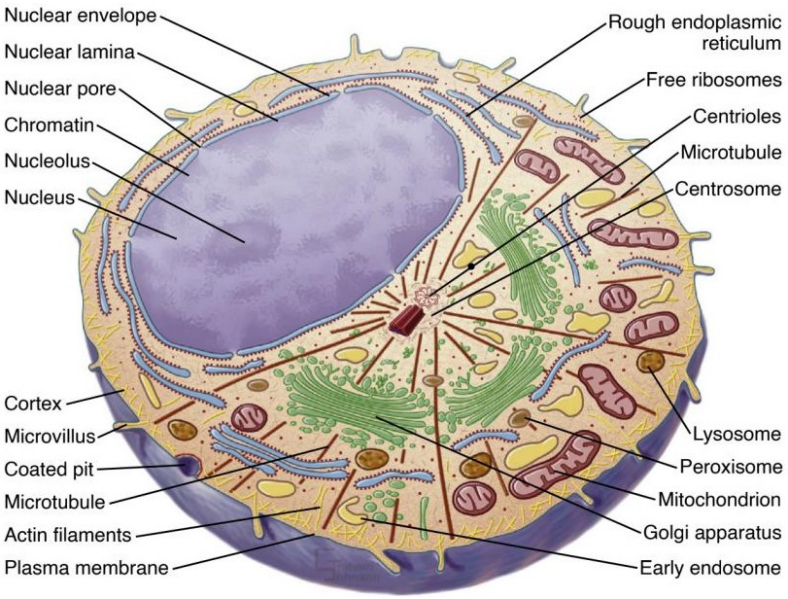

Figura 2.1: Rappresentazione schematica della cellula eucariote; si possono notare i principali organuli tra cui i mitocondri, lisosomi e perossiomi, il reticolo endoplasmatico, e il nucleo [2].

maggior parte delle attività cellulari[1]; questo processo è anche chiamato *respirazione cellulare* perché consumando l'ossigeno viene rilasciata anidride carbonica. Oltre ad essere la fonte energetica primaria della cellula, i mitocondri hanno anche importanti ruoli nella regolazione del metabolismo, del ciclo cellulare, delle risposte antivirali e anche della morte della cellula [19]–[21].

Il *reticolo endoplasmatico* è invece un organulo molto esteso e svolge molteplici funzioni. Tra questi compiti rientrano quelli di traslocazione di proteine e il ripiegamento delle proteine (*protein folding*) [19], [22]. I *lisosomi* si occupano di degradare e riciclare gli scarti cellulari e giocano un ruolo fondamentale per l'omeostasi della cellula[2], il suo sviluppo e il suo invecchiamento [23]–[25]. Infine, i *perossiomi* sono delle piccole vescicole che forniscono un ambiente protetto per gestire molecole tossiche come gli acidi grassi i quali sono smaltiti tramite la β-ossidazione [19], [26].

L'organulo più importante della cellula rimane il *nucleo*. Racchiuso nell'*involucro nucleare*, all'interno di questo organulo sono presenti tutte le informazioni genetiche, racchiuse in una lunga molecola di acido desossiribonucleico (comunemente noto come DNA), che, una volta impacchettato forma il *cromosoma* [2], [19]. La molecola di DNA è una struttura a doppia elica formata da *nucleotidi*. Osservando la Figura 2.2, i nucleotidi sono composti a loro volta da tre elementi fondamentali: una *base azotata*, uno *zucchero* e un *gruppo fosfato*[3]. Le basi azotate

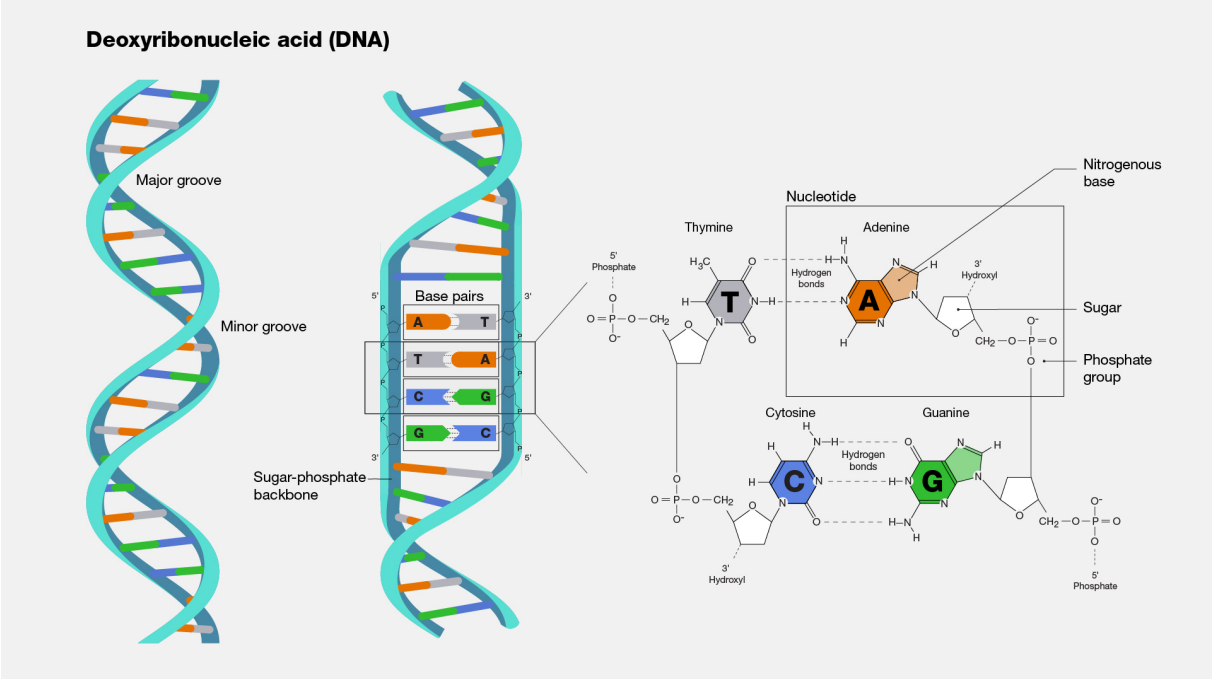

Figura 2.2: Rappresentazione schematica del DNA in cui si possono osservare le coppie di basi azotate, legate tra loro attraverso gli zuccheri e i gruppi fosfati [27].

[1]Questa sostanza è detta *adenosintrifosfato* o ATP ed ha una struttura simile ad un nucleotide: è infatti composta dall'Adenina, da uno zucchero e da tre gruppi fosfati.

[2]Con omeostasi cellulare si intende l'insieme di meccanismi necessari per mantenere ad un livello ottimale le funzioni della cellula.

[3]I gruppi fosfati hanno una carica negativa e forniscono alla molecola le proprietà di un acido.

sono quattro — Adenina (A), Citosina (C), Guanina (G) e Timina (T) — e si uniscono tra loro mediante dei legami ad idrogeno e secondo un preciso criterio: l'Adenina si lega solamente con la Timina (formando il legame *AT*) mentre la Citosina si unisce solo con la Guanina (creando la coppia *CG*) [1], [28]. Si osserva infine che il nucleotide di una coppia e quello successivo si legano mediante zucchero e gruppo fosfato sempre allo stesso modo: il gruppo fosfato di un nucleotide si lega sempre allo zucchero dell'altro. Di conseguenza, preso un filamento della doppia elica, le due estremità non sono uguali in quanto una termina con un gruppo fosfato (terminazione $5'$) e l'altra con uno zucchero (terminazione $3'$).

Attraverso una serie di ripiegamenti, una molecola di DNA lunga circa due metri riesce a raggomitolarsi in un cromosoma di grandezza inferiore a 2 micron (Figura 2.3). Il processo di *DNA-packaging* inizia avvolgendo la doppia elica di DNA attorno a delle proteine dette *istoni* e formando dei *nucleosomi*. In secondo luogo i nucleosomi si ammassano vicini tra loro formando una fibra, chiamata *cromatina* che, a sua volta si impacchetta su se stessa creando il cromosoma [29], [30].

2.1 Dogma centrale

La rilevanza del DNA è data delle informazioni essenziali che questa molecola contiene. Tali informazioni risiedono nei geni, che sono delle sequenze genomiche che codificano uno o più prodotti biologici operativi [32]. L'*espressione genica* è il processo che permette di utilizzare i dati contenuti nel gene per la creazione di macromolecole, come le proteine. Per esempio, le cellule della pelle a contatto con luce solare intensa possono esprimere geni che regolano la pigmentazione della pelle [33]. L'espressione genica è divisa in due fasi principali: la *trascri-*

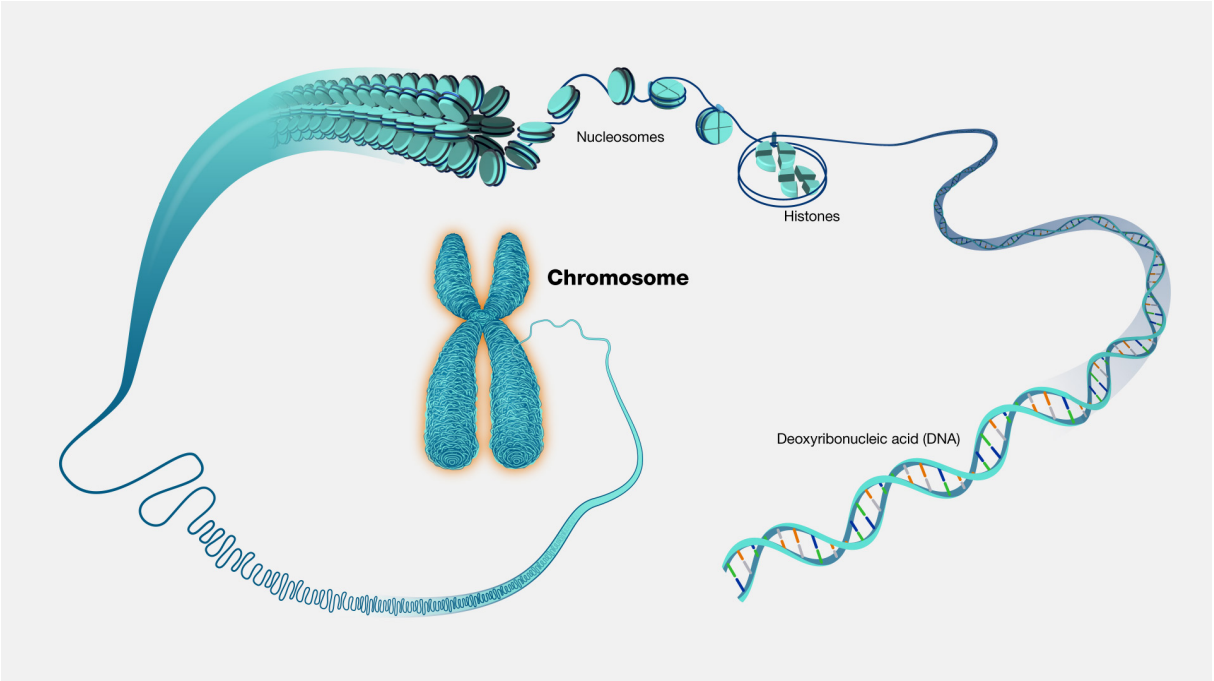

Figura 2.3: Il processo di impacchettamento del DNA che permette di compattare la struttura a doppia eilca nel cromosoma [31].

zione — che si occupa di produrre delle molecole di RNA che rispecchino il gene da esprimere — e la *traduzione* — la quale traduce le informazioni dell'RNA sintetizzando la proteina.

Nella prima fase dell'espressione genica, è necessario trascrivere il DNA in una molecola molto simile ovvero l'RNA — chiamato anche acido ribonucleico. Questa molecola differisce dall'acido desossiribonucleico per una base azotata — anziché la Timina è presente l'Uracile (U) — e per lo zucchero — da desossiribosio a ribosio [34]. La trascrizione del DNA in RNA inizia quando delle proteine, chiamate *fattori di trascrizione* (TF), attratte dagli *enhancer* del DNA, riconoscono la regione che delimita l'inizio della molecola del gene da esprimere, detta *promoter*. Dopo aver riconosciuto l'inizio della sequenza, queste proteine permettono ad un enzima chiamato *RNA polimerasi* di attaccarsi ed aprire la doppia elica del DNA [35]. Una volta aperta la doppia elica, inizia la vera e propria trascrizione in RNA: il filamento del DNA viene preso come modello per la creazione dell'RNA; in particolare il nucleotide dell'RNA sarà il complementare rispetto a quello del DNA (di conseguenza $A \to U$, $C \to G$, $G \to C$ e $T \to A$). Così facendo l'acido ribonucleico viene creato un nucleotide alla volta, analizzando quello del DNA [34]. La trascrizione termina nel momento in cui gli enzimi e le proteine incontrano la regione terminatrice del gene che determina la separazione dal filamento e la terminazione dell'RNA *messaggero* (mRNA) che contiene le informazioni presenti nel gene da esprimere. L'intero processo di trascrizione è illustrato nella Figura 2.4.

Prima di uscire dal nucleo l'RNA messaggero subisce una serie di elaborazioni necessarie per rendere le informazioni immagazzinare sicure: diverse sono le malattie che emergono per mutazioni presenti nell'mRNA tra cui la distrofia miotonica[4] [37]. La prima elaborazione viene

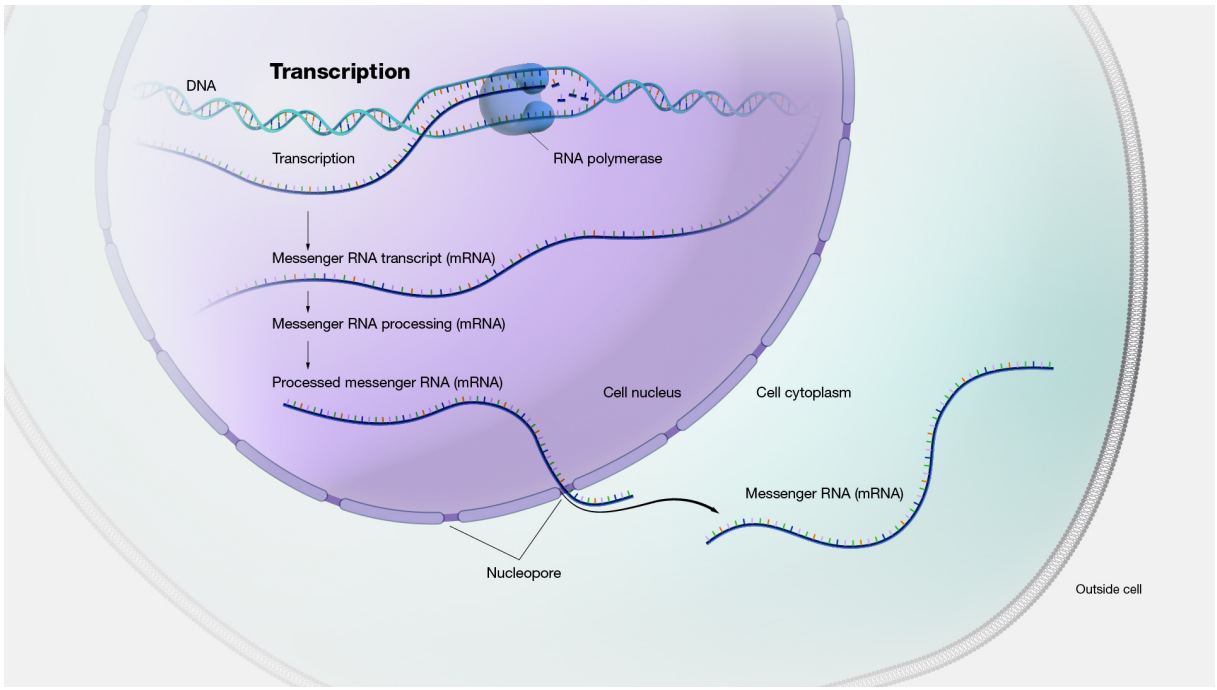

Figura 2.4: Il processo di trascrizione del DNA del gene in RNA mediante la RNA polimerasi [36].

[4]Le distrofie miotoniche sono patologie che colpiscono principalmente l'apparato muscolo-scheletrico.

chiamata 5′-*end capping* e si occupa di aggiungere alla terminazione 5′ dell'mRNA una Guanina attraverso un collegamento inusuale che garantisce maggiore stabilità alla molecola. In secondo luogo avviene lo *splicing* che si occupa di rimuovere le zone non codificanti — dette *introni*— dal gene trascritto mantenendo solo quelle che verranno utilizzate per essere sintetizzate in proteine — gli *esoni* — e quindi facilitando il processo di traduzione. Infine con il 3′-*end processing* viene aggiunta alla terminazione 3′ dell'mRNA una coda di Adenine — datta anche *poly*A *tail* — che, in maniera molto simile al 5′-*end capping* garantisce una stabilità del filamento di acido ribonucleico [38], [39].

Dopo essere uscito dal nucleo attraverso i *pori*, l'RNA messaggero raggiunge il citoplasma ed è pronto per iniziare la seconda fase dell'espressione genica, la traduzione. La traduzione non è altro che la traduzione dell'mRNA in un *polipeptide*, ovvero una sequenza di aminoacidi che compongono la proteina. Gli aminoacidi sono più di 20, di conseguenza anziché codificare un solo nucleotide dell'RNA messaggero, vengono codificati tre nucleotidi alla volta: questa tripletta viene chiamata *codone*. Durante la fase della traduzione, giocano un ruolo fondamentale i *ribosomi* i quali sono degli organuli nei quali avviene la traduzione. I ribosomi sono composti da due sotto unità, ciascuna delle quali ha tre siti per l'RNA di *trasporto* (*tRNA*). Delle due sotto unità del ribosoma, quella dimensionalmente minore si lega all'mRNA e agli *anticodoni* (sequenze specifiche di tre basi nel tRNA) e controlla che la traduzione avvenga con successo. La sotto unità più voluminosa invece si prende carico di catalizzare il legame peptidico tra l'aminoacido trasportato dal tRNA e la catena di aminoacidi in crescita [39]–[41]. In questo modo i ribosomi, analizzando codone dopo codone riescono a creare la catena polipeptidica mediante l'RNA di trasporto, come mostrato nella Figura 2.5.

Una volta creata la sequenza polipeptidica, inizia il processo di ripiegamento della proteina.

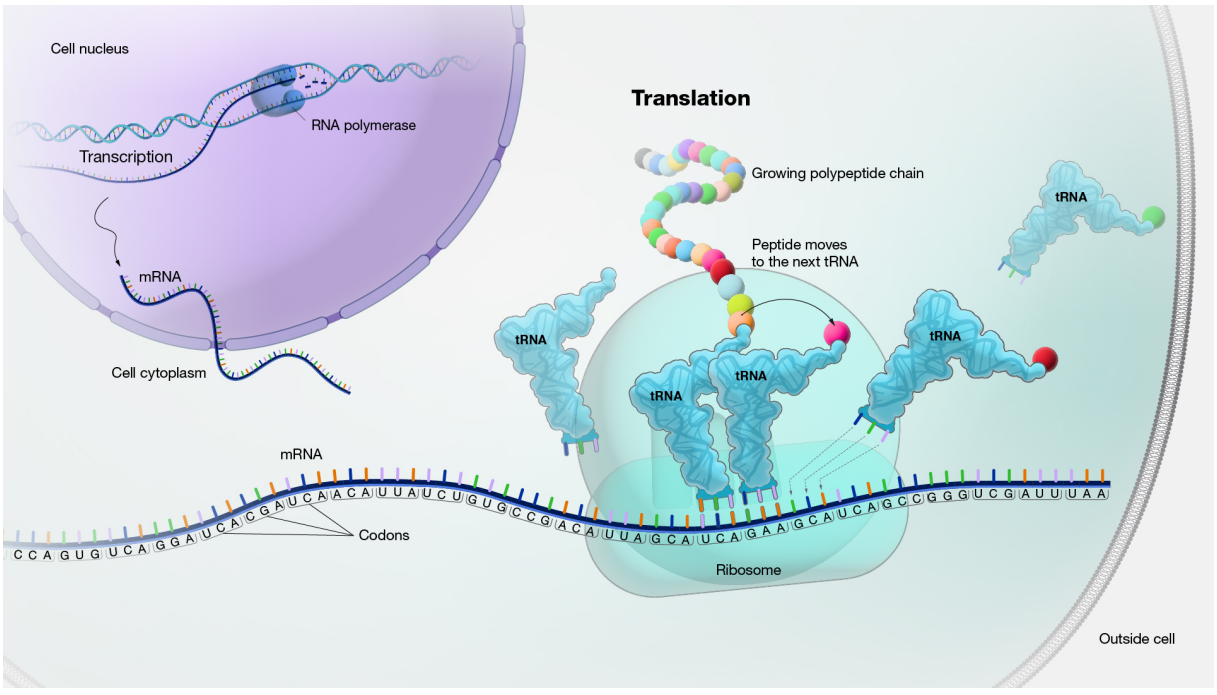

Figura 2.5: Il processo di traduzione da RNA messaggero a polipeptide attraverso il tRNA e i ribosomi [42].

In maniera molto simile a quanto visto per l'impacchettamento del DNA nel cromosoma, la sequenza di polipeptidi inizialmente si arrotola creando delle bobine che sono comunemente chiamate α-*helix*. Queste ultime poi si ripiegano nuovamente arrivando alla struttura terziaria della proteina, ovvero la proteina tridimensionale effettiva [43]. Una volta creata la proteina il gene è stato espresso definitivamente. Questo passaggio di informazioni dal DNA alla creazione della proteina è gergo definito come il *dogma della biologia molecolare*.

Come accennato all'inizio del capitolo, la cellula possiede la notevole capacità di replicarsi. In genere una cellula si duplica durante la crescita e lo sviluppo dell'organismo, quando deve essere rimpiazzata o rigenerata oppure nella riproduzione asessuata di alcuni micro organismi [44]. Il processo replicazione cellulare, chiamato *mitosi*, è preceduto dall'*interfase*, processo fondamentale in cui la cellula cresce di dimensioni e il DNA nei cromosomi si duplica, favorendo la replicazione cellulare. La mitosi può essere suddivisa in quattro fasi principali [44]–[47] le quali sono riassunte anche nella Figura 2.6:

1. Nella *profase* i cromosomi duplicati si condensano nel nucleo ed iniziano ad avvicinarsi dei microtuboli al nucleo, chiamati *centrosomi*; allo stesso tempo la membrana nucleare inizia a svanire;
2. Dopo che i microtuboli si sono attaccati ai cromosomi (fase intermedia detta *prometafase*) si giunge alla *metafase*, situazione in cui tutti i cromosomi sono allineati lungo la linea equatoriale della cellula;
3. Durante l'*anafase*, ciascuna coppia di cromosomi si divide e raggiunge i poli della cellula;
4. La fase finale della mitosi è la *telofase* nella quale le due cellule si dividono; le membrane nucleari delle due cellule si riformano attorno ai cromosomi divisi.

Affinché la mitosi abbia successo, è prima necessario duplicare il DNA all'interno della cellula. Il processo di replicazione di DNA che precede la mitosi è anche definito come *fase*

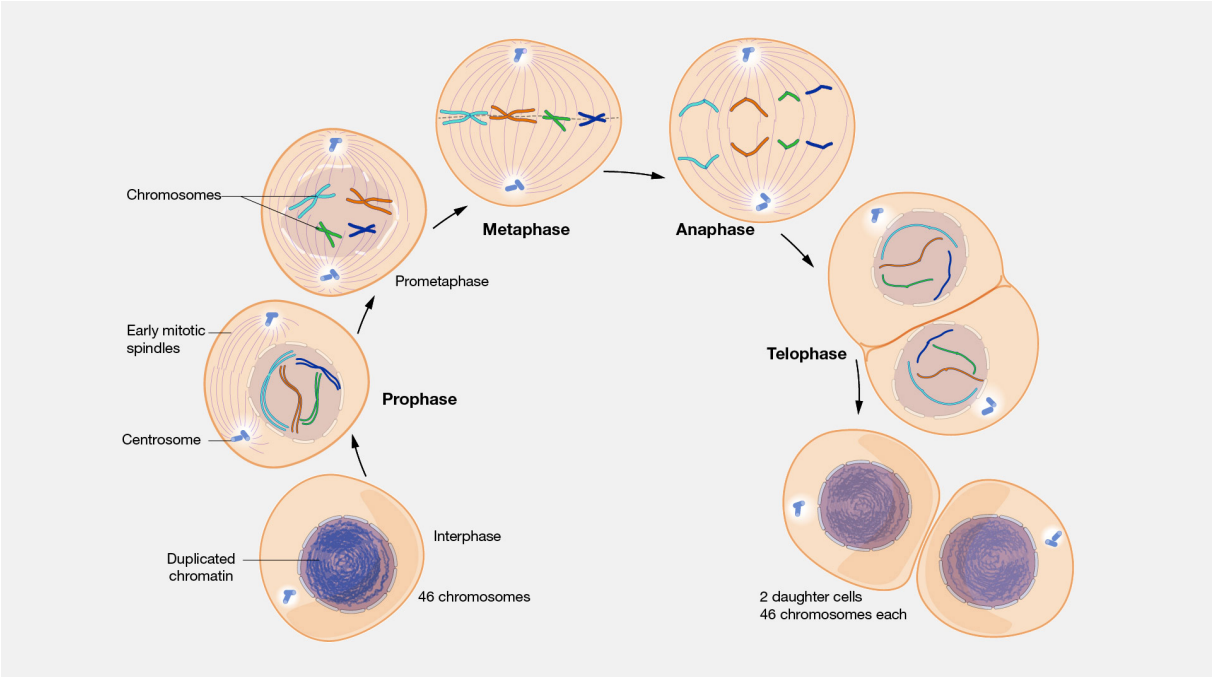

Figura 2.6: Rappresentazione delle quattro fasi che comprendono la mitosi cellulare [48].

di sintesi — S-phase. La duplicazione del DNA inizia con l'identificazione dell'*origine della replicazione*, ovvero una sequenza del DNA che specifica da quale punto della sequenza il DNA deve essere replicato (ci sono più di cento mila siti che segnalano un punto di orine nel DNA di una cellula). Una *proteina iniziatrice* è legata al punto di origine promuovendo l'attaccamento al DNA del *replisoma* che è composto da un enzima chiamato *elicasi* che si occupa di dividere i due filamenti di DNA procedendo nella direzione $5' \rightarrow 3'$. A questo punto il *RNA prime* inizia la sintesi del DNA favorendo l'attaccamento della *DNA polimerasi* entrambi i filamenti per duplicare il DNA. Essendo che il genoma è complementare, un filamento avrà un verso $5' \rightarrow 3'$ (*leading strand*) mentre l'altro filamento avrà verso opposto, $3' \rightarrow 5'$ (*lagging strand*). Di conseguenza, nel filamento concorde al replisoma, la polimerasi non incontrerà problemi nella duplicazione, invece nel filamento $3' \rightarrow 5'$ il DNA dovrà essere duplicato a segmenti, detti *frammenti di Okazaki* che verranno collegati tramite la *DNA ligasi* [49]–[52]. La Figura 2.7 racchiude quanto descritto sulla fase di sintesi.

2.2 VARIANTI NON CODIFICANTI

Come descritto fino ad ora, il ruolo del DNA è fondamentale in quanto trasmesso da cellula a cellula durante la replicazione per poi essere utilizzato nell'espressione genica creando le proteine. Alle regioni del DNA che prendono parte all'espressione genica, si contrappongono le regioni che non vengono codificate, come gli enhancers ed i promoters, ma anche gli introni di un gene. Le regioni non codificanti del DNA rappresentano tra il 98% e il 99% dell'intera molecola: si pensava che queste zone non avessero una funzione effettiva, tanto che vennero definite *junk DNA*, ovvero DNA spazzatura. Contrariamente a quanto si supponeva, mutazione

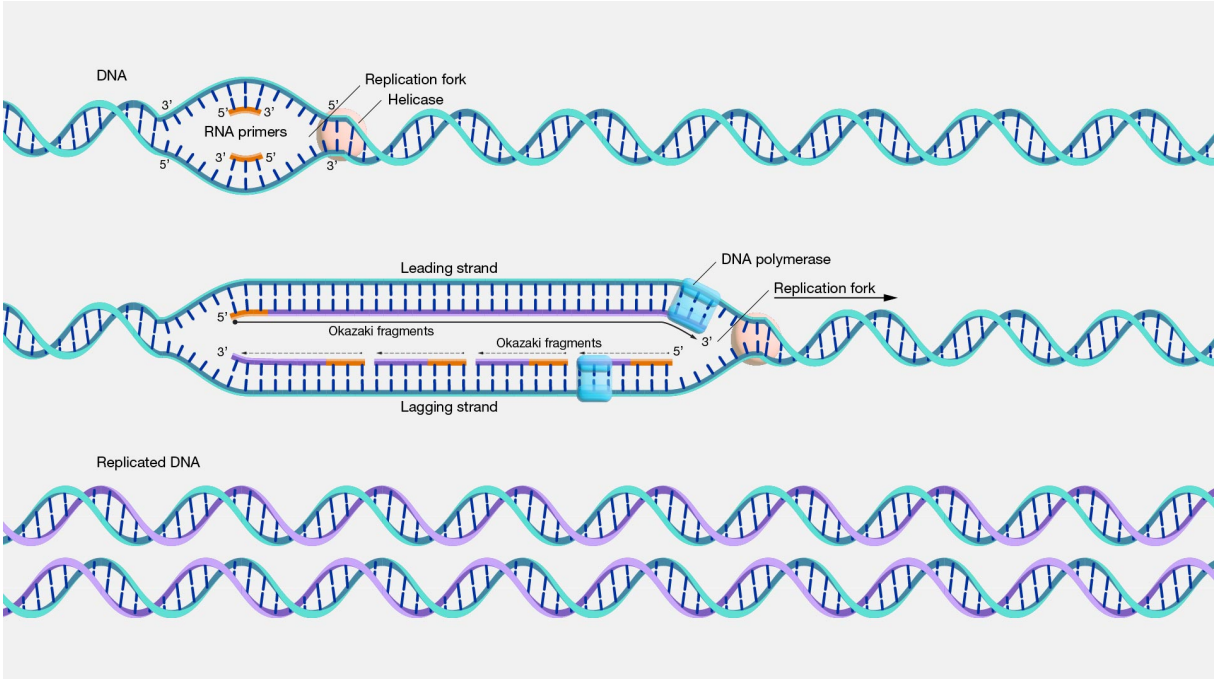

Figura 2.7: Il processo di replicazione del DNA durante la fase di sintesi [53].

in queste zone può causare diversi disturbi genetici in quanto l'organizzazione della cromatina, il processo di replicazione del DNA e l'espressione genica possono essere compromessi. Nel comprendere l'associazione tra varianti genomiche e disturbi da esse causati, il *Genome-Wide Association Study* (GWAS) ha svolto un ruolo rilevante, analizzando genomi di numerosi soggetti e correlando la presenza di una mutazione e un disturbo. Tra le varianti identificata dal GWAS, diverse sono mutazioni non codificanti [3], [54], [55].

Alcune varianti non codificanti possono alterare lo splicing di un gene trascritto. Come accennato precedentemente, lo splicing è il processo che permette la rimozione delle zone non codificanti del gene (gli introni) prima che questo sia tradotto in una proteine. Lo splicing fa parte dei processi che l'mRNA subisce prima di lasciare il nucleo ed iniziare la traduzione. All'interno degli introni sono presenti diverse sequenze che sottolineano l'inizio di un esone, tra cui il *donor*, l'*acceptor*, i *branch points* e i *tratti polipirimidinici*. Le variazioni in queste regioni possono condurre alla non codifica di una esone o alla conservazione di un introne: più del 15% dei disturbi ereditari sono dovuti proprio a questi inconvenienti. Oltre agli introni, anche altre regioni non tradotte (UTR) dell'mRNA possono condurre ugualmente a disturbi ereditari. Queste regioni sono fondamentali per gestire il processo seguente alla trascrizione: si occupano infatti di rendere il filamento di mRNA più stabile e robusto e rendono la sua localizzazione più semplice per poter iniziare il processo di traduzione [4], [8].

Le mutazioni non codificanti, oltre a verificarsi nell'RNA messaggero, possono anche occorrere nelle regioni non codifcianti del DNA (ncDNA), come i promoters i quali attirano i fattori di trascrizione (TF). Queste proteine sono responsabili di agganciare alla molecola di DNA la RNA polimerasi, che si occupa di sciogliere la doppia elica ed inizia la trascrizione del gene. La variazione di queste importanti regioni rende la trascrizione del gene incorretta, conducendo ad un'erronea traduzione in proteina. Le variazione dei promoter sono causa di malattie mendeliane e di alcuni tipi di tumori. Per lo stesso motivo, anche le mutazioni in altre sequenze *cis-regolatorie* del DNA — come enhancers e *silencers*[5] — conducono a malattie ereditarie. Tra questi disturbi, oltre a quelli citati precedentemente si aggiungono l'aritmia cardiaca, la sindrome della gamba senza riposo[6] e agenesia pancreatica[7] [3], [4], [6], [8].

In aggiunta alle variazioni di enhancers e promoters del DNA non codificante, le mutazioni possono anche causare anche l'espasione dei *tandem repeats*, sia negli esoni, che in regioni non codificanti del DNA. I tandem repeats sono delle lunghe ripetizioni di ridotte sequenze di DNA. Le conseguenze delle viariazioni dei repeat codificanti sono ben note (tra cui il morbo di Huntington), contrariamente all'effetto dell'espasione dei repeat non codificanti. A queste mutazioni si associano disturbi nella fase di trascrizione del DNA e l'intrappolamento delle proteine coinvolte durante lo splicing [4].

Le mutazioni del ncDNA, possono anche introdurre dei cambiamenti strutturali della croma-

[5] A differenza degli enhancer, che stimolano una particolare trascrizione genica, lo scopo dei silencers è quello di ridurre o inibire la trascrizione di un particolare gene

[6] Questa sindrome neurologica suscita al soggetto una insopportabile necessitàdi muovere gli arti inferiori.

[7] Questo disturbo determina l'assenza di massa pancreatica sin dalla nascita.

tina, fibra che costituisce i cromosomi. In particolare, la cromatina, mentre si impacchetta per formare il cromosoma (Figura 2.3), forma delle particolari regioni, dette TAD (*Topologically Associating Domains*), le quali contengono gruppi di geni che interagiscono frequentemente tra loro. Ne consegue che anche una piccola variazione strutturale può condurre ad una espressione genica imperfetta, causata da una socorretta interazione tra enhancers e promoters. Si è osservato che la variazione strutturale delle TAD, oltre a causare disturbi mendeliani, può portare a diverse malattie neurologiche [4], [56].

Contrariamente alle mutazioni nel DNA codificante, gli effetti delle variazioni nel ncDNA sono tuttora poco comprese. Questo è dovuto alla difficoltà di determinare se una variante non codificante influenzi effetto sul *fenotipo* — ovvero l'insieme delle caratterstiche strutturali e funzionali di un individuo. Diversi sono i tool bioinformatici che sono stati sviluppati per riconoscere se una mutazione non codificante abbia un effetto funzionale, ma, poiché tali funzioni dipendono anche dal contesto specifico della sequenza le previsione possono rivelarsi imprecise, fornendo quindi informazioni non del tutto chiare sull'effetto fenotipico della mutazione [57], [58].

Accenna anche alle DHS e agli Histone-marks, cita direttamente DeepSEA

3

Reti neurali

Il primo modello di intelligenza artificiale (AI) risale al 1943, dove E. McCulloch e W. Pitts cercarono di modellare un neurone come una semplice funzione predefinita. Nel modello, il neurone, generava un valore in output nel caso in cui le variabili booleane di input, una volta elaborate, superavano una soglia prestabilita [59, "A logical calculus of the ideas immanent in nervous activity"]. Poco dopo, nel 1950, Alan Turing, pubblicò un articolo che definiva una metodologia per testare l'intelligenza di un modello [60, "Computing machinery and intelligence"]. Questo test — noto anche come *imitation game* — consisteva nel valutare se una macchina potesse imitare l'intelligenza umana stabilendo così un obiettivo per il campo dell'intelligenza artificiale, termine che venne conianto per la prima volta nella conferenza di Dartmouth nel 1956.

Dopo due anni, nel 1958, lo psicologo F. Resenblatt introdusse il *percettrone* che, a differenza del modello del '43, processava input non booleani e disponeva di pesi per bilanciare l'output [61, "The perceptron: a probabilistic model for information storage and organization in the brain."]. Anche se il percettrone sarà alla base delle reti neurali artificiali moderne, nei dieci anni successivi alla pubblicazione dell'articolo, le aspettative iniziali non vennero soddisfatte. Nel 1968 venne pubblicato un libro il quale analizzava le prestazioni del percettrone e constatava le forti limitazioni del modello, come l'impossibilità di risolvere problemi non linearmente separabili [62, "Perceptrons"]. In seguito ad un secondo articolo del 1973, dove si evidenziavano gli scarsi risultati ottenuti in paragone con le grandi aspettative, iniziò il *Primo Inverno dell'Intelligenza Artificiale* dove fino alla metà degli anni Ottanta molte organizzazioni governative smisero di finanziare la ricerca sull'Intelligenza Artificiale (AI). L'Inverno della AI terminò nel 1985 con l'introduzione del *Gradient Descent Optimization*, algoritmo che permetteva di aggiornare i pesi in modo tale da minimizzare l'errore in una rete. Un anno dopo venne introdotto l'algoritmo della *back-propagation*, fondamentale per lo sviluppo di reti neurali, costituite da più livelli di neuroni, ciascuno dei quali è collegato al livello successivo [63, "Learning representations by back-propagating errors"].

Nonostante il grande sviluppo nella parte algoritmistica, l'hardware non era computazionalmente prestante da supportare le richieste di calcolo delle reti neurali artificiali. Questa carenza

nella potenza di calcolo portò al *Secondo Inverno dell'Intelligenza Artificiale*, periodo in cui l'interesse scientifico si spostò su modelli che richiedevano meno potenza di calcolo, come le *Support Vector Machines* (SVM) introdotte nel 1963. Il Secondo Inverno della AI terminò a metà degli anni Novanta quando il progresso dell'hardware riuscì a soddisfare i requisiti computazionali dei modelli basati su reti neurali. Il costante sviluppo culminò nell'ultimo ventennio quando venne introdotta la GPU, che, insieme all'aumento dei dati disponibili, accelerò notevolmente i progressi nel campo dell'AI [64], [65].

3.1 PRINCIPI DI BASE ED EVOLUZIONE

Le reti neurali artificiali (ANN) — comunemente note come reti neurali (NN) — mirano a rappresentare un modello semplificato del cervello, trattato come una struttura composta da neuroni. Risulta quindi essenziale comprendere il funzionamento del singolo *neurone artificiale* per poi esplorare la struttura di una rete neurale, che è un insieme di neuroni artificiali, collegati tra loro secondo precisi criterio

Come un neurone biologico, il neurone artificiale (Figura 3.1) riceve un numero indefinito n di segnali in *input*, che possono essere rappresentati con la notazione $x_1, x_2, \ldots, x_n$. Tali valori possono essere raggruppati nel vettore di input $\mathbf{x} = [x_1, x_2, \ldots, x_n]$. Per descrivere il risultato di tutti i segnali in ingresso del neurone, si introduce una funzione g, che generalmente è una semplice somma algebrica dei segnali di input x_i, con $i \in [1, n]$. Ogni segnale di input, prima di essere sommato viene moltiplicato per il rispettivo peso (*weight*) w_i, con $i \in [1, n]$, che appartiene al vettore dei pesi $\mathbf{w} = [w_1, w_2, \ldots, w_n]$. Ne consegue che il segnale in uscita, indicato con v, comprenderà la somma del prodotto del segnale i-esimo (x_i) con il rispettivo peso i-esimo (w_i):

$$v = g(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x} \tag{3.1}$$

Per attivare un neurone, è necessario che il segnale v prodotto in *output* sia superiore ad una soglia prescelta, indicata con b. Questo si traduce nel sottrarre tale valore al segnale v e verificare

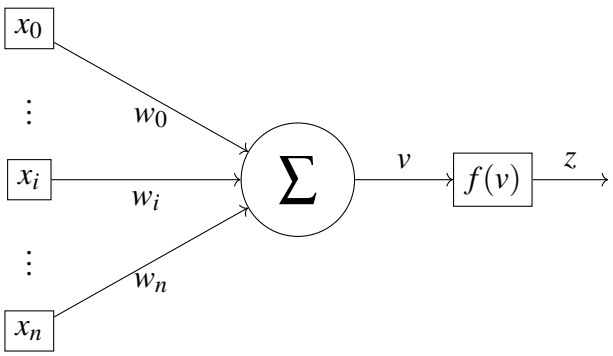

Figura 3.1: Rappresentazione schematica del funzionamento di un neurone artificiale. I segnali in input X_i vengono moltiplicati con i rispettivi pesi W_i e sommati tra loro; il risultato v viene processato dalla funzione di $f(v)$ che restituisce l'output z.

che il risultato sia maggiore o minore di zero.

$$\sum_{i=1}^{n} w_i x_i \geq b \quad \Rightarrow \quad \sum_{i=1}^{n} w_i x_i - b \geq 0 \tag{3.2}$$

Il valore sottratto alla somma, detto *bias* può essere inglobato nella somma algebrica traducendolo in un ulteriore input x_0 con associato il peso w_0.

$$v = g(\mathbf{w}, \mathbf{x}) = \sum_{i=0}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x} \tag{3.3}$$

Il criterio che descrive l'attivazione del neurone artificiale è riassunto nella *funzione di attivazione*, chiamata $z = f(v)$. La funzione di attivazione più semplice che descrive tale criterio la funzione *gradino* $\mathbf{1}(v)$, descritta graficamente nella Figura 3.2:

$$z(v) = \mathbf{1}(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases}$$

Analogamente alle reazioni agli stimoli esterni di un neurone biologico, anche neurone artificiale deve essere in grado di rispondere in un determinato modo quando riceve determinate informazioni. Affinchè il neurone sia capace di comporatarsi correttamente di fronte a specifici dati, è fondamentale allenarlo: è dunque necessario utilizzare un *training dataset* che contenga numerosi vettori di input $\mathbf{x}$ e, per ciascuno di essi sia presente la risposta corretta che il neurone dovrebbe fornire. Formalmente definiamo il dataset $\mathscr{D}$ come:

$$\mathscr{D} = \left\{ \{X_1, y_1\}, \{X_2, y_2\}, \ldots \{X_j, y_j\}, \ldots \{X_M, y_M\} \right\}$$

Il dataset[1] è composto da M vettori di input, definiti con la notazione X_j ($j \in [1, M]$), e da esattamente M risposte y_j, che rappresentano il comportamento atteso del neurone se il vettore

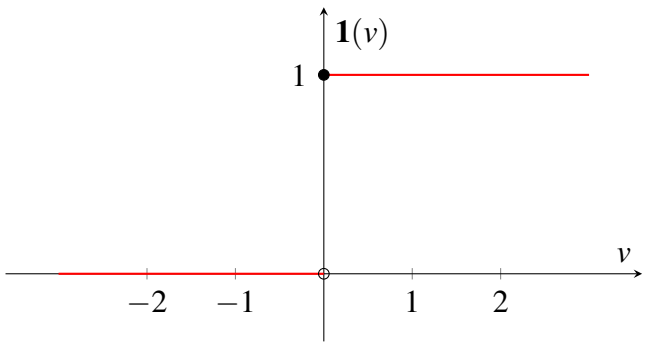

Figura 3.2: Grafico della funzione gradino $\mathbf{1}(v)$. Si osserva che vale zero per valori strettamente minori di zero e uno per valori maggiori o ugali a zero.

[1]Un dataset così definito è utilizzato nel *supervised learning*, dove all'interno del dataset sono presenti sia i vettori in ingresso che la risposta attesa. Si contrappone l'*unsupervised learning* — che non verrà trattato — dove sono presenti solo i vettori in ingresso e sono sconosciute le risposte attese.

di ingresso è X_j. È quindi possibile definire la matrice $\mathbf{X}$ che contiene esattamente M vettori — ciascuno in una riga — i quali sono composti esattamente da n componenti (o *feature*), che sono le componenti associate al vettore dei pesi $\mathbf{w}$, precedentemente introdotto. Alla matrice $\mathbf{X}$ è associato il vettore $\mathbf{y}$, anch'esso di dimensione M tale per cui la componente y_j sia la risposta attesa del vettore di input X_j.

$$\mathbf{X} = \begin{bmatrix} X_1^0 & X_1^1 & \cdots & X_1^i & \cdots & X_1^n \\ X_2^0 & X_2^1 & \cdots & X_2^i & \cdots & X_2^n \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_j^0 & X_j^1 & \cdots & X_j^i & \cdots & X_j^n \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_M^0 & X_M^1 & \cdots & X_M^i & \cdots & X_M^n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \\ \vdots \\ y_M \end{bmatrix}$$

Da questo deriva che il vettore X_j è dimensionalmente compatibile con il vettore di pesi $\mathbf{w}$ in quanto hanno esattamente la stessa dimensione.

$$X_j = \left[X_j^0, X_j^1, \ldots, X_j^n\right] \qquad \mathbf{w} = [w_0, w_1, \ldots, w_n]$$

Perciò il dataset $\mathscr{D}$ può essere riscritto attraverso la matrice $\mathbf{X}$ ed il vettore di risposte attese associato $\mathbf{y}$.

$$\mathscr{D} = \{\mathbf{X}, \mathbf{y}\}$$

Con un dataset di partenza si può definire un algoritmo generico che sia in grado di descrivere il processo di apprendimento di neurone artificiale. Come descritto nell'Algoritmo 1, dopo aver inizializzato il vettore di pesi $\mathbf{w}$, per ogni vettore X_j del dataset $\mathscr{D}$, vengono calcolati il segnale v e il valore della funzione di attivazione $z = f(v)$. L'idea alla base dell'algoritmo è quella di modificare il vettore di pesi $\mathbf{w}$ in funzione del risultato della funzione di attivazione rispetto al segnale processato. In questo modo, una volta processati tutti i vettori presenti nel dataset, è

Algoritmo 1 Allenamento del neurone artificiale

Input: Dataset $\mathscr{D}$

Inizializza $\mathbf{w}$ con numeri casuali
$j \leftarrow 1$
while $j \leq M$ **do**
  Calcola v in funzione di X_j e $\mathbf{w}$
  Determina z in rapporto a v e y_j
  Modifica $\mathbf{w}$ a seconda del risultato z
  $j \leftarrow j + 1$
end while

possibile constatare se il neurone sia stato in grado di apprendere il comportamento desiderato in funzione del vettore in ingresso [64].

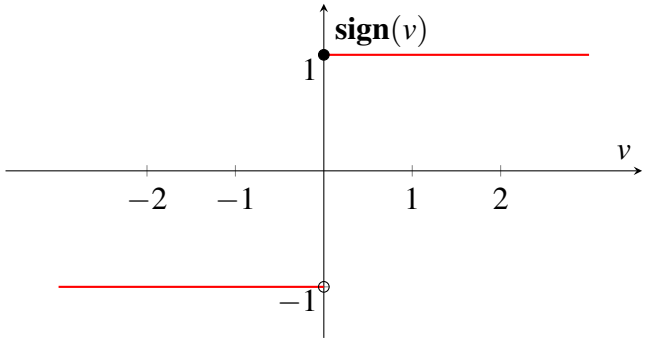

Figura 3.3: Grafico della funzione segno **sign**(v). Si osserva che vale -1 per valori strettamente minori di zero e 1 per valori maggiori o ugali a zero.

Uno tra i modelli di neuroni artificiali utilizzati nelle reti neurali è il percettrone. Dato un vettore in ingresso X_j e un vettore di pesi $\mathbf{w}$, il segnale v è dato dalla somma algebrica dei prodotti di ciascuna delle componenti:

$$v = g\left(\mathbf{w}, X_j\right) = \sum_{i=0}^{n} w_i X_j^i = \mathbf{w} \cdot X_j$$

La funzione di attivazione $f(v)$ del percettrone è la funzione *"segno"* (Figura 3.3) — chiamata anche *gradino bipolare* — definita come segue.

$$z(v) = \mathbf{sign}(v) = \mathbf{sign}(\mathbf{w} \cdot X_j) = \begin{cases} 1 & \text{se } v \geq 0 \\ -1 & \text{se } v < 0 \end{cases}$$

Il particolare più importante del percettrone è la *learning rule*, ovvero il criterio secondo il quel il vettore di pesi è aggiornato a seconda della predizione del modello. Se la predizione z_j rispetto al vettore X_j è diversa dalla risposta attesa y_j, allora il vettore di pesi è modificato come segue [64], [66].

$$w_i = w_i + y_j X_j^i$$

Al percettrone si aggiunge anche il *neurone sigmoideo*, che è molto simile al percettrone se non per la funzione di attivazione. Anzichè avere la funzione discontinua "segno", viene introdotta la funzione *sigmoide* $\sigma(v)$, che è una funzione continua e restituisce valori compresi tra zero e uno (Figura 3.4). Questa funzione, definita di seguito, rimuove le discontinuità della funzione di attivazione del percettrone e fornisce un raggio di valori reali, piuttosto che un risultato binario.

$$\sigma(v) = \frac{1}{1+e^{-v}} = \frac{1}{1+e^{-\mathbf{w} \cdot X_j}}$$

Questa importante differenza rende il neurone sigmoideo più flessibile rispetto al percettrone classico e più adatto all'utilizzo nelle reti neurali [66].

La learning rule introdotta nel percettrone, per quanto semplice, spesso viene sostituita dalla

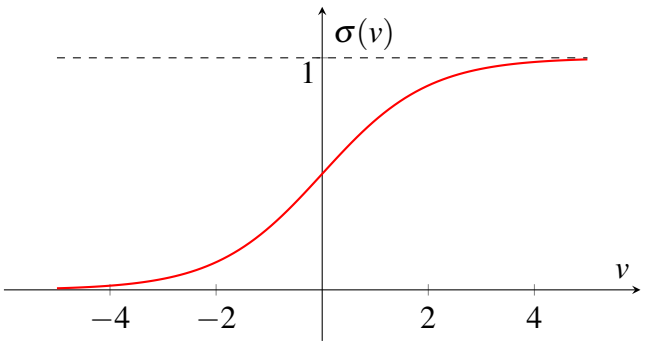

Figura 3.4: Grafico della funzione sigmoide $\sigma(v)$.

learning rule derivante dal *Gradient Descent* (GD, rappresentato nella Figura 3.5), un approccio che mira a trovare il minimo di una funzione differenziabile, spesso chiamata funzione "costo". Fornito il vettore di pesi $\mathbf{w} = [w_0, w_1, \ldots, w_n]$, durante l'allenamento del neurone, l'algoritmo cerca di modificare tale vettore cercando di minimizzare sempre di più l'errore tra il valore predetto dal modello e il risultato atteso. Più precisamente, data una funzione di costo $C(w)$ — detta anche *loss function* — la learning rule del GD vale:

$$\mathbf{w} = \mathbf{w} - \eta \nabla C(\mathbf{w})$$

Dove η è un parametro positivo, chiamato *learning rate* e serve per bilanciare la rapidità con cui il vettore dei pesi sia aggiornato durante l'esecuzione dell'algoritmo. Il termine $\nabla C(\mathbf{w})$ è invece il gradiente della loss function. Il gradiente di una funzione a più dimensioni è un vettore che

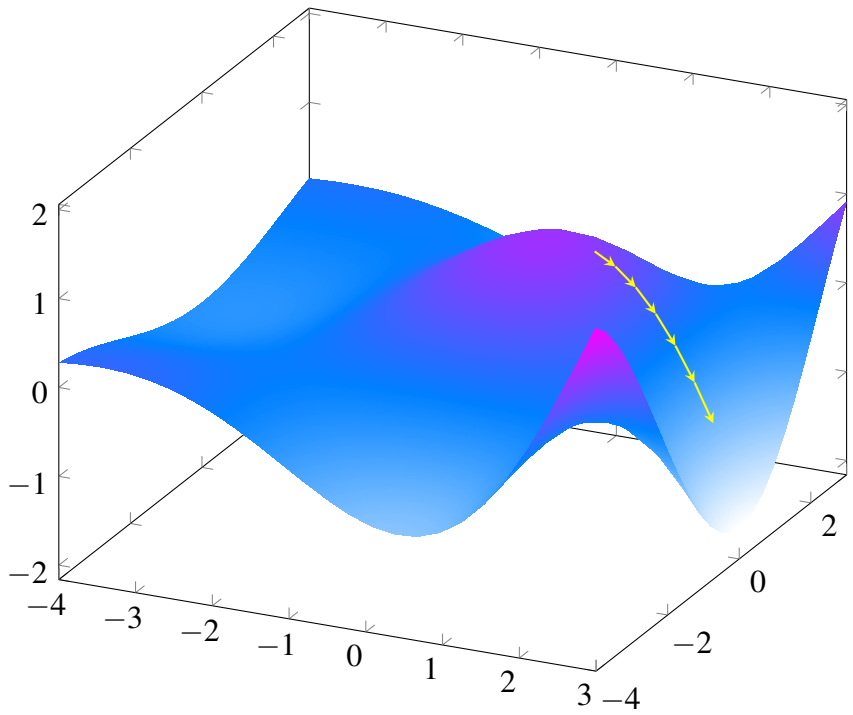

Figura 3.5: Funzionamento del gradient descent in una funzione bidimensionale. Si osserva che l'algoritmo, attraverso il gradiente della funzione, riesce a spostarsi verso il minimo (locale o assoluto) della funzione, come una biglia di vetro.

indica la direzione di massima crescita del valore della funzione; conseguentemente la direzione opposta è quella da seguire per raggiungere il minimo (locale o assoluto) della funzione. La Figura 3.5 mostra graficamente come l'approccio GD cerchi di raggiungere tale valore ad ogni iterazione. Si osserva che in questo caso la funzione rappresentata in due dimensioni ma in generale la loss function è n-dimensionale. Questa funzione può essere una qualsiasi funzione errore: una tra le più comuni è la funzione che calcola la media del quadrato degli errori (in gergo chiamata MSE, *Mean Squared Errors*). Per fare un esempio pratico, avendo il vettore di pesi $\mathbf{w}$, un dataset $\mathscr{D}$ — definito come in precedenza — ed una funzione di attivazione z, la MSE è definita come:

$$C(\mathbf{w}) = \frac{1}{M} \sum_{j=1}^{M} \left[y_j - z(\mathbf{w}, X_j) \right]^2$$

L'approccio adottato da questo algoritmo, per quanto efficacie non è completamente efficiente. La complessità computazionale richiesta è notevole: ad ogni iterazione dell'allentamento del neurone, è necessario trovare il valore della derivata della loss function, che si traduce nel computare la predizione di tutti i vettori di input X_j del dataset $\mathscr{D}$ e compararli con la risposta attesa y_j. Questo approccio, per dataset di dimensioni medio-grandi è molto poco efficiente, per quanto preciso. Inoltre è possibile che la discesa del gradiente conduca l'aggiornamento del vettore di pesi in un minimo locale della funzione errore, giungendo quindi ad una soluzione sub-ottima. Questi problemi possono entrambi essere gestiti da una variante di questo algoritmo, chiamata *Stochastic Gradient Descent* (SGD). Per ogni iterazione dell'algoritmo di allenamento, vengono presi un numero $m < M$ di vettori di input con le rispettive risposte attese e aggiornato il vettore $\mathbf{w}$ in base a questo sotto insieme del dataset, chiamato anche $mini-batch$. In questo modo la complessità computazionale viene drasticamente diminuita in quanto calcolare la derivata della loss function per una frazione ridotta del dataset richiede meno risorse. In aggiunta la casualità della scelta del mini-batch può evitare la convergenza in un minimo locale poiché rende la discesa lungo la funzione di errore leggermente più imprevedibile, diminuendo la possibilità di risultati sub-ottimi [66]–[68].

Una rete neurale è una struttura composta da neuroni artificiali i quali sono organizzati in livelli (*layers*). In una rete neurale multilivello, è sempre presente un input layer e un output layer. I livelli che si trovano tra questi sono detti livelli nascosti — *hidden* layers. I neuroni dello stesso livello non sono mai collegati tra loro ma solo con i neuroni del livello precedente e del livello successivo. Più precisamente, data una rete neurale, i neuroni del livello ℓ ricevono il segnale dai neuroni del livello $\ell - 1$ e, dopo aver elaborato le informazioni, inviano il segnale processato ai neuroni del livello $\ell + 1$: si dice che la rete è di tipo *feedforward* in quanto non sono presenti cicli nella struttura. Osservando la Figura 3.6 si introduce una nuova notazione per le reti neurali: si definisce con $N_j^{(\ell)}$ il neurone j-esimo che si trova nel livello ℓ; conseguentemente l'output della sua funzione di attivazione è identificato da $a_j^{(\ell)}$. Il peso $w_{j,i}^{(\ell)}$ è il peso che collega il neurone $N_i^{(\ell-1)}$ al neurone $N_j^{(\ell)}$, ovvero il collegamento tra il neurone alla posizione i del livello precedente $\ell - 1$ e il neurone $N_j^{(\ell)}$. Inoltre, per ogni neurone vengono esplicitati i bias:

$b_j^{(\ell)}$ indica il bias associato al j-esimo neurone nel livello ℓ. A questo punto, presupponendo che si utilizzino dei neuroni sigmoidei all'interno della rete neurale, l'output del neurone $a_j^{(\ell)}$, vale esattamente:

$$a_j^{(\ell)} = \sigma(v) = \sigma\left(\sum_i w_{j,i}^{(\ell)}\, a_i^{(\ell-1)} + b_j^{(\ell)}\right)$$

Questa formula indica che l'attivazione del neurone $N_j^{(\ell)}$ è dato dalla sigmoide calcolata sull'input del neurone. L'input è dato dalla somma algebrica degli output dei neuroni del livello precedente, moltiplicati per i rispettivi pesi e sommati al bias de neurone. Questa notazione può essere riscritta in forma matriciale per renderla più elegante. Si definisce la matrice $\mathbf{W}^{(\ell)}$ che contiene i pesi che collegano i neuroni del livello $\ell-1$ al livello ℓ. Allo stesso modo, si può definire il vettore $\mathbf{b}^{(\ell)}$, che contiene i bias dei neuroni nel livello ℓ. Infine il vettore $\mathbf{a}^{(\ell)}$ contiene i valori delle funzioni di attivazione di tutti i neuroni al livello ℓ. La formula può quindi essere riscritta come segue:

$$\mathbf{a}^{(\ell)} = \sigma\left(\mathbf{W}^{(\ell)}\,\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right) = \sigma\left(\mathbf{v}^{(\ell)}\right)$$

Questo risultato fornisce una visione di insieme dell'output del livello ℓ, rispetto che al singolo neurone $N_j^{(\ell)}$. Inoltre si osserva che è stato definito un vettore $\mathbf{v}^{(\ell)}$ il cui valore non è altro che l'input della funzione sigmoide, ovvero la somma pesata degli input del livello $\ell-1$.

Come per il caso del singolo neurone artificiale, anche una ANN deve essere allenata. Per allenare una rete neurale si utilizza l'algoritmo della *backpropagation*, introdotto per la prima volta nel 1986. La backpropagation, sfruttando i principi del GD, modifica i pesi della rete con

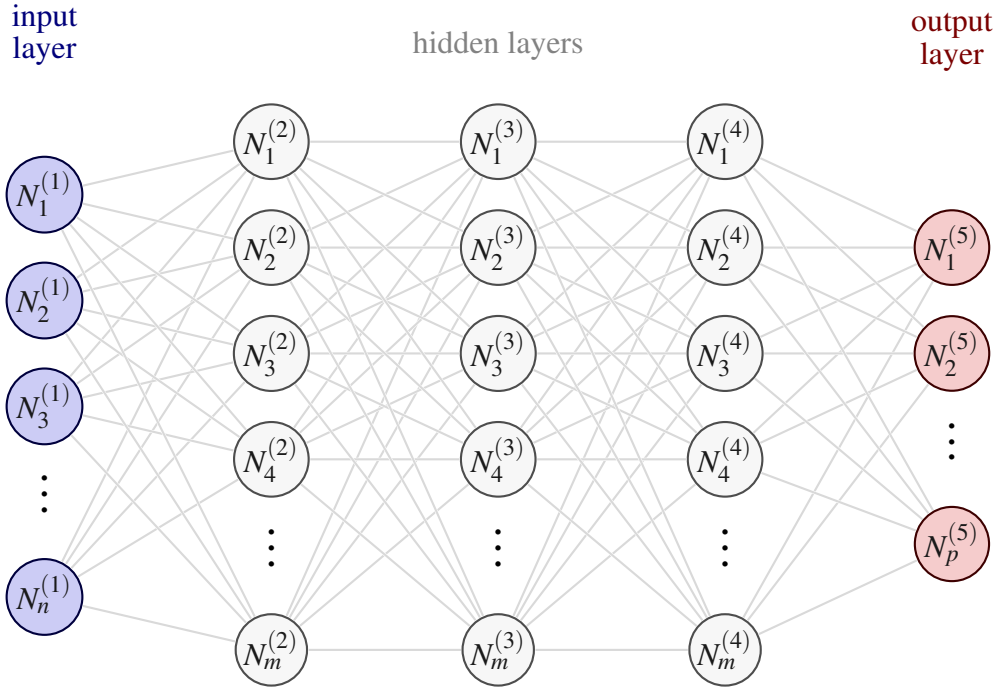

Figura 3.6: Rappresentazione di una rete neurale multilivello.

l'obettivo di minimizzare una *cost function* C. Questo si traduce nel calcolare il gradiente di C, in particolare la derivata parziale della funzione C rispetto al peso di un neurone e rispetto al suo bias:

$$\frac{\partial C}{\partial w_{j,i}^{(\ell)}} \qquad\qquad \frac{\partial C}{\partial b_j^{(\ell)}}$$

Oltre a questo viene definito anche l'*errore del neurone* $N_j^{(\ell)}$, chiamato $\delta_j^{(\ell)}$ indicato come:

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial v_j^{(\ell)}}$$

Questo importante valore, indica di quanto varia la cost function in rapporto ad un piccolo cambiamento nell'input del neurone $N_j^{(\ell)}$. Supponendo di essere al livello ℓ, modificare il vettore $\delta^{(\ell)}$, quindi modificare di poco il vettore di input $\mathbf{v}^{(\ell)}$, può portare alla riduzione complessiva della loss function C, propagando ai livelli successivi il cambiamento dell'input al livello ℓ. L'algoritmo della backpropagation fornisce un modo per calcolare il valore $\delta_j^{(\ell)}$ e collegarlo alle derivate parziali.

La backpropagation può essere descritta attraverso quattro equazioni fondamentali [66]. La prima equazione fornisce un modo per calcolare il valore $\delta_j^{(L)}$, ovvero l'errore di predizione nel livello di output L. Come si può osservare dall'equazione 3.4, tale valore è il prodotto di due derivate. La prima derivata parziale indica il cambiamento della cost function rispetto all'output del neurone $N_j^{(L)}$: se il neurone ha poca influenza sull costo finale, allora $\delta_j^{(L)}$ è un valore molto piccolo. La seconda derivata indica quanto la funzione sigmoide varia nel punto $v_j^{(L)}$.

$$\delta_j^{(L)} = \frac{\partial C}{\partial a_j^{(L)}} \, \sigma'\left(v_j^{(L)}\right) \tag{3.4}$$

La seconda equazione che descrive la backpropagation mette in relazione $\delta^{(\ell)}$ con l'errore del livello successivo $\delta^{(\ell+1)}$, come mostrato nell'equazione 3.5. Moltiplicare la matrice trasposta dei pesi $\left(\mathbf{W}^{(\ell+1)}\right)^T$ con l'errore $\delta^{(\ell+1)}$ è come, intuitivamente, misurare l'errore in uscite al livello ℓ. Questo risultato viene moltiplicato con $\sigma'\left(v^{(\ell)}\right)$ attraverso l'operatore "$\odot$", ovvero il prodotto *element-wise*[2]. Questo passaggio permette di propagare l'errore "indietro" attraverso la funzione di attivazione, fornendo quindi il valore di $\delta^{(\ell)}$. Questa equazione fornisce un modo per calcolare i pesi del livello precedente propagando all'indietro l'errore mediante la trasposta dei pesi e la derivata della funzione di attivazione.

$$\delta^{(\ell)} = \left[\left(\mathbf{W}^{(\ell+1)}\right)^T \delta^{(\ell+1)}\right] \odot \sigma'\left(v^{(\ell)}\right) \tag{3.5}$$

[2]Dati due vettori, $a = [a_1, a_2]$ e $b = [b_1, b_2]$, allora il loro prodotto vale $a \odot b = [a_1 b_1, a_2 b_2]$

La terza equazione (3.6), mette in relazione la derivata della loss function rispetto al bias di un neurone ($b_j^{(\ell)}$) e il suo errore $\delta_j^{(\ell)}$.

$$\frac{\partial C}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)} \tag{3.6}$$

Infine la quarta equazione (3.7) lega la derivata della cost function rispetto al peso di un neurone e l'errore del neurone. Più precisamente, l'errore $\delta_j^{(\ell)}$ del neurone $N_j^{(\ell)}$ è legato alla derivata della funzione di costo rispetto al peso $w_{j,i}^{(\ell)}$ mediante il valore in output $a_i^{(\ell-1)}$ del neurone i-esimo che si trova al livello precedente $\ell - 1$.

$$\frac{\partial C}{\partial w_{j,i}^{(\ell)}} = a_i^{(\ell-1)}\, \delta_j^{(\ell)} \tag{3.7}$$

Dopo aver definito le 4 equazioni fondamentali dell'algoritmo, è possibile comprendere lo pseudocodice della backpropagation, fornito nell'Algoritmo 2. Si osserva che questo algoritmo calcola i valori degli errori dei neuroni partendo dalla fine e propagandoli all'indietro giungendo infine a computare il gradiente della funzione di costo C.

Algoritmo 2 Backpropagation

Input: insieme delle funzioni di attivazione dell'input layer, $\mathbf{a}^{(1)}$

Per ogni $\ell \in [2, L]$, calcolare $\mathbf{v}^{(\ell)} = \mathbf{W}^{(\ell)}\, \mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}$ e $\mathbf{a}^{(\ell)} = \sigma\left(\mathbf{v}^{(\ell)}\right)$

Calcolare l'errore di predizione $\delta_j^{(L)} = \frac{\partial C}{\partial a_j^{(L)}}\, \sigma'\left(v_j^{(L)}\right)$

Per ogni $\ell \in [L-1, 2]$, calcolare $\delta^{(\ell)} = \left[\left(\mathbf{W}^{(\ell+1)}\right)^T \delta^{(\ell+1)}\right] \odot \sigma'\left(v^{(\ell)}\right)$

Output: gradiente della cost function, fornito da $\frac{\partial C}{\partial w_{j,i}^{(\ell)}} = a_i^{(\ell-1)}\, \delta_j^{(\ell)}$ e $\frac{\partial C}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)}$

Questo fondamentale algoritmo, calcolando il gradiente della funzione costo per ogni iterazione — o, in gergo, *epoca* — dell'allenamento della rete cerca di raggingere il minimo della loss function C ottimizzando i valori dei pesi. In questo modo la rete neurale artificiale è in grado di apprendere dai dati forniti in ingresso in modo tale da fornire predizioni valide a dati non presenti nel dataset. Eseguire un algoritmo così potente richiede uno sforzo computazionale notevole: le reti neurali moderne sono composte da migliaia di neuroni per livello che richiedono milioni di pesi da ottimizzare precisamente. Proprio per questo motivo, con lo sviluppo di hardware avanzato, solo nell'ultimo decennio si è riusciti ad affrontare queste sfide computazionali [64], [66], [69].

3.2 RETI NEURALI CONVOLUZIONALI

Le reti neurali convoluzionali (CNN o ConvNet) sono state introdotte per la prima volta nel 1998, con la pubblicazione dell'articolo "Gradient-based learning applied to document recognition" [70] il quale descrive una rete artificiale, chiamata *LeNet-5*, il cui identificare documenti scritti a mano. Questo articolo getta le basi per la progettazione di una rete convoluzionale [71].

Una rete convoluzionale è una rete neurale feedforward che contiene dei particolari livelli, volti a ridurre il numero di pesi della rete. Tre sono i layer fondamentali che caratterizzano una CNN: *convolution layer*, *padding layer* e *ReLU layer*. Queste reti, sono organizzate in modo tale da riconoscere le caratteristiche spaziali di un immagine. Per questo motivo, generalmente, si rappresenta l'input layer come una matrice bidimensionale[3] di pixel: in questo modo viene anche facilitata la comprensione delle operazioni che processano l'input.

Nelle reti neurali convoluzionali, i parametri sono raggruppati in unità bidomensionali chiamati filtri o *kernel*. Questi sono delle matrici, generalmente quadrate, dimensionalmente minori rispetto alla grandezza dell'input. Tali unità rendono possibile l'operazione di convoluzione con il livello precedente. Come mostrato nella Figura 3.7, la convoluzione fa scorrere il filtro K sulla matrice M e, per ogni spostamento, calcola il prodotto scalare tra il kernel e i valori di M selezionati in quell'istante. Si osserva che il numero di valori risultanti da questa operazione è esattamente il numero di neuroni che sono presenti nel livello successivo. Questo valore si può calcolare in maniera semplice, conoscendo le dimensioni della matrice iniziale e del kernel. Fornite il numero di colonne della matrice, ovvero la sua larghezza w, e il numero di righe, ovvero la sua altezza h, allora, applicando una convoluzione con un filtro $n \times n$ le dimensioni al livello $\ell + 1$ valgono:

$$w^{(\ell+1)} = w^{(\ell)} - n + 1 \qquad\qquad h^{(\ell+1)} = h^{(\ell)} - n + 1$$

Osservando la Figura 3.7 si nota che la matrice M di dimensione 7×7 diventa una matrice 5×5 quando applicata una convoluzione con un filtro di dimensione 3×3. Si specifica inoltre che tra

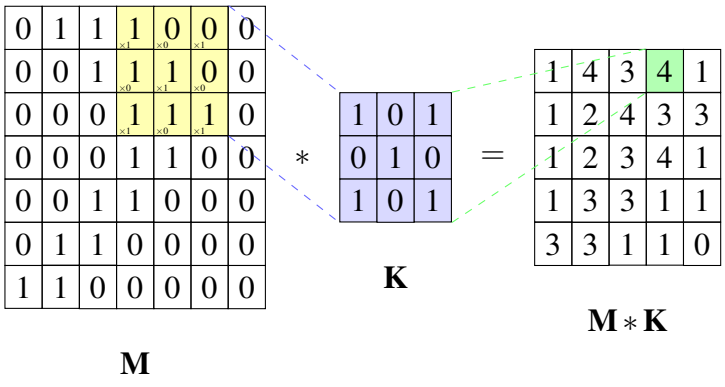

Figura 3.7: L'operazione di convoluzione in una matrice bidimensionale.

[3] Ipotizzando, per semplicità, che l'immagine sia in scala di grigi.

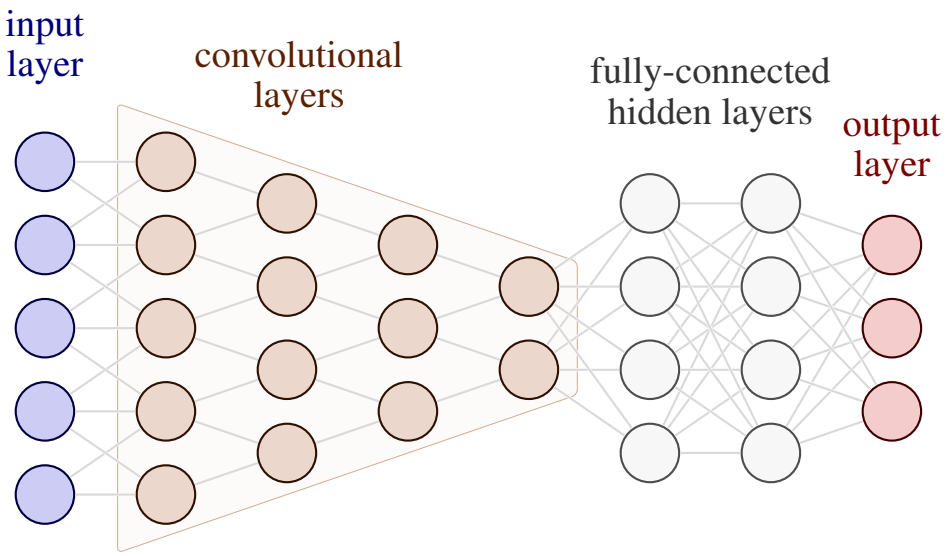

Figura 3.8: Rappresentazione di una rete neurale convoluzionale. Si fa notare la presenza di più livelli convoluzionali che rendono l'estrazione di pattern progressivamente più complessi.

un livello e quello successivo, non si esegue una sola operazione di convoluzione, bensì vengono compiute più convoluzioni, ciascuna con un filtro diverso (anche se della stessa dimensione). Di conseguenza, il risultato di queste operazioni sarà un numero di matrici d, che corrispondono al numero di filtri con cui è stata attuata la convoluzione sul livello precedente. L'applicazione di più filtri sullo stesso livello rende possibile riconoscere pattern particolari: per esempio è possibile utilizzare un insieme di filtri per riconoscere una sagoma all'interno di un'immagine. In questo modo, in una CNN è possibile avere diversi livelli convoluzionali tali per cui ciascuno riconosce un particolare pattern dall'immagine iniziale sempre più complesso. Avere più livelli convoluzionali che processano l'immagine fa si che l'ultimo livello convoluzionale esamini porzioni più ampie dell'immagine, riconoscendo caratteristiche e strutture più articolate [71], [72].

Come notato in precedenza, dopo ogni livello di convoluzione, la dimensione dell'immagine è ridotta in proporzione alla grandezza del filtro. In genere, la riduzione dimensionale comporta perdita di informazioni e va evitata. Per questo motivo, prima di fare l'operazione di convoluzione, si effettua il padding. Attraverso questa operazione, si aggiungono una serie di "0"[4] lungo i bordi dell'immagine in modo tale che, una volta processata, rimanga della stessa dimensione. In ogni bordo dell'immagine vengono aggiungi esattamente $(n-1)/2$ pixel, dove n è la dimensione del filtro. La Figura 3.9 illustra l'operazione di convuluzione presentata nella Figura 3.7 con la differnza che la matrice iniziale ha subito l'operazione di padding. Notando le due figure si osserva che nel primo caso il risultato della convoluzione ha ridotto la dimensione iniziale delle matrice mentre nel secondo caso, attraverso il padding, la dimensione rimane invariata grazie ai valori inseriti lungo i bordi della matrice [71].

In genere, ogni livello convoluzionale è seguito da un ReLU layer. Questo livello sostituisce

[4]Si è scelto il valore zero proprio perché non ha effetto e non distorce il risultato numerico della convoluzione.

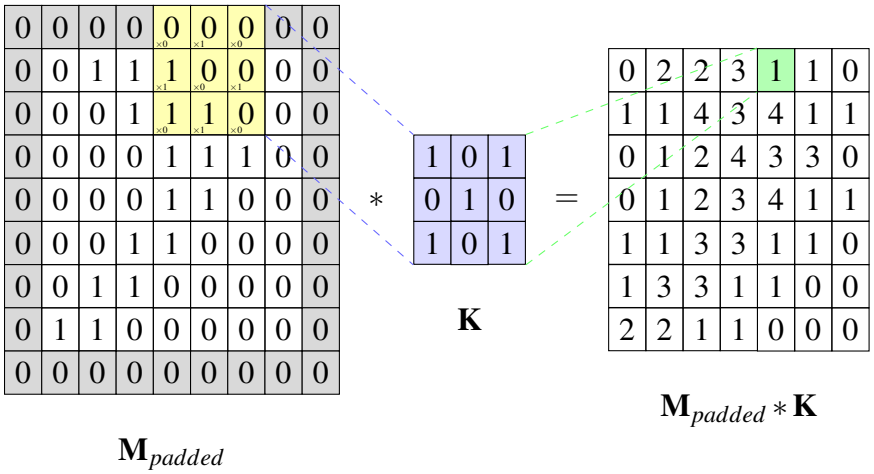

Figura 3.9: Padding di una matrice che precede la convoluzione. In questo modo, la dimensione della matrice rimane invariata dopo la convoluzione.

la funzione di attivazione sigmoidale con una funzione detta rettificatrice, definita come segue:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{se } v > 0 \\ 0 & \text{se } v \leq 0 \end{cases}$$

La funzione di attivazione ReLU, rappresentata nella Figura 3.10, è stata introdotta recentemente nelle reti neurali in quanto fornisce diversi vantaggi rispetto alle funzioni sigmoidali, soprattutto per quanto riguarda la velocità dell'allenamento[5] [71].

Dopo aver processato le infomazioni mediante la convoluzione, al fine di alleggerire il carico computazionale, si cerca di concentrare le informazioni estratte attraverso il *pooling*. Il tipo di pooling più comune nelle NN è il *max-pooling*. Questa operazione, in maniera simile ad un filtro, scorre lungo la matrice ed estrae dalla zona selezionata il valore massimo (Figura 3.11). L'idea dietro al max-pooling è quella di tenere le caratteristiche prevalenti dalla matrice fornita, in modo da ridurre l'informazione per rendere la fase di allenamento meno onerosa. Al max-

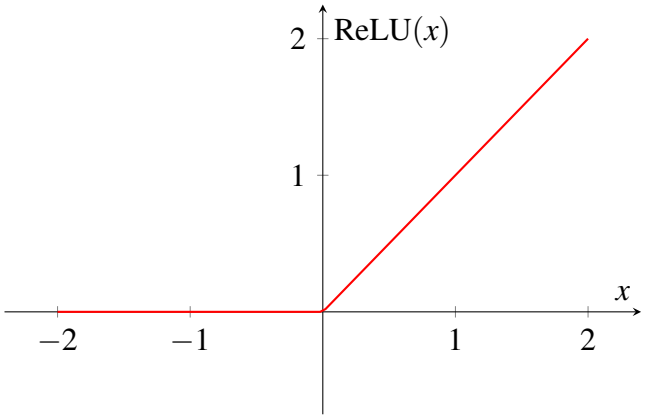

Figura 3.10: Grafico della funzione **ReLU**(x).

[5]Questo perché la derivata di una retta è molto più semplice rispetto alla derivata della funzione sigmoide.

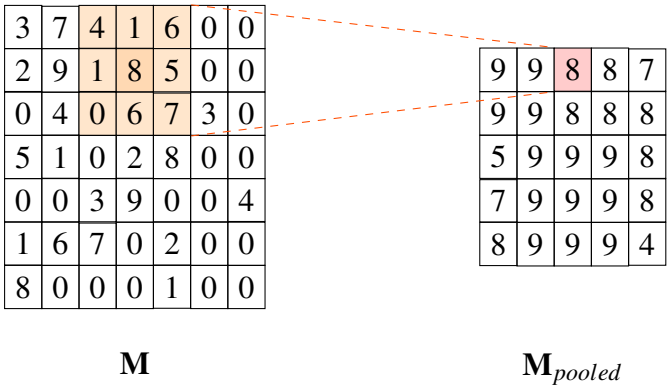

Figura 3.11: Operazione di max-pooling in una matrice.

pooling si aggiunge anche il *avg-pooling* che si occupa di calcolare la media dei pixel presenti nella porzione selezionata, estraendo quindi la "proprietà media" anzichè quella prevalente [71], [73].

Infine, come in tutte le reti neurali, in una ConvNet sono presenti i *fully-connected layers*. Questi livelli, che sono presenti solo dopo i livelli convoluzionali (come mostrato nella Figura 3.8), hanno la stessa funzione di una classica feedforward NN: effettuare una serie di calcoli che permette di predire correttamente la risposta, in funzione dell'input fornito [71], [73].

Anche se fino ad ora le CNN sono state descritte come tool che processa solo immagine, il loro utilizzo è in realtà molto più vasto. Tra le loro numerose applicazioni, vengono citate il *Natural Language Processing* (NLP) — disciplina che si occupa di processare il linguaggio naturale, attraverso il riconiscimento vocale e la classificazione del testo — e *Computer Vision* — che comprende il riconoscimento visivo, la *scene labelling* e il riconoscimento di oggetti e azioni umane [74]–[76]. Oltre a quanto menzionato, le reti neurali convoluzionali, sono largamente utilizzate nel campo della bioinformatica. Più precisamente, le CNN 1-dimensionali[6] analizzando sequenze di DNA ed RNA, apprendendo pattern — che in bioinformatica vengono anche chiamati *motif* — sequenziali man mano più complessi attraverso i diversi layer convoluzionali. In questo modo è possibile approfondire i problemi che riguardano gli elementi regolatori dell'espressione genica, tra cui le varianti non codificanti [77].

Le CNN durante l'allenamento modificano i pesi dei kernel

parla anche delle metodologie di evaluation

[6]Ovvero che l'input si estende in una dimensione, proprio come una sequenza.

4

Reti convoluzionali e varianti non codificanti

In questo capitolo verranno descritti tre tool basati sulle reti neurali convoluzionali, pubblicati con l'obiettivo di far più chiarezza sulla comprensione funzionale delle mutazioni non condificanti. I modelli in questione sono *DeepSEA* [16, "Predicting effects of noncoding variants with deep learning–based sequence model"], *Basset* [17, "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks"] e *DeepSATA* [18, "DeepSATA: A Deep Learning-Based Sequence Analyzer Incorporating the Transcription Factor Binding Affinity to Dissect the Effects of Non-Coding Genetic Variants"].

Di questi modelli si esploreranno importanti carattersitiche strutturali della rete, tra cui il numero e il tipo di layer presenti, il dataset iniziale utilizzato e

finisci introduzione

I picchi delle sequenze sono le regioni aperte del DNA

4.1 DeepSEA

Il primo tool che verrà analizzato è DeepSEA (*Deep learning-based Saquence Analizer*), introdotto nel 2015 con l'obiettivo di predire l'effetto delle varianti non codifcanti nella cromatina. Al fine di garantire una previsione accurata, questo modello considera lunghe sequenze di DNA in modo da conoscere in maniera migliore il contesto in cui avviene la mutazione e quindi comprenderne le funzionalità, anche grazie alla struttura gerarchica dei livelli convoluzionali, che permettono di esaminare pattern locali e globali. Inoltre, DeepSEA è in grado di imparare contemporaneamente diverse funzionalità della cromatina utilizzando un approccio di multitask learning, che permette di addestrare il modello su più compiti correlati nello stesso momento. Per comprendere in maniera più approfondita gli effetti funzionali delle mutazioni non codificanti, è stato allenato in modo da predire 919 profili di cromatina suddivisi in tre macro categorie:

- 690 tipi di sequenze associate ai siti di binding dei fattori di trascrizone (TF), che giocano un ruolo fondamentale durante la trascrizione;

- 125 profili di regioni ipersensibili all'enzima DNasi I (DHS), che indicano la presenza o meno di elementi regolativi nel DNA, anche questi importanti durante la fase di trascrizione;
- 104 profili di modifiche istoniche, ovvero mutazioni negli istoni che rendono poco accessibile il DNA nella cromatina, impedendo quindi una corretta trascrizione;

Il modello — implementato con la libreria *Torch7* — è composto da esattmente tre livelli convoluzionali: il primo livello contiene 320 filtri, il secondo 480 e l'ultimo contiene 960 kernel. I filtri sono delle *Position Weight Matrix* (PWM) che analizzano la sequenza in input e, attraverso la convoluzione, estraggono pattern significativi, man mano spostando la finestra di una base della sequenza. Nel primo livello il kernel ha dimesnione $M \times 4$, dove M è la lunghezza della finestra mentre 4 sono il numero delle basi azotate. I livelli convoluzionali successivi hanno invece dimensione $M \times k$, dove in questo caso k è il numero di kernel che sono stati utilizzati nel livello convoluzionale precedente. Dopo ogni livello convoluzionale viene applicata una ReLU (Figura 3.10) e poi viene inserito un max-pooling layer, volto ad estrarre la feature predominante dal risultato della convoluzione. In questo caso la finestra di pooling non ha uno step di uno, bensì lo step coincide con la lunghezza della finestra stessa. Infine, dopo i tre livelli convoluzionali, è presente un fully connected layer — che riceve il risultato del max-pooling layer della terza convoluzione e ci applica una ReLU — e l'output layer, che processa le informazioni con una funzione sigmoide (Figura 3.4) la quale calcola la probabilità che la sequenza in input corrisponda o meno a una tra i 919 profili.

I 919 profili di cromatina sono stati ottenuti dai progetti "*Encyclopedia of DNA Elements*" (ENCODE) e "*Roadmap Epigenomics*". Le sequenze estratte sono state divise in frammenti (*bin*) da 200bp[1], per un totale di 521 636 200bp. Ciascuno di questi bin è stato poi etichettato ad uno tra i 919 profili di cromatina se almeno più della metà della sequenza corrispondeva al profilo teorico. Ogni bin estratto è stato centrato su una sequenza di 1000bp di genoma umano, al fine di fornire a DeepSEA maggiore contesto per ricavare informazioni più significative. Infine,ciascuna sequenza di input è stata trasformata in una matrice 1000×4 attraverso il *One-Hot encoding* e associata al rispettivo vettore di *label* che contiene la risposta attesa dalla rete.

Per allenare DeepSEA in maniera efficace si è definita la cost function come la *Negative Log Likelihood* (NLL) — chiamata anche *Binary Cross Entropy* — definita come segue:

$$\mathbf{NLL} = -\sum_{s}\sum_{p} \log\left[y_{s,p}\,\sigma_p\left(X_s\right) + \left(1 - y_{s,p}\right)\left(1 - \sigma_p\left(X_s\right)\right)\right]$$

In questa formula, s è l'indice del campione s-esimo (X_s) del dataset e p è l'indice del profilo della cromatina. Ne consegue che il valore $y_{s,p}$ è il valore corretto del campione s rispetto al profilo di cromatina p mentre $\sigma_p\left(X_s\right)$ è la predizione di DeepSEA sul campione X_s rispetto al profilo p. L'effettiva cost function in realtà è definita come la funzione NLL sommata ad altri valori con l'obiettivo di evitare situazioni di *overfitting*[2]. Il gradiente della loss function è stato

[1] Con "bp", si intende *base pair*, ovvero la lunghezza della sequenza espressa nel numero di coppie di basi.

[2] L'overfitting è la situazione in cui il modello è completamente allineato con il dataset di allenamento, diventando quindi meno accurato nella predizioni di dati che non sono presenti nel datase.

calcolato attraverso l'algoritmo della backpropagation, per poi essere utilizzato nell'ottimizzazione della rete. L'algoritmo di ottimizzazione utilizzato è stato il SGD con momento, che è una variante utilizzata per aumentare ancora di più le possibilità di non cadere nei minimi locali [71]. Inoltre si è scelto di applicare il *dropout training* il quale implica di annullare l'effetto di alcuni neuroni durante le epoche dell'allenamento al fine di rendere la rete più robusta al rischio di overfitting [66].

L'efficacia predittiva del modello è stata valutata tramite l'AUC (*Area Under the Curve*), una metrica che misura la capacità del modello di distinguere tra i vari profili, con valori compresi tra 0 e 1. Il modello ha ottenuto un'AUC di 0.958 per i siti di binding dei fattori di trascrizione (TF), 0.923 per i siti DHS e 0.856 per le modifiche istoniche.

Utilizzando reti neurali convoluzionali profonde (CNN), Basset è in grado di apprendere in modo efficace i motivi sequenziali e la logica regolativa che determinano l'accessibilità del DNA in contesti cellulari specifici. Questo approccio consente di ottenere previsioni accurate riguardo a come le sequenze di DNA influenzano l'espressione genica.

4.2 BASSET

Basset[3] è un potente strumento sviluppato nel 2016, progettato per analizzare sequenze di DNA e prevedere l'accessibilità di 164 siti ipersensibili alla DNase I (DHS), che indicano la presenza di elementi regolativi. Mediante le ConvNet, questo modello è quindi in grado di fornire importanti informazioni rigaurdo la fase di trascrizione da DNA ad RNA anlizzando questi particolari siti della cromatina. Proprio come DeepSEA, la presenza di più livelli convoluzionali facilita la comprensione degli aspetti funzionali derivanti dalle mutazioni nei DHS.

Anche Basset, come DeepSEA, è stato implementato utilizzando la libreria *Torch7*. Questo tool dà la possibilità di personalizzare il modello a piacimento, indicando il numero per ciascun tipo di livello, il numero di filtri (PWM) nei livelli convoluzionali, la loro dimensione, la grandezza delle finestre di pooling, il numero di neuroni nei fully-connected layer e vari iperparametri per la fase di allenamento e di test. Attraverso l'ottimizzazione Bayesiana si sono trovati i layer e gli iperparametri ideali per l'architettura. La struttura è composta da tre livelli convoluzionali: il primo livello contiene 300 filtri di lunghezza 19bp, il secondo è composto da 200 kernel di lunghezza 11bp e il terzo layer ne contiene 200 di lunghezza 7bp. È importante sottolineare che dopo ciascun livello convoluzionale, segue un livello che normalizzi l'output della convoluzione (*batch normalization*) seguito da una ReLU ed un max-pooling layer. In seguito ai tre livelli convoluzionali sono presenti tre fully connected layer, alternati con due ReLU ed infine l'output layer, che attraverso la funzione sigmoide calcola la probabilità che l'input appartenga ad uno dei 164 DHS.

[3]Il nome richiama il bassotto, noto per le sue capacità olfattive, in analogia con l'abilità del modello di riconoscere pattern.

Dei siti ipersensibili alla DNasi I, 125 sono stati estratti da ENCODE e 39 "*Roadmap Epigenomics*". I dati estratti sono stati processati e tutti i siti sono stati isolati ed arricchiti fino ad avere un dataset iniziale di compostp da sequenze di 600bp, per un totale di 2 071 886bp. Ciascuna delle sequenze presenti nel dataset sono state poi associate al vettore di label che indicava a quale dei 164 tipi la sequenza apparteneva. Dei 164 tipi, il 17% dei siti sono stati associati a promoters, il 47% è stato classificato come siti intragenici — ovvero all'interno dei geni — e il restante 36% è stato etichettato come siti intergenici — cioè tra i geni. Infine, prima di essere utilizzati nella rete, i siti sono stati processati mediante il One-Hot encoding, formando quindi sequenze di input di dimensione 600×4. Del dataset totale, 71 886 basi sono state utilizzate per la fase di test e altre 70 000 per la fase di validazione.

Il modello è stato allenato attraverso il GD stocastico, cercando di ottimizzare la funzione binary cross entropy, il cui gradiente è stato calcolato mediante la backpropagation. Per prevenire il rischio di overfitting si è applicata la tecnica dell'*early stopping*, facendo terminare l'allenamento dopo 12 epoche che la *validation loss* rimane invariata. Dopo l'allenamento e la validazione, il modello è stato testato, ottenendo un valore AUC medio di 0.895 sui DHS.

4.3 DEEPSATA

Pubblicato nel 2023, DeepSATA è il terzo tool basato su una CNN che si occupa di prevedere l'effetto funzionale di mutazioni non codificanti nei siti di TF binding non solo in sequenze genomiche umane, ma anche di altre specie animali quali maiali, galline, bovini e topi.

DeepSATA è un modelo basato su DeepSEA. È quindi composto da tre layer convoluzionali, con rispettivamente da 320, 480 e 960 kernel. Ogni livello convluzionale è seguito da un layer che applica la funzione ReLU e poi un max pooling layer estrae dal risultato le feature predominanti. Infine è presente anche un dropout layer che, come per DeepSEA, aiuta la rete a prevenire il rischio di overfitting. Anche in questo tool, ai tre layer convoluzionali, seguono dei fully-conected layer che preparano i dati all'output layer che, attraverso la funzione sigmoide,

Dl-based sequence analizer che incorpora i TF binding sites for crossspecies prediction. Questo modello è cotroutio basandosi su DeepSEA. DeepSATA è stato procettato per identificare le regioni parte della cormatina (OCR) e le varianti non codificanti (functional annotation).

Per sequenze di DNA, DeepSATA estendo il concetto di one hot encoding in uno spazio tridimensionale che incorpara i TF bining sites che sono importanti nei contesti biologici specifci. Questi TF sono sono selezionati in modo da arricchire i motivi di DNA binding nelle OCR. Ogni layer bidimensionale codifica la binding affinity di uno specifico TF. L'encoding trategi utilizzata è quella della position weight probability matrix (PWPM) del TF site.

MODEL DESIGN

TRAINING DATASET

In maniera del tutto analoga a DeepSEA. Specifically, the genome was divided into bins of 200 base pairs (bp). Each bin was assigned a label of 1 if more than half of the 200 bp fell within the ATAC peak regions and a label of 0 if not. Anche qui ci sono 400 flaning sequnces con deepsea.

5 Discussione

Modello	Struttura	Codifica	Kernel	Dataset	Funzione costo
DeepSEA		One-hot Encoding	PWM		*Binary Cross Entropy*
Basset		One-hot Encoding	PWM		*Binary Cross Entropy*
DeepSATA		One-hot Encoding			

Modello	Topi	Maiali	Bovini	Umani	Polli
DeepSEA	0.796	0.775	0.769	0.755	0.736
Basset	0.778	0.719	0.768	0.717	0.722
DeepSATA	0.854	0.779	0.772	0.759	0.744

Tabella che specifica e riassume per ogni tool encoding, dataset etc

riassume quanto analizzato prima, riporta i risultati del paper piu recente in modo da avere un momento in cui riassumo la situa

Sperimentalmente, per risorse a disposizione, per il confronto ci si basa sui risultati dell'ultimo paper

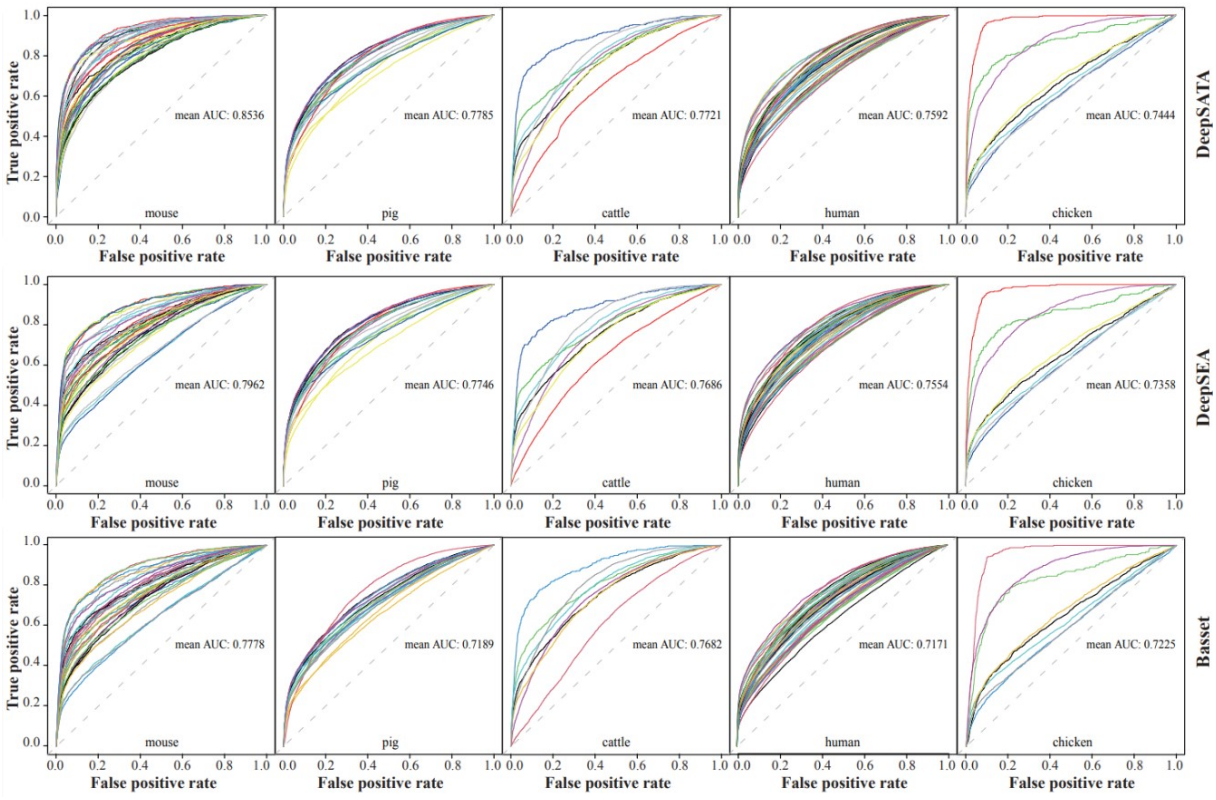

Figura 5.1: Confronto delle prestazioni predittive dei tre tool su diverse specie animali.

6

Conclusioni

Limiti, punti di forza e future ricerche. Se ci sono in ballo futuri lavori, nuove direzioni da seguire, da osservare dagli articoli dei 3 tool

Bibliografia

[1] S. S. Sahu e G. Panda, «Identification of protein-coding regions in DNA sequences using a time-frequency filtering approach,» *Genomics, Proteomics and Bioinformatics*, vol. 9, n. 1-2, pp. 45–55, 2011.

[2] T. D. Pollard, W. C. Earnshaw, J. Lippincott-Schwartz e G. Johnson, *Cell Biology E-Book: Cell Biology E-Book*. Elsevier Health Sciences, 2022.

[3] F. Zhang e J. R. Lupski, «Non-coding genetic variants in human disease,» *Human molecular genetics*, vol. 24, n. R1, R102–R110, 2015.

[4] J. French e S. Edwards, «The role of noncoding variants in heritable disease,» *Trends in Genetics*, vol. 36, n. 11, pp. 880–891, 2020.

[5] H. Chial, «Mendelian genetics: patterns of inheritance and single-gene disorders,» *Nature Education*, vol. 1, n. 1, p. 63, 2008.

[6] S. Pagni, J. D. Mills, A. Frankish, J. M. Mudge e S. M. Sisodiya, «Non-coding regulatory elements: Potential roles in disease and the case of epilepsy,» *Neuropathology and Applied Neurobiology*, vol. 48, n. 3, e12775, 2022.

[7] A. Kapoor et al., «An enhancer polymorphism at the cardiomyocyte intercalated disc protein NOS1AP locus is a major regulator of the QT interval,» *The American Journal of Human Genetics*, vol. 94, n. 6, pp. 854–869, 2014.

[8] E. Khurana, Y. Fu, D. Chakravarty, F. Demichelis, M. A. Rubin e M. Gerstein, «Role of non-coding sequence variants in cancer,» *Nature Reviews Genetics*, vol. 17, n. 2, pp. 93–108, 2016.

[9] J. Tian et al., «Systematic functional interrogation of genes in GWAS loci identified ATF1 as a key driver in colorectal cancer modulated by a promoter-enhancer interaction,» *The American Journal of Human Genetics*, vol. 105, n. 1, pp. 29–47, 2019.

[10] S. E. Bojesen et al., «Multiple independent variants at the TERT locus are associated with telomere length and risks of breast and ovarian cancer,» *Nature genetics*, vol. 45, n. 4, pp. 371–384, 2013.

[11] K. Michailidou et al., «Association analysis identifies 65 new breast cancer risk loci,» *Nature*, vol. 551, n. 7678, pp. 92–94, 2017.

[12] C. S. Pareek, R. Smoczynski e A. Tretyn, «Sequencing technologies and genome sequencing,» *Journal of applied genetics*, vol. 52, pp. 413–435, 2011.

[13] S. K. Burley, H. M. Berman, G. J. Kleywegt, J. L. Markley, H. Nakamura e S. Velankar, «Protein Data Bank (PDB): the single global macromolecular structure archive,» *Protein crystallography: methods and protocols*, pp. 627–641, 2017.

[14] N. M. Luscombe, D. Greenbaum e M. Gerstein, «What is bioinformatics? A proposed definition and overview of the field,» *Methods of information in medicine*, vol. 40, n. 04, pp. 346–358, 2001.

[15] C. Caudai et al., «AI applications in functional genomics,» *Computational and Structural Biotechnology Journal*, vol. 19, pp. 5762–5790, 2021.

[16] J. Zhou e O. G. Troyanskaya, «Predicting effects of noncoding variants with deep learning–based sequence model,» *Nature methods*, vol. 12, n. 10, pp. 931–934, 2015.

[17] D. R. Kelley, J. Snoek e J. L. Rinn, «Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks,» *Genome research*, vol. 26, n. 7, pp. 990–999, 2016.

[18] W. Ma et al., «DeepSATA: A Deep Learning-Based Sequence Analyzer Incorporating the Transcription Factor Binding Affinity to Dissect the Effects of Non-Coding Genetic Variants,» *International Journal of Molecular Sciences*, vol. 24, n. 15, p. 12 023, 2023.

[19] B. Alberts et al., *Essential cell biology*. Garland Science, 2015.

[20] P. F. Chinnery e E. A. Schon, «Mitochondria,» *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 74, n. 9, pp. 1188–1199, 2003.

[21] H. M. McBride, M. Neuspiel e S. Wasiak, «Mitochondria: more than just a powerhouse,» *Current biology*, vol. 16, n. 14, R551–R560, 2006.

[22] G. K. Voeltz, M. M. Rolls e T. A. Rapoport, «Structural organization of the endoplasmic reticulum,» *EMBO reports*, vol. 3, n. 10, pp. 944–950, 2002.

[23] A. Ballabio, «The awesome lysosome,» *EMBO molecular medicine*, vol. 8, n. 2, pp. 73–76, 2016.

[24] C. Yang e X. Wang, «Lysosome biogenesis: Regulation and functions,» *The Journal of cell biology*, vol. 220, n. 6, 2021.

[25] E. C. Dell'Angelica, C. Mullins, S. Caplan e J. S. Bonifacino, «Lysosome-related organelles,» *The FASEB Journal*, vol. 14, n. 10, pp. 1265–1278, 2000.

[26] M. Islinger, S. Grille, H. D. Fahimi e M. Schrader, «The peroxisome: an update on mysteries,» *Histochemistry and cell biology*, vol. 137, pp. 547–574, 2012.

[27] National Human Genome Research Institute, *Deoxyribonucleic acid (DNA) Image*, `https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid`, 2024.

[28] C. Fonseca Guerra, F. M. Bickelhaupt, J. G. Snijders e E. J. Baerends, «Hydrogen bonding in DNA base pairs: reconciliation of theory and experiment,» *Journal of the American Chemical Society*, vol. 122, n. 17, pp. 4117–4128, 2000.

[29] A. Jansen e K. J. Verstrepen, «Nucleosome positioning in Saccharomyces cerevisiae,» *Microbiology and molecular biology reviews*, vol. 75, n. 2, pp. 301–320, 2011.

[30] G. Zheng, *The packaging of DNA in chromatin*. Rutgers The State University of New Jersey, School of Graduate Studies, 2010.

[31] National Human Genome Research Institute, *Chromosome Image*, `https://www.genome.gov/genetics-glossary/Chromosome`, 2024.

[32] M. B. Gerstein et al., «What is a gene, post-ENCODE? History and updated definition,» *Genome research*, vol. 17, n. 6, pp. 669–681, 2007.

[33] R. J. White, *Gene transcription: mechanisms and control*. John Wiley & Sons, 2009.

[34] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts e P. Walter, «From DNA to RNA,» in *Molecular Biology of the Cell. 4th edition*, Garland Science, 2002.

[35] P. Cramer, «Organization and regulation of gene transcription,» *Nature*, vol. 573, n. 7772, pp. 45–54, 2019.

[36] National Human Genome Research Institute, *Transcription Image*, `https://www.genome.gov/genetics-glossary/Transcription`, 2024.

[37] A. Philips e T. Cooper*, «RNA processing and human disease,» *Cellular and Molecular Life Sciences CMLS*, vol. 57, pp. 235–249, 2000.

[38] S. Hocine, R. H. Singer e D. Grünwald, «RNA processing and export,» *Cold Spring Harbor perspectives in biology*, vol. 2, n. 12, a000752, 2010.

[39] M. Livingstone, E. Atas, A. Meller e N. Sonenberg, «Mechanisms governing the control of mRNA translation,» *Physical biology*, vol. 7, n. 2, p. 021 001, 2010.

[40] V. Ramakrishnan, «Ribosome structure and the mechanism of translation,» *Cell*, vol. 108, n. 4, pp. 557–572, 2002.

[41] J. Lemonnier, N. Lemonnier, S. Pascolo e C. Pichon, «The Marathon of the Messenger,»

[42] National Human Genome Research Institute, *Translation Image*, `https://www.genome.gov/genetics-glossary/Translation`, 2024.

[43] G. E. Schulz e R. H. Schirmer, *Principles of protein structure*. Springer Science & Business Media, 2013.

[44] R. M. Bavle, «Mitosis at a glance,» *Journal of Oral and Maxillofacial Pathology*, vol. 18, n. Suppl 1, S2–S5, 2014.

[45] C. E. Walczak, S. Cai e A. Khodjakov, «Mechanisms of chromosome behaviour during mitosis,» *Nature reviews Molecular cell biology*, vol. 11, n. 2, pp. 91–102, 2010.

[46] X. Li, F. Yang e B. Rubinsky, «A theoretical study on the biophysical mechanisms by which tumor treating fields affect tumor cells during mitosis,» *IEEE Transactions on Biomedical Engineering*, vol. 67, n. 9, pp. 2594–2602, 2020.

[47] M. Sullivan e D. O. Morgan, «Finishing mitosis, one step at a time,» *Nature reviews Molecular cell biology*, vol. 8, n. 11, pp. 894–903, 2007.

[48] National Human Genome Research Institute, *Mitosis Image*, https://www.genome.gov/genetics-glossary/Mitosis, 2024.

[49] R. A. Laskey, M. P. Fairman e J. J. Blow, «S phase of the cell cycle,» *Science*, vol. 246, n. 4930, pp. 609–614, 1989.

[50] S. P. Bell e A. Dutta, «DNA replication in eukaryotic cells,» *Annual review of biochemistry*, vol. 71, n. 1, pp. 333–374, 2002.

[51] A. Dutta e S. P. Bell, «Initiation of DNA replication in eukaryotic cells,» *Annual review of cell and developmental biology*, vol. 13, n. 1, pp. 293–332, 1997.

[52] «Chapter 42 - S Phase and DNA Replication,» in *Cell Biology (Third Edition)*, T. D. Pollard, W. C. Earnshaw, J. Lippincott-Schwartz e G. T. Johnson, cur., Third Edition, Elsevier, 2017, pp. 727–741, ISBN: 978-0-323-34126-4. DOI: https://doi.org/10.1016/B978-0-323-34126-4.00042-6. indirizzo: https://www.sciencedirect.com/science/article/pii/B9780323341264000426.

[53] National Human Genome Research Institute, *DNA Replication Image*, https://www.genome.gov/genetics-glossary/DNA-Replication, 2024.

[54] P. M. Visscher, M. A. Brown, M. I. McCarthy e J. Yang, «Five years of GWAS discovery,» *The American Journal of Human Genetics*, vol. 90, n. 1, pp. 7–24, 2012.

[55] M. Z. Ludwig, «Functional evolution of noncoding DNA,» *Current opinion in genetics & development*, vol. 12, n. 6, pp. 634–639, 2002.

[56] V. B. Kaiser e C. A. Semple, «When TADs go bad: chromatin structure and nuclear organisation in human disease,» *F1000Research*, vol. 6, 2017.

[57] M. Schipper e D. Posthuma, «Demystifying non-coding GWAS variants: an overview of computational tools and methods,» *Human molecular genetics*, vol. 31, n. R1, R73–R83, 2022.

[58] E. G. Peña-Martínez e J. A. Rodríguez-Martínez, «Decoding Non-coding Variants: Recent Approaches to Studying Their Role in Gene Regulation and Human Diseases,» *Frontiers in bioscience (Scholar edition)*, vol. 16, n. 1, p. 4, 2024.

[59] W. S. McCulloch e W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[60] A. M. Turing, *Computing machinery and intelligence*. Springer, 2009.

[61] F. Rosenblatt, «The perceptron: a probabilistic model for information storage and organization in the brain.,» *Psychological review*, vol. 65, n. 6, p. 386, 1958.

[62] M. Minsky e S. A. Papert, *Perceptrons, reissue of the 1988 expanded edition with a new foreword by Léon Bottou: an introduction to computational geometry*. MIT press, 2017.

[63] D. E. Rumelhart, G. E. Hinton e R. J. Williams, «Learning representations by back-propagating errors,» *nature*, vol. 323, n. 6088, pp. 533–536, 1986.

[64] M. Flasiński, *Introduction to artificial intelligence*. Springer, 2016.

[65] N. Muthukrishnan, F. Maleki, K. Ovens, C. Reinhold, B. Forghani, R. Forghani et al., «Brief history of artificial intelligence,» *Neuroimaging Clinics of North America*, vol. 30, n. 4, pp. 393–399, 2020.

[66] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015, vol. 25.

[67] J. Lu, «Gradient descent, stochastic optimization, and other tales,» *arXiv preprint arXiv:2205.00832*, 2022.

[68] M. Andrychowicz et al., «Learning to learn by gradient descent by gradient descent,» *Advances in neural information processing systems*, vol. 29, 2016.

[69] R. Rojas e R. Rojas, «The backpropagation algorithm,» *Neural networks: a systematic introduction*, pp. 149–182, 1996.

[70] Y. LeCun, L. Bottou, Y. Bengio e P. Haffner, «Gradient-based learning applied to document recognition,» *Proceedings of the IEEE*, vol. 86, n. 11, pp. 2278–2324, 1998.

[71] C. C. Aggarwal et al., *Neural networks and deep learning*. Springer, 2018, vol. 10.

[72] J. Wu, «Introduction to convolutional neural networks,» *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, n. 23, p. 495, 2017.

[73] K. O'Shea, «An introduction to convolutional neural networks,» *arXiv preprint arXiv:1511.08458*, 2015.

[74] A. S. Shamsaldin, P. Fattah, T. A. Rashid e N. K. Al-Salihi, «A study of the applications of convolutional neural networks,» *J. Sci. Eng*, vol. 3, pp. 31–39, 2019.

[75] Z. Li, F. Liu, W. Yang, S. Peng e J. Zhou, «A survey of convolutional neural networks: analysis, applications, and prospects,» *IEEE transactions on neural networks and learning systems*, vol. 33, n. 12, pp. 6999–7019, 2021.

[76] D. Bhatt et al., «CNN variants for computer vision: History, architecture, application, challenges and future scope,» *Electronics*, vol. 10, n. 20, p. 2470, 2021.

[77] S. Min, B. Lee e S. Yoon, «Deep learning in bioinformatics,» *Briefings in bioinformatics*, vol. 18, n. 5, pp. 851–869, 2017.

Ringraziamenti