UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL' INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

RETI CONVOLUZIONALI PER LA CLASSIFICAZIONE DI ENZIMI

Relatore

Prof. Loris Nanni

Laureando

Giacomo Valentini

Anno Accademico 2021/2022

Padova, 14 novembre 2022

Introduzione

Nell'ambito dell'Intelligenza Artificiale, in particolare del Deep Learning, le reti neurali convoluzionali (CNN) sono una tipologia di reti neurali artificiali (ANN) feed-forward che hanno avuto crescente successo negli ultimi anni, grazie alle elevate performance dimostrate nella risoluzione di problemi di Computer Vision. Con le più tradizionali reti Multi-Layer Perceptron (MLP), caratterizzate principalmente da strati completamente connessi, si incappa facilmente nel problema dell'overfitting, fenomeno per il quale la rete neurale si adatta al training set e riporta performance non ottimali nella fase di validation. Per ovviare a questo problema le MLP ricorrono a tecniche di regolarizzazione per penalizzare la complessità del modello. Le reti di tipo CNN adottano invece una strategia diversa, che consiste nell'utilizzo di filtri (o kernel) per il downsampling dei dati in input. Tipicamente gli strati nascosti di una rete neurale convoluzionale possono essere, appunto, convoluzionali, di pooling, fully-connected, e altre tipologie di layer atte alla semplificazione del modello e di conseguenza alla prevenzione dell'overfitting. Contemporaneamente allo sviluppo delle tecnologie per il Deep Learning le dimensioni dell'archivio digitale Protein Data Bank [1] (PDB) sono aumentate esponenzialmente. Il sito https://www.wwpdb.org riporta un sostanziale incremento di deposizioni e download annui: quest'ultimo dato passa da 328 362 536 nel 2009, a 2 364 150 827 nel 2021. Ciò ha permesso la diffusione di modelli che cercano di predire l'Enzyme Commission number (EC number), ovvero il tipo di reazione chimica catalizzata, tra Ossidoreduttasi (EC1), Transferasi (EC2), Idrolasi (EC3), Liasi (EC4), Isomerasi (EC5) e Ligasi (EC6), partendo dalla composizione chimica e struttura del dato enzima. Questo elaborato di tesi propone un ensemble per la classificazione dell'EC number basato sulla fusione di score provenienti dall'addestramento di due reti CNN e un modello SVM.

Indice

Elenco delle figure

Elenco delle tabelle

1 Reti Neurali Convoluzionali

Questo capitolo introduce il concetto di Rete Neurale Convoluzionale, in inglese Convolutional Neural Network (CNN), nell'ambito del Deep Learning e i vantaggi offerti da questa tipologia di reti neurali in termini di performance, quando applicate a problemi di Computer Vision.

1.1 Reti neurali artificiali

Per poter parlare di Reti Neurali Convoluzionali è d'obbligo introdurre, seppur brevemente, i concetti di Neurone Artificiale e Rete Neurale Artificiale (ANN), in quanto fondamenti della branca dell'Intelligenza Artificiale che si occupa della risoluzione di problemi prendendo ispirazione dalla controparte biologica, il cervello umano.

1.1.1 Neuroni Biologici

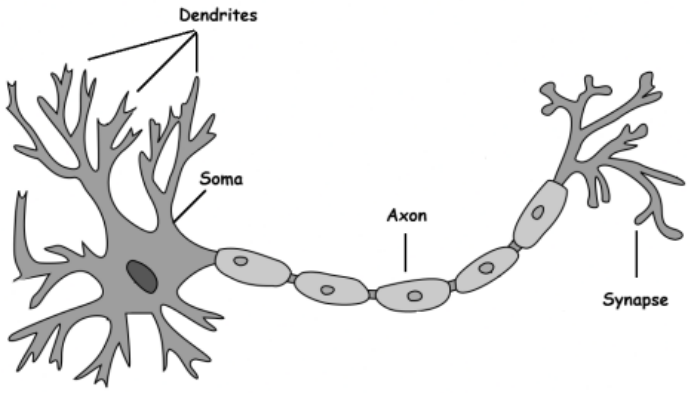


Figura 1.1: Neurone biologico

In biologia la cellula neuronale (fig 1.1) costituisce l'unità fondamentale del tessuto nervoso ed è in grado di ricevere, elaborare e trasmettere impulsi nervosi. L'anatomia di queste cellule, in una conformazione tipica, presenta le seguenti componenti:

- il **soma**, il corpo principale del neurone, adibito all'elaborazione dei segnali recepiti.
- i **dendriti**, fibre minori che si ramificano dal soma, costituiscono le linee di input del neurone.
- **l'assone**, ramificazione principale collegata alle estremità con il soma e i terminali assiomatici, le linee di output del neurone.

1.1.2 Neuroni artificiali

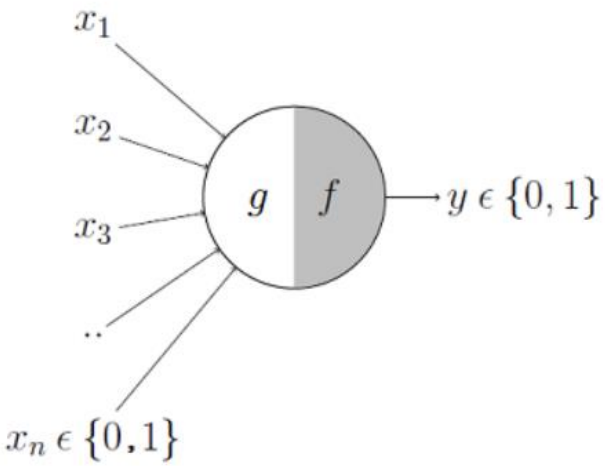


Figura 1.2: Neurone artificiale

Nell'intelligenza artificiale il neurone è una funzione matematica che svolge un compito analogo a quello dei neuroni biologici: di ricezione, elaborazione e trasmissione del segnale all'interno della rete neurale. Il primo modello di neurone artificiale (fig 1.2), introdotto da Warren McCulloch e Walter Pitts nel 1943 [4], prevede una serie di input $x_i \in \{0, 1\}$ e pesi$w_i \in -1, +1$. Lo stato di eccitazione g del neurone viene calcolato con la formula $g = \sum w_i x_i$, successivamente l'output assume il valore $y = f(g)$con f funzione a gradino e $y \in 0,1$.

1.1.3 Percettroni

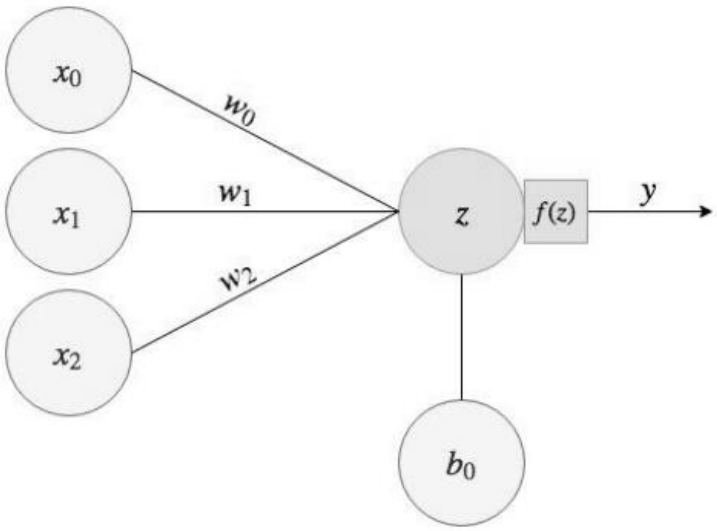


Figura 1.3: Percettrone

Il concetto di percettrone (fig 1.3), sviluppato da Frank Rosenblatt, costituisce un miglioramento in termini di flessibilità rispetto al neurone artificiale, in quanto i valori di input, pesi e output non sono più ristretti al dominio binario. Lo stato di eccitazione z del percettrone viene calcolato con la formula $z = b + \sum w_i x_i$, con b valore di bias. L'output assume il valore $y = f(g)$, con f funzione di trasferimento tipicamente a gradino o sigmoide.

1.1.4 Reti Neurali e Multilayer Perceptron

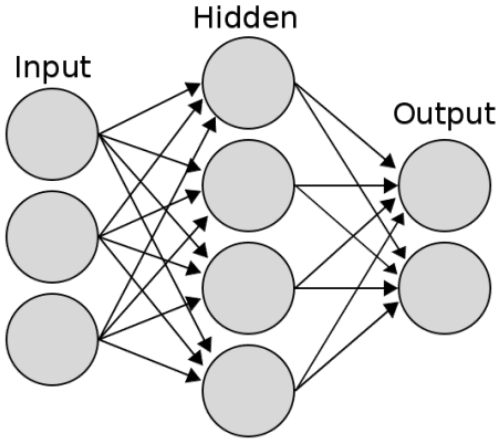


Figura 1.4: ANN a 3 strati fully-connected

Le Reti Neurali Artificiali (ANN) sono sistemi computazionali formati dall'organizzazione di più neuroni in strati, o layers, successivi (fig 1.4). Il primo e l'ultimo layer costituiscono rispettivamente input e output della rete neurale; in mezzo sono

solitamente presenti uno o più strati che prendono il nome di hidden layers. Nelle reti di tipo feed-forward, le più comuni, i collegamenti neuronali di ogni strato interessano solo i neuroni dello strato successivo. Il Multilayer Perceptron (MLP) è una rete feed-forward con almeno 3 livelli e funzioni di attivazione non lineari. Il teorema *Universal approximation theorem* [5] dimostra che una MLP con un solo hidden layer è in grado di approssimare qualsiasi funzione continua che mappa intervalli di numeri reali su un intervallo di numeri reali.

1.1.5 Addestramento

L'addestramento di una rete neurale consiste nel determinare i valori ottimali dei pesi $w_{i,j}$ per mappare correttamente l'input all'output. Nel caso di un classificatore l'addestramento ha lo scopo di garantire la massima accuratezza nella classificazione dei pattern forniti in input alla rete.

I concetti chiave dell'addestramento sono gli step di forward e backward propagation per l'aggiornamento dei pesi:

- **forward-propagation** è la propagazione delle informazioni in avanti, dall'input all'output. Prendendo come esempio una MLP a 3 livelli (x, y, z) fully-connected, l'informazione si propaga al nodo z_k di output secondo il modello:

$$\boldsymbol{z_k = f\left(\sum_{j=1} w_{jk} y_j + w_{0k}\right) = f\left(\sum_{j=1} w_{jk} f\left(\sum_{i=1} w_{ij} x_i + w_{0j}\right) + w_{0k}\right)}$$

(1.1)

- **backward-propagation** è il processo di aggiornamento dei pesi contro gradiente, dall'output all'input. Sempre con riferimento alla rete MLP a 3 livelli consideriamo $z = [z_1, z_2, \dots, z_n]$ l'output prodotto dallo step di forward-propagation in corrispondenza dei pattern $x = [x_1, x_2, \dots, x_n]$. Il target della rete è $t = [t_1, t_2, \dots, t_n]$. La funzione obiettivo, o loss function, rappresenta lo scarto tra la predizione della rete e l'output desiderato:

$$J(\mathrm{w}, \mathrm{x}) = \frac{1}{2} \sum_{1=1\dots\mathrm{n}} (t_i - z_i)^2$$

(1.2)

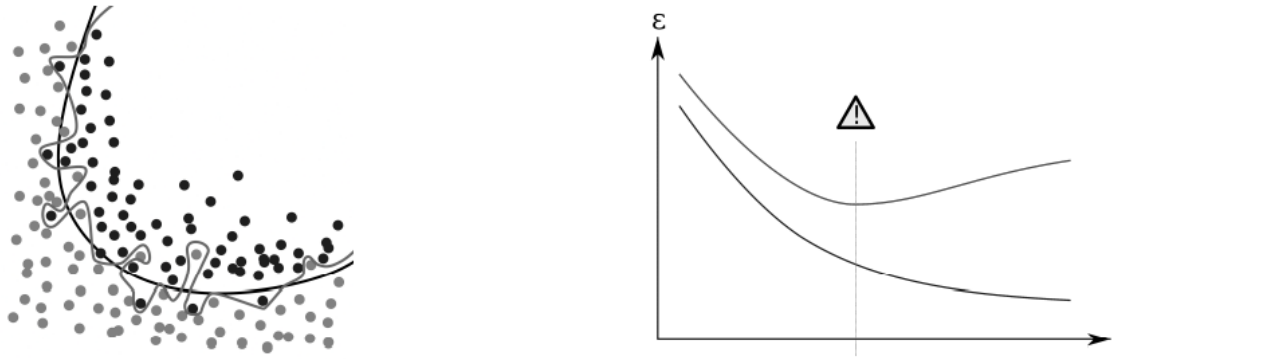
La backpropagation si occupa dunque di modificare i pesi w in direzione opposta al gradiente di J. La *Stochastic Gradient Descent* (SGD) è il metodo più comunemente utilizzato per implementare la backpropagation.

Gli step di forward e backward propagation si ripetono per ogni pattern del training set, per ogni epoca di addestramento.

1.2 CNN per il Deep Learning

Questa sezione si occupa di riportare la differenza di approccio alla risoluzione di problemi tra le tradizionali reti MLP e le reti CNN, e i vantaggi dimostrate da queste ultime nell'ambito del Deep Learning, in particolare per la Computer Vision.

1.2.1 Overfitting



(a) Differenza tra modello overfittato e regolarizzato

(b) Andamento della loss function $\mathcal{E}$, train e validation

Figura 1.5: Rappresentazione dell'overfitting

Il problema dell'overfitting (fig 1.5) indica l'eccessivo adattamento della rete neurale ai dati del training set. Si incorre spesso in questo fenomeno in presenza di reti MLP che non adottano tecniche di regolarizzazione, ovvero di semplificazione del modello matematico. Le soluzioni più comuni a questo problema coinvolgono l'aggiunta di meccanismi di penalizzazione o vincoli, oppure il taglio (dropout) di alcune sinapsi tra i layer, per la semplificazione della rete.

1.2.2 La struttura delle CNN

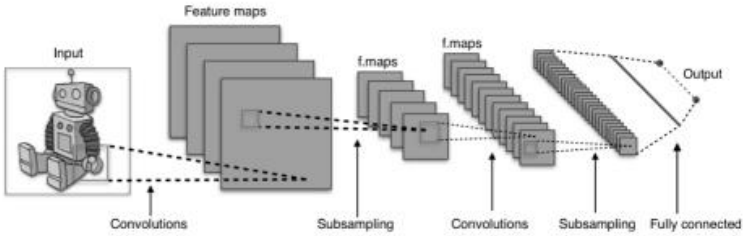


Figura 1.6: Struttura tipica di una CNN

Le Reti Neurali Convoluzionali (CNN) sono variazioni delle reti MLP progettate per minimizzare la pre-elaborazione dei dati. Questa caratteristica le rende particolarmente apprezzate per Computer Vision, vista l'elevata complessità dell'estrazione manuale di un numero ristretto di feature significative a partire da immagini. L'approccio adottato dalle reti CNN per la semplificazione del modello e la risoluzione del problema dell'overfitting prende la forma di filtri (o kernel) multidimensionali, che rappresentano la topologia delle sinapsi tra layer ed operano downsampling dei dati di input. L'inferiore complessità sinaptica di questi filtri permette di spingersi più profondità (fig 1.6) rispetto a reti esclusivamente fully-connected, come le tradizionali MLP.

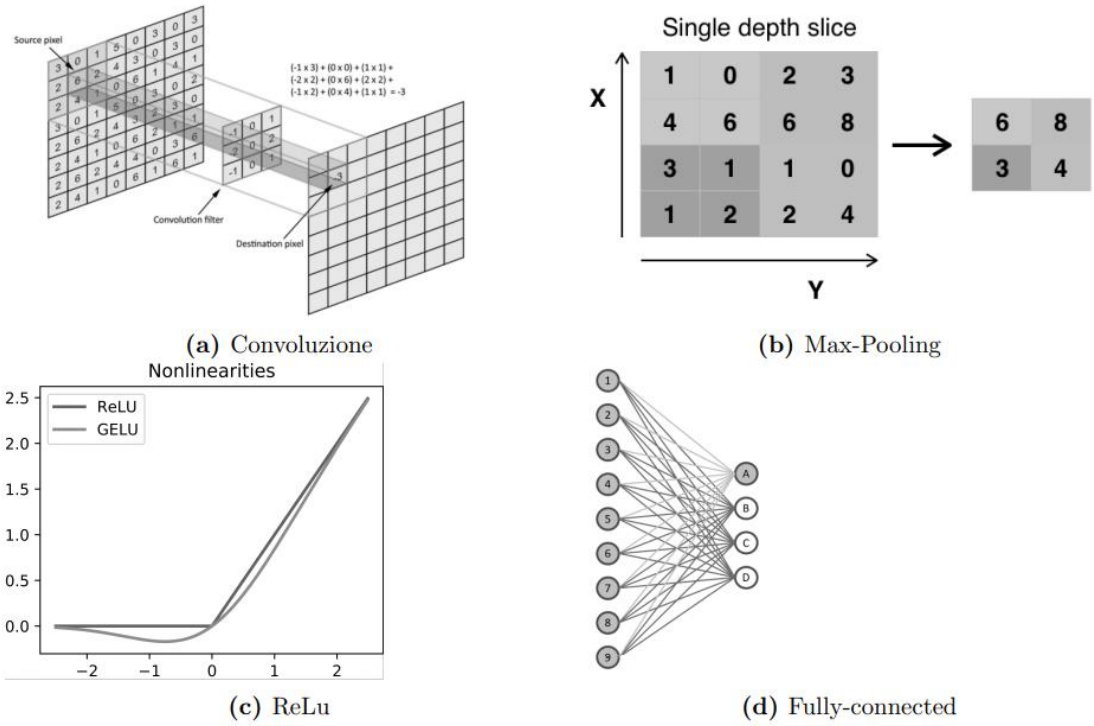


Figura 1.7: Principali tipologie di layer presenti nelle CNN

I principali "blocchi" che compongono le CNN sono dunque (fig 1.7):

- Layer **Convoluzionali**, che mappano ogni neurone ad una porzione del layer precedente. Il valore in output si calcola con il prodotto scalare tra la finestra di input e il kernel utilizzato secondo $net = \sum_i w_i in_i + b$. La grandezza di ogni dimensione del layer di output si calcola con:

$$W_o = \frac{W_i - F + 2P}{S} + 1$$

(1.3)

Wi dimensione dell'input, F dimensione del filtro, P spessore (eventuale) bordo ed S passo di spostamento del filtro (stride).

- Layer **Pooling**, che assegnano ad ogni neurone un valore all'interno di una finestra del layer precedente, secondo una funzione di decisione. Nel caso di max-pooling con un filtro 2x2 e passo pari a 2 la funzione risulta:

$$f_{X,Y}(S) = \max_{a,b=0} S_{2X+a,2Y+b}$$

(1.4)

La formula per il calcolo delle dimensioni del layer di output non varia da quella mostrata per i layer convoluzionali.

- Layer **ReLu**, che usano il rettificatore, o funzione rampa $f(x) = max(0, x)$, come funzione di attivazione per sostituire i valori negativi con 0. Varianti della funzione ReLu risolvono il potenziale problema della scomparsa del gradiente, che sorge quando grandi porzioni della rete sono desensibilizzate agli input e "muoiono", in quanto il gradiente ha valori solo nell'intervallo $(0, +\infty)$.
- Layer **Fully-Connected**, che presentano neuroni con sinapsi che li collegano a tutti i neuroni dello strato precedente. Nelle reti CNN un layer fully-connected prende la posizione finale, dopo vari layer convoluzionali e/o pooling.

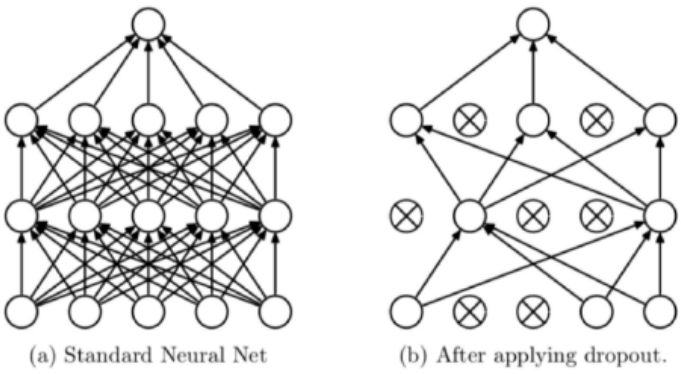


Figura 1.8: Dropout

Tra le tecniche di regolarizzazione adottate nelle reti CNN è importante ricordare il Dropout [6], che permette di ignorare ogni nodo con probabilità p per ovviare al problema dell'overfitting del layer fully-connected.

2 Support Vector Machine

Nell'ambito dell'Intelligenza Artificiale, in particolare del Machine Learning, le Support Vector Machine (SVM) sono modelli di apprendimento supervisionato per la determinazione di superfici decisionali tra le classi. I modelli SVM massimizzano il margine tra la superficie individuata e i pattern di classi differenti: maggiore è questo margine nella fase di training, migliore è tendenzialmente la gestione dei pattern nella fase di test.

2.1 Tipologie e Kernel

2.1.1 SVM lineari

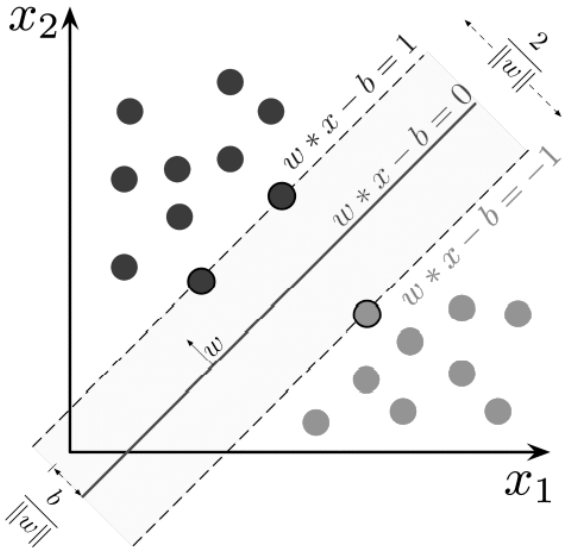


Figura 2.1: SVM lineare

Dato un dataset contenente n campioni $(x_1, y_1), \dots, (x_n, y_n)$, dove x_i sono i pattern multidimensionali e $y_i \in \{+1, -1\}$ le label ad essi associati, il modello SVM lineare (fig 2.1) trova l'iperpiano che divide i pattern appartenenti alla classe $+1$ da quelli appartenenti alla classe -1, massimizzando la distanza da entrambi i gruppi.

La formula che esprime un iperpiano è:

$$\mathrm{D(x)} = \mathrm{w^T} \cdot \mathrm{x} + \mathrm{b} \tag{2.1}$$

dove w è il vettore normale all'iperpiano, $\frac{b}{||w||}$ esprime la distanza di tale iperpiano dall'origine lungo il vettore w, e $D(x) = 0$ è il luogo dei vettori sul piano.
Si distinguono principalmente due casi:

- Se i pattern sono **linearmente separabili** si possono selezionare due iperpiani paralleli a distanza $\frac{1}{||w||}$ da ambo i lati, per $D(x) = \pm 1$. Questi nuovi iperpiani, tra loro distanti $\tau = \frac{2}{||w||}$, soddisfano le equazioni:

$$w \cdot \mathrm{x_i} + \mathrm{b} \geq +1 \qquad \text{se} \qquad \mathrm{y_i} = +1 \tag{2.2}$$

$$\mathrm{w} \cdot \mathrm{x_i} + \mathrm{b} \leq -1 \qquad \text{se} \qquad \mathrm{y_i} = -1 \tag{2.3}$$

  riscrivibili in

$$y_i(w^T \cdot x_i + b) \geq 1 \quad \text{per} \quad i = 1, \dots, n \tag{2.4}$$

  Il modello SVM ottimo separa correttamente le classi e massimizza il margine:

$$\mathrm{Loss} = \frac{\left|\left|\mathrm{w}\right|\right|^2}{2} \tag{2.5}$$

- Nel caso invece che **non tutti i pattern siano linearmente separabili** si possono introdurre variabili di slack positive $\xi_i, i = 1, \dots, n$ che rappresentano la deviazione dei pattern x_i dal margine. Per i pattern separabili $\xi_i = 0$. I vincoli di separazione tra classi risultano quindi:

$$\mathrm{y_i}(\mathrm{w^T} \cdot \mathrm{x_i} + \mathrm{b}) \geq 1 - \xi_\mathrm{i} \qquad \text{per} \qquad \mathrm{i} = 1, \dots, \mathrm{n} \tag{2.6}$$

  In questo caso il modello SVM massimizza il margine, ma allo stesso tempo minimizza il numero di elementi erroneamente classificati:

$$\mathrm{Loss} = \frac{\left|\left|\mathrm{w}\right|\right|^2}{2} + \mathrm{C} \sum_{\mathrm{i}=1,\dots,\mathrm{n}} \xi\mathrm{i} \tag{2.7}$$

con C iperparametro che rappresenta il peso di ogni pattern errato.

Conseguenza importante di questa descrizione geometrica è che il margine dell'iperpiano sia esclusivamente determinato dai pattern xi che giacciono più vicini: questi ultimi sono detti support vector.

2.1.2 SVM non lineari

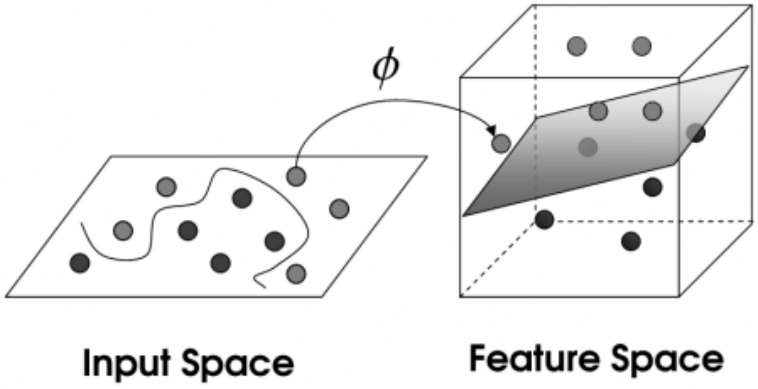


Figura 2.2: SVM non lineare

Il modello SVM non lineare viene applicato nei casi in cui il modello lineare fornisce risultati di scarso interesse a causa di un alto errore di classificazione. I modelli non lineari operano nel seguente modo:

1. Applicazione di un **mapping ϕ non lineare** (fig 2.2) dei pattern dallo spazio iniziale R^d verso uno spazio R^m a dimensionalità maggiore, detto feature space.
2. Separazione dei pattern tramite un **iperpiano** all'interno del feature space, grazie ai maggiori gradi di libertà dovuti all'aumento di dimensionalità.
3. Ritorno allo **spazio iniziale** per la classificazione.

Viene identificata con il nome di Kernel la funzione:

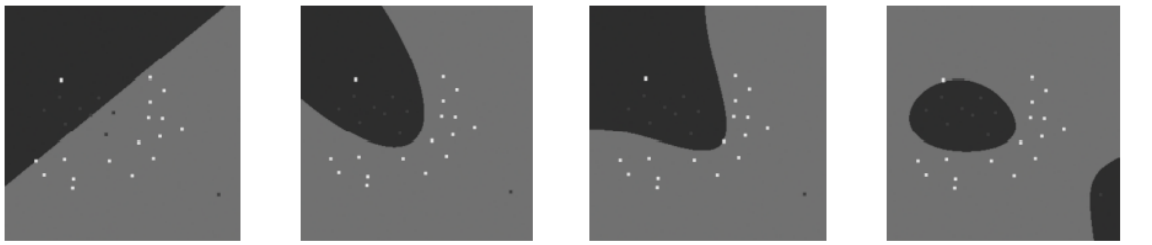
$$\mathrm{K}(\mathrm{x},\mathrm{x}') = \langle \phi(\mathrm{x}), \phi(\mathrm{x}') \rangle \tag{2.8}$$

dove $x, x' \in R^d$. Il fatto che il kernel possa essere rappresentato dal prodotto scalare tra i due vettori mappati nello spazio delle feature prende il nome di Kernel trick. La superficie di separazione è dunque espressa dalla formula:

$$\mathrm{D(x)} = \sum_{\mathrm{i=1,\dots,n}} \alpha_{\mathrm{i}}^{*} y_i K(x, x') + b^{*}$$

(2.9)

Alcuni esempi di funzione Kernel sono riportati in seguito (fig 2.3).



(a) lineare **(b)** polinomiale g.2 **(c)** polinomiale g.4 **(d)** radial based (rbf)

Figura 2.3: Esempi di Kernel function

È necessario notare che il rischio di overfitting aumenta al crescere della complessità della funzione Kernel, viceversa per l'underfitting, quindi è solitamente considerata buona pratica selezionare la funzione più semplice che non underfitti il problema

2.2 SVM multiclasse

Il modello SVM è stato ideato per trovare la superficie decisionale che separa 2 classi. Nel caso di problemi multiclasse si possono adottare diverse tecniche per raggiungere una soluzione adeguata:

- **One-against-one**, prevede l'addestramento di tutti i $s * (s-1)/2$ classificatori binari delle s classi. Lo step di classificazione dà un punteggio al pattern x da parte di ogni classificatore binario: infine tale pattern viene assegnato alla classe che ha ricevuto più voti (majority rule vote).
- **One-against-all**, prevede il calcolo di tutte le superfici decisionali tra i pattern di una classe e i rimanenti pattern di tutte le altre classi, è necessario dunque il training di s classificatori. Infine il pattern x viene assegnato alla classe per la quale la distanza dalla superficie decisionale è massima ($k^{*} = arg\, \underset{k}{max}\{D_{k(x)}\}$).

3 Enzimi

Questo capitolo introduce il concetto di enzima e la classificazione di tale macromolecola in base alla metrica dell'EC number.

3.1 Struttura

Gli enzimi sono proteine globulari che possiedono la funzione di catalizzatori biologici. Per comprendere appieno la loro funzione bisogna dunque accennare la struttura delle proteine.

3.1.1 Topologia

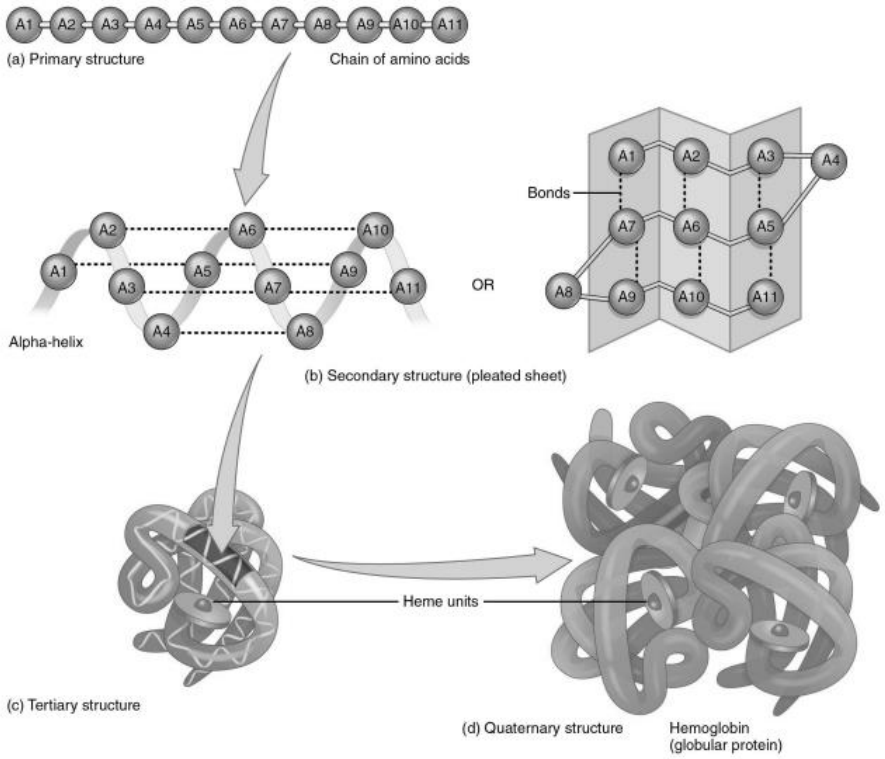


Figura 3.1: Struttura delle proteine

Le proteine presentano una struttura tridimensionale molto complessa e ad ogni configurazione è associata una funzione biologica. L'elemento chiave di ogni proteina è la sequenza di amminoacidi, molecole organiche con gruppo funzionale $-NH_2$ (amminico) o $-COOH$ (carbossilico).

Questa sequenza si sviluppa in 4 strutture tridimensionali principali (fig 3.1):

- struttura **primaria**, sequenza lineare di amminoacidi connessi dal legame peptidico (-CO-NH-) per formare un peptide.
- struttura **secondaria**, torsioni o ripiegamenti delle catene peptidiche in α-eliche o β-foglietti. La stabilità di questa struttura è garantita da legami idrogeno.
- struttura **terziaria**, disposizione tridimensionale delle strutture secondarie grazie a interazioni quali ponti disolfuro (S-S) e forze di Van der Waals (attrattive o repulsive tra molecole).
- struttura **quaternaria**, interazione tra strutture terziarie tramite legami deboli. La struttura terziaria alla base degli enzimi è di tipo globulare (sferica) e solitamente presenta dimensioni molto maggiori rispetto al substrato. Al contrario il punto di contatto con il substrato durante la catalisi (sito attivo) ha dimensioni ridotte.

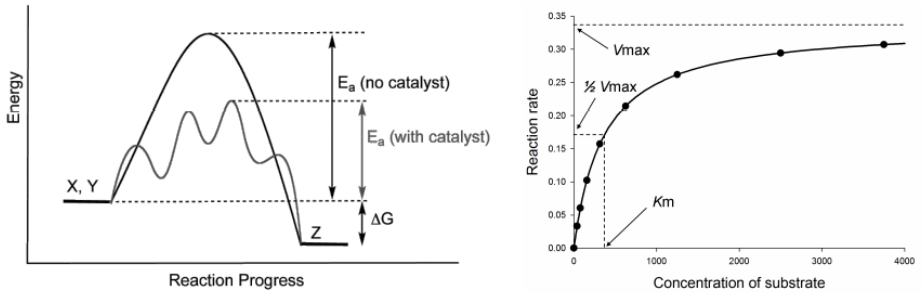
La struttura terziaria alla base degli enzimi è di tipo globulare (sferica) e solitamente presenta dimensioni molto maggiori rispetto al substrato. Al contrario il punto di contatto con il substrato durante la catalisi (sito attivo) ha dimensioni ridotte.

3.1.2 Rappresentazione

I metodi di rappresentazione degli enzimi di principale interesse sono:

- la **sequenza di amminoacidi** nella forma di una stringa: $sequenza = [a_1, a_2, \dots, a_n]$.
- l'insieme di **coordinate atomiche** della backbone, ovvero dell'ossatura dell'enzima. Tali coordinate sono nella forma (x, y, z) e identificano la successione dei singoli atomi della macromolecola.
- gli **angoli di torsione** presenti tra gli elementi della catena polipeptidica e la **distanza tra amminoacidi**. Questa rappresentazione non è stata presa in considerazione nell'ensemble proposto

3.2 Funzionamento



(a) Energia potenziale durante la reazione $X+Y \rightarrow Z$

(b) Velocità di reazione in funzione della concentrazione del substrato

Figura 3.2: Catalisi

In biochimica il ruolo degli enzimi risulta fondamentale per la catalisi dei processi biologici, ovvero la variazione dell'energia di attivazione e di conseguenza della velocità delle reazioni chimiche (fig 3.2). In una reazione chimica l'energia di attivazione rappresenta il livello minimo di energia potenziale necessaria per la collisione delle molecole dei reagenti. Data la generica reazione $A + nB \rightarrow mC$, dove per ogni m molecole di C scompaiono una molecola di A e n molecole di B, la velocità di reazione v è definita come la variazione della concentrazione molare di ognuna di queste sostanze nel tempo:

$$\mathrm{v} = -\frac{\mathrm{d[A]}}{\mathrm{dt}} = -\frac{1}{\mathrm{n}}\frac{\mathrm{d[B]}}{\mathrm{dt}} = \frac{1}{\mathrm{m}}\frac{\mathrm{d[C]}}{\mathrm{dt}} = \frac{\mathrm{d}\xi}{\mathrm{dt}} \tag{3.1}$$

con ξ grado di avanzamento della reazione.

Tale equazione può essere espressa anche come:

$$\mathrm{v} = \mathrm{k[A]^a[B]^b} \tag{3.2}$$

con a e b determinati sperimentalmente.

Il valore di k si calcola invece con l'equazione di Arrhenius

$$\mathrm{k} = \mathrm{Ae}^{-\frac{\mathrm{E_a}}{\mathrm{RT}}} \tag{3.3}$$

con A fattore pre-esponenziale, E_a energia di attivazione, R costante universale dei gas, T temperatura in Kelvin. Questi passaggi mettono in luce la relazione presente tra energia di attivazione e velocità di reazione.

3.2.1 Modelli

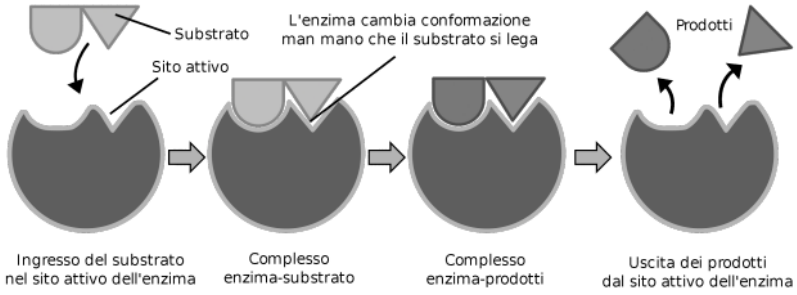


Figura 3.3: Catalisi, modello dell'adattamento indotto

Una delle peculiarità degli enzimi che li rende catalizzatori efficienti è il loro alto livello di specificità per la reazione catalizzata e i substrati coinvolti.

- Il primo modello che esprime appieno la specificità degli enzimi è definito come **chiave-serratura** [7], suggerito da Hermann Emil Fischer nel 1894, che prevede un alto livello di rigidità della struttura enzimatica.
- Il secondo modello definito come **adattamento indotto** [8] (fig 3.3), proposto nel 1958 da Daniel Koshland, sottolinea la flessibilità strutturale degli enzimi e suggerisce l'idea che il sito attivo possa adattarsi al substrato, risultando in un legame più stabile.

3.2.2 Classificazione

La classificazione degli enzimi segue lo schema dell'Enzyme Commission (EC) number: Ossidoreduttasi (EC1), Transferasi (EC2), Idrolasi (EC3), Liasi (EC4), Isomerasi (EC5) e Ligasi (EC6) sono le reazioni catalizzate.

4 EnzyNet

Questo capitolo introduce EnzyNet [2], un classificatore di reti neurali convoluzionali 3D progettato per predire l'Enzyme Commission number degli enzimi in base alla loro rappresentazione spaziale a voxel. Il paper di EnzyNet è stato pubblicato nel 2018 e fornisce la base dalla quale questo elaborato di tesi è partito per lo sviluppo dell'ensemble.

4.1 Dataset

Il dataset è costituito da 63 558 enzimi scaricabili nel Marzo del 2017 da RCSB PDB (https://www.rcsb.org/) ed è stato suddiviso con proporzioni 80/20% tra training e test, inoltre il 20% del training è stato riservato per lo step di validation

4.2 Estrazione delle feature

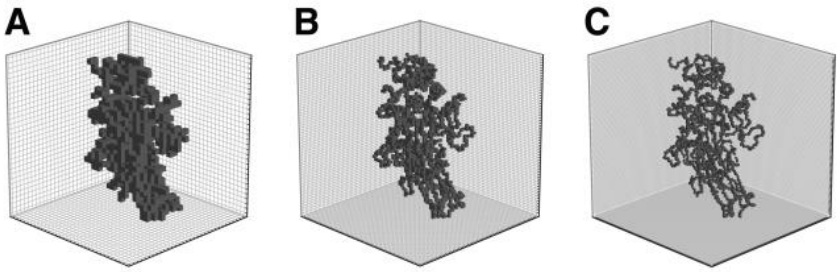


Figura 4.1: Voxelizzazione con risoluzione l pari a 32,64,96

La parte di estrazione delle feature rappresenta il punto di forza del progetto EnzyNet, in quanto semplifica notevolmente il modello tridimensionale della backbone degli enzimi e riduce notevolmente i tempi di addestramento della rete CNN.
Gli step necessari per estrarre le feature dal dataset delle coordinate sono i seguenti:

1. **Estrazione delle informazioni strutturali**, Le coordinate degli atomi della backbone vengono lette dai file PDB.
2. **Interpolazione**. Coordinate consecutive $\left(\overrightarrow{A_\iota}, \overrightarrow{A_{\iota+1}}\right)$ vengono interpolate con p nuovi punti equidistanti per completare eventuali buchi

$$\frac{(1-k+1)*\overrightarrow{A_\iota}+k*\overrightarrow{A_{\iota+1}}}{p+1} \quad (4.1)$$

dove k varia da 1 a p.

3. **Regolazione delle dimensioni**. Il baricentro S delle coordinate viene traslato in $(0,0,0)$ e viene applicata una trasformazione omotetica di centro S e fattore

$$\lambda = \left\lfloor \frac{l}{2} - 1 \right\rfloor * \frac{1}{\mathrm{R_m ax}} \tag{4.2}$$

per ottenere un grado di occupazione uniforme per ogni enzima del volume $lxlxl$.

4. **Orientamento**. L'enzima viene orientato tramite l'analisi delle componenti principali, in inglese principal component analysis (PCA)
5. **Augmentation con flip**. La p_{flip} indica la probabilità con la quale le coordinate vengono specchiate rispetto ai tre assi (x, y, z). Il numero di trasformazioni possibili è quindi $2^3 - 1$.
6. **Voxelizzazione**. Il baricentro S viene traslato in $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ e le coordinate vengono trasformate in voxel binari, notevolmente più semplici rispetto alle coordinate originali per il training della CNN.
7. **Outliers**. Vengono eliminati i voxel senza vicini.

4.3 Struttura della rete

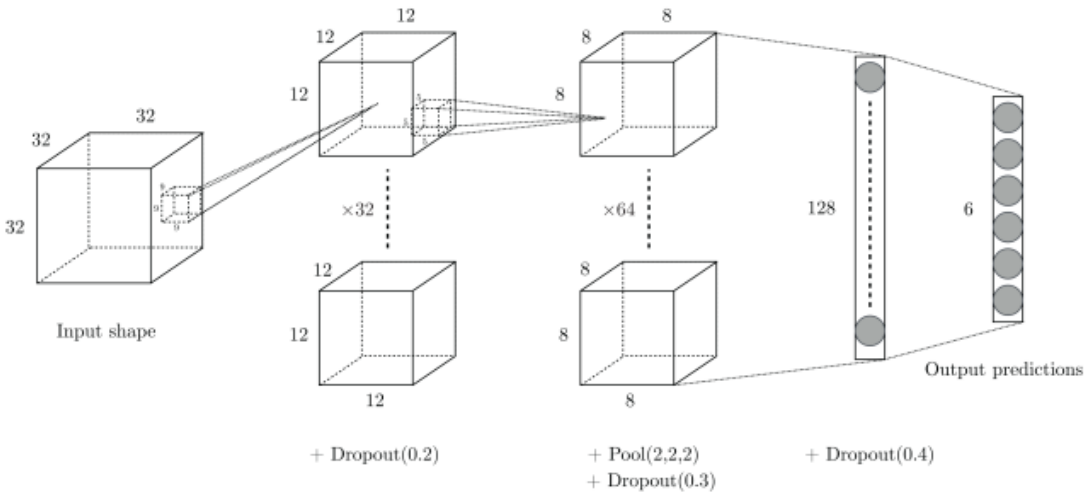


Figura 4.2: CNN alla base del progetto EnzyNet

Come rete neurale convoluzionale è stata considerata la seguente architettura di layer:

1. **Input** di dimensioni 32x32x32 contenente i dati dei voxel.

2. **Convoluzionale** con 32 filtri di dimensione 9x9x9 e stride 2
3. **Convoluzionale** con 64 filtri di dimensione 5x5x5 e stride 1.
4. **Max-Pooling** di dimensione 2x2x2 e stride 2.
5. **Fully-Connected** con 128 unità.
6. **Fully-Connected** con 6 (numero di classi EC number) unità.
7. **Softmax** per l'output delle probabilità delle singole classi.

Sono state tenute in considerazione anche tecniche per la regolarizzazione quali:

- **Leaky ReLU** con parametro α = 0.1 come funzione di attivazione in seguito ad ogni layer convoluzionale.
- **L2 regularization** con parametro $\lambda = 0.001$.
- **Dropout** con parametri 0.2, 0.3, 0.4.

In totale la rete contiene 804614 parametri distinti (bias inclusi), che corrisponde ad una diminuzione di circa 13% rispetto alla rete VoxNet [9], punto di riferimento per il progetto EnzyNet.

L'ottimizzatore scelto per il processo di gradient descent è Adam [10].

4.4 Performance

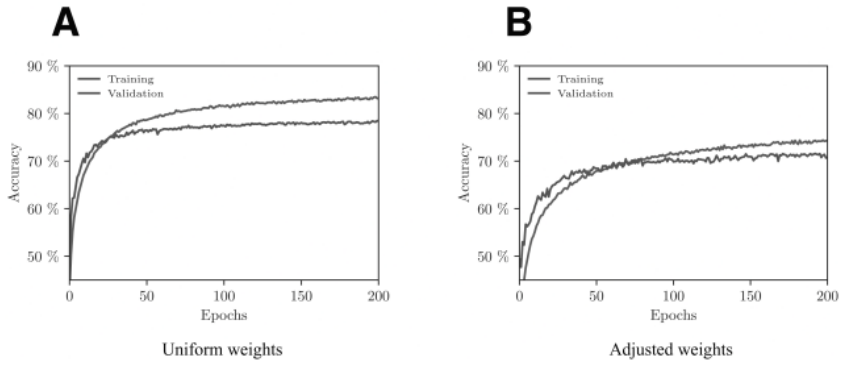


Figura 4.3: Performance mostrate nel paper EnzyNet

I risultati delle performance di EnzyNet mostrano un'accuracy pari a 77.6% nella classificazione dell'EC number, ottenuta con i seguenti iperparametri:

- **Raggio** $R_{max} = 40$, dimensione dimostratasi perfetta per racchiudere al meglio tutti gli enzimi.
- **Interpolazione** di $p = 0$ punti, in quanto non efficace per il volume 32x32x32.
- **Probabilità di flip** $p_{flip} = 0$, alto costo computazionale, rivelatasi efficace solo per classi sotto-rappresentate.

5 Classificatori

I classificatori adottati per il questo studio utilizzano una CNN basata sul progetto EnzyNet [2] per l'elaborazione della rappresentazione spaziale degli enzimi come voxel binari: il primo utilizza due varianti dell'ottimizzatore Adam, il secondo utilizza l'ottimizzatore Sgdm, ed entrambi forniscono la somma degli score dopo 5 retrain. Il terzo classificatore è un SVM multiclasse che implementa la libreria LibSVM [3] ed elabora la rappresentazione degli enzimi come sequenze di amminoacidi. L'ambiente di sviluppo utilizzato è MATLAB R2022a.

5.1 Dataset

Il dataset utilizzato fa riferimento al progetto EnzyNet, 63 556 file XXXX.pdb in formato PDB (https://www.wwpdb.org/documentation/file-format), contro i 63 558 di EnzyNet in quanto 2 enzimi non risultavano più presenti nella Protein Data Bank [1].

La prima parte del codice si occupa dell'estrazione delle informazioni strutturali e delle sequenze dai file PDB, letti con il comando PDBStruct = pdbread(File).

- Le **coordinate atomiche** sono nella forma (x, y, z) per ogni atomo della backbone e vengono estratte leggendo i valori in PDBStruct.Model.Atom. La matrice così ottenuta è scritta su file.
- Le **sequenze di amminoacidi** sono stringhe nella forma: $sequenza = [c_1, c_2, \dots, c_n]$, estratte da PDBStruct.Sequence. La singola stringa è scritta su file.

5.2 Feature extraction

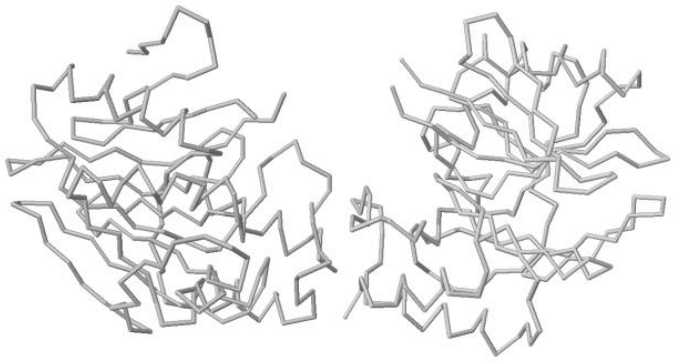


Figura 5.1: Enzima 2BFZ, esempio di una backbone a 2 catene distinte

Nella fase di feature extraction il codice si occupa di estrarre i valori che verranno poi direttamente utilizzati nel training dei modelli di classificazione. Ogni pattern viene associato alla label corrispondente (coppie features-label) tramite un dizionario creato con il file dataset_single.csv di EnzyNet. Infine il dataset è diviso in dsTrain e dsTest in proporzioni 80/20%, seguendo lo schema del file partition_single.csv di EnzyNet, e con l'unione di training e validation

- **Voxel binari** sono estratti dalle coordinate atomiche (esempio fig 6.1), seguendo lo schema esposto nel paragrafo 4.2. Il volume è di 32x32x32 voxel, vengono interpolati $p = 0$ punti in quanto l'interpolation non offrirebbe vantaggi a tale risoluzione, $R_{max} = 40$ per racchiudere al meglio tutti gli enzimi, non vengono implementati step di data augmentation.
- **Vettori 1x40** numerici di feature sono calcolati a partire delle sequenze di amminoacidi. Questo processo prevede l'utilizzo di un vettore pre-formato dal quale vengono estratti ed elaborati i valori opportuni a seconda della sequenza.

5.3 Training

Il training dei modelli di classificazione ha come scopo l'output di tabelle di score di ogni classe, per ogni pattern, dalle quali può essere calcola l'ACC.

- Le **feature-voxel** vengono trattate come immagini tridimensionali a singolo canale e forniscono l'input delle CNN così costruite (vedi 4.3):

```
layers = [...
        image3dInputLayer([32 32 32 1], 'Normalization', 'none'), ...
        convolution3dLayer(9, 32, 'Stride', 2), ...
        leakyReluLayer(0.1), ...
        dropoutLayer(0.2), ...
        convolution3dLayer(5, 64, 'Stride', 1), ...
        leakyReluLayer(0.1), ...
        maxPooling3dLayer(2), ...
        dropoutLayer(0.3), ...
        fullyConnectedLayer(128), ...
        dropoutLayer(0.4), ...
        fullyConnectedLayer(6), ...
        softmaxLayer];
```

La scelta degli iperparametri ricade su: 10 retrain della rete, mini-batch di dimensione 15, 20 epoche con shuffle ogni epoca, learning rate pari a 1e-3. Tra i due classificatori che si basano sulla CNN il primo implementa le varianti 15 e 17 (Exp e ExpLR) dell'ottimizzatore Adam definite nel file WeightLR.m, il secondo implementa l'ottimizzatore Sgdm.

- Le **feature-sequenze** vengono classificate con un modello SVM che adotta la tecnica one-against-all per la gestione di più classi. I parametri scelti sono: svmtrain(training_label, training_set, '-t 2 -g 0.1 -c 1000'); Il Kernel utilizzato è quindi una Radial Basis Function (RBF): $K(x, x') = \exp(-\gamma\|x - x'\|2)$, con $\gamma = 0.1$ e costo C = 1000. Le tabelle degli score vengono poi normalizzate e sommate tra loro per la creazione dell'ensemble.

5.4 ACC, ROC Curve e AUC

I risultati ottenuti mostrano l'efficacia della fusione dei metodi basati sulle coordinate atomiche con quello basato sulla sequenza di amminoacidi. Le metriche utilizzate sono:

- **Accuracy**, permette di stimare il livello di precisione mediante confronto della classe scelta con le label del test set $\big(ACC = (TP + TN)/(P + N)\big)$.
- **ROC Curve**, grafico che mette in relazione il *False Positive Rate* (FPR) con il *True Positive Rate* (TPR). L'**AUC**, l'area sottostante tale grafico con valori da 0 a 1, indica il potere discriminante del modello.

In seguito sono riportati i valori medi di ACC e AUC, sia per i singoli classificatori che per l'ensemble. Si nota un sostanziale miglioramento in seguito alla fusione degli score.

	Adam	Sgdm	SVM	Ensemble
ACC	68.42%	68.87%	67.99%	74.75%
1-AUC	0.1429	0.1333	0.1401	0.0835

Tabella 5.1: Confronto tra i classificatori in ACC e AUC

6 Architettura dei Modelli

Questo studio si focalizza sulla variazione della topologia dell'ottimizzatore Sgdm(vedi cap 5) e l'utilizzo di pre-training per migliorarne le prestazioni. In particolare vengono presentate due nuove topologie: SGDM_eluLayer e SGDM_bach_norm, assieme a SGDM_pretrain. I dataset utilizzati sono gli stessi dei classificatori basati su EnzyNet (vedi 5.1) con la differenza che per i training set anziché utilizzare 50844 proteine, ne vengono utilizzate 5084.

Inoltre presenta due nuove funzioni per la creazione dei voxel (vedi 5.2)

6.1 SGDM_eluLayer

La prima CNN proposta utilizza, come funzione di attivazione, *Exponential Linear Unit* (ELU) anziché *Leaky Rectified Linear Unit* (ReLU) (vedi cap 4.3).

6.1.1 Exponential linear unit (ELU)

L' *Exponential Linear Unit* (ELU) è una funzione di attivazione per le reti neurali. A differenza dei ReLU, gli ELU hanno valori negativi che consentono loro di spingere le attivazioni medie delle unità più vicino allo zero, come la normalizzazione dei batch, ma con una complessità computazionale inferiore.
Un livello di attivazione ELU esegue l'operazione di identità su input positivi e una non linearità esponenziale su input negativi, attraverso la seguente operazione:

$$\begin{aligned} f(x) &= x && \text{se} \quad x \geq 0 \\ f(x) &= \alpha(exp(x) - 1) && \text{se} \quad x < 0 \end{aligned} \tag{6.1}$$

6.1.2 Topologia

La CNN è così costituita:

```
layers = [...
    image3dInputLayer([32 32 32 1], 'Normalization', 'none'), ...
    convolution3dLayer(9, 32, 'Stride', 2), ...
    eluLayer(0.1, 'Name','elu1'), ...
    dropoutLayer(0.2), ...
```

```
convolution3dLayer(5, 64, ’Stride’, 1), ...
eluLayer(0.1, 'Name','elu2'), ...
maxPooling3dLayer(2), ...
dropoutLayer(0.3), ...
fullyConnectedLayer(128), ...
dropoutLayer(0.4), ...
fullyConnectedLayer(6), ...
softmaxLayer];
```

La scelta degli iperparametri ricade su: 5 retrain della rete, mini-batch di dimensione 15, 20 epoche con shuffle ogni epoca, learning rate pari a 1e-3.

6.2 SGDM_bach_norm

La seconda CNN proposta utilizza un *batch normalization layer* in aggiunta alla normale topologia della Sgdm

6.2.1 Batch normalization layer

Un livello di normalizzazione batch normalizza un mini-batch di dati in tutte le osservazioni per ciascun canale in modo indipendente. Per accelerare l'addestramento della rete neurale convoluzionale e ridurre la sensibilità all'inizializzazione della rete.

6.2.2 Topologia

La CNN è così costituita:

```
layers = [...
    image3dInputLayer([32 32 32 1], ’Normalization’, ’none’), ...
    convolution3dLayer(9, 32, ’Stride’, 2), ...
    batchNormalizationLayer('Name','batchnorm1'), ...
    eluLayer(0.1, 'Name','elu1'), ...
    dropoutLayer(0.2), ...
    convolution3dLayer(5, 64, ’Stride’, 1), ...
    eluLayer(0.1, 'Name','elu2'), ...
    maxPooling3dLayer(2), ...
    dropoutLayer(0.3), ...
    fullyConnectedLayer(128), ...
```

```
dropoutLayer(0.4), ...
fullyConnectedLayer(6), ...
softmaxLayer];
```

La scelta degli iperparametri ricade su: 5 retrain della rete, mini-batch di dimensione 15, 20 epoche con shuffle ogni epoca, learning rate pari a 1e-3.

6.3 Pretrain

L'ultima CNN proposta utilizza la stessa topologia di Sgdm iniziale. Sostanzialmente la differenza, rispetto al precedente modello, è dovuta dal fatto che la rete viene pre-addestrata su un primo set di dati, subendo un ciclo di pretraining,, prima dell'effettivo allenamento. Il dataset utilizzato per il pretraining è della stessa grandezza di quello utilizzato per l'allenamento.

6.4 Creazione Voxel

Le due nuove funzioni proposte per la creazione dei voxel nella fase di feature extraction(vedi 5.2) sono *vectorizedMakeVoxels* e *vectorizedMakeVoxels_bsxfun*. Entrambe tengono lo stesso scheletro della funzione originale, utilizzando in aggiunta le funzioni *all()* e *bsxfun()* volte a ridurre il tempo di computazione.

7 Risultati

I risultati ottenuti mostrano l'efficacia del pretraining della rete e delle due differenti topologie proposte. La metrica utilizzata è accuracy(vedi 5.4). In seguito sono riportati i valori di ACC sia del classificatore Sgdm iniziale, sia dei nuovi modelli.

	Sgdm	ELU Layer	Bach Normalization Layer	Pretarin
ACC	56.62%	57.18%	56.37%	57.76%

Tabella 7.1: Confronto tra i classificatori in ACC

I risultatati ottenuti per la creazione dei voxel sono ottenuti attraverso le funzioni di MatLab *tic* e *toc*. Queste misurano il tempo impiegato per eseguire il codice, in particolare *tic* registra l'ora corrente, mentre *toc* utilizza il valore registrato per calcolare il tempo trascorso. In seguito sono riportati, in secondi, i tempi della funzione originale (*makeVoxel*) e delle due nuove versioni, per la computazione della fase di feature extraction con volumi 32x32x32 e 48x48x48.

Volume size	makeVoxel	vectorizedMakeVoxel	vectorizedMakeVoxel_bsxfun
32	0.0822	0.0787	0.0790
48	49.315	49.398	49.285

Tabella 7.2: Confronto tra le funzioni per la creazione dei voxel

7.1 Conclusioni

Nonostante il numero inferiore dei dati rispetto allo modello preso in considerazione, si evince l'incremento di performance, sebbene non sostanziale, per quanto riguarda ACC rispetto alle nuove topologie. Ciò comporta che, sostituendo i nuovi modelli a quelli originali, si otterrebbero risultati migliori anche nell'ensemble. Per quanto riguarda la fase di feature extraction, utilizzando le nuove versioni per la creazione dei voxel, si ottengono risultati migliori per volumi di 32x32x32 mentre per volumi di 48x48x48 i risultati sono pressoché analoghi.

Bibliografia

[1] https://www.wwpdb.org *Protein Data Bank*

[2] Afshine Amidi, Shervine Amidi, Dimitrios Vlachakis, Vasileios Megalooikonomou, Nikos Paragios, Evangelia I. Zacharaki (2018) *EnzyNet: enzyme classification using 3D convolutional neural networks on spatial representation*

[3] Chih-Chung Chang, Chih-Jen Lin (2001) *LIBSVM: A Library for Support Vector Machines*

[4] Warren S. McCulloch, Walter Pitts (1943) *A Logical Calculus of the Ideas Immanent in Nervous Activity*

[5] Kurt Hornik, Maxwell Stinchcombe, Halbert White (1989) *Multilayer Feedforward Networks are Universal Approximators*

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014) *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*

[7] Hermann Emil Fischer (1894) *Einfluss der Configuration auf die Wirkung der Enzyme*

[8] Daniel Edward Koshland (1958) *Application of a Theory of Enzyme Specificity to Protein Synthesis*

[9] Daniel Maturana, Sebastian Scherer (2015) *VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition*

[10] Diederik P. Kingma, Jimmy Lei Ba (2014) *Adam: A Method for Stochastic Optimization*