Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Dal Natural Language Processing alle proteine: apprendimento automatico di sequenze mediante Transformer

Relatore:
Chiar.mo Prof.
Andrea Asperti

Correlatrice:
Chiar.ma Prof.ssa
Alessandra Carbone

Presentata da:
Andrea Mancini

Sessione III
Anno Accademico 2023/2024

Abstract

Classificare milioni di proteine in base alla loro funzione rappresenta una sfida computazionale significativa. I metodi tradizionali come BLAST o modelli di Markov mostrano limiti nel riconoscimento di pattern distanti e funzioni distinte. Partendo da ProfileView, un software per l'identificazione di classi funzionali, il nostro obiettivo è classificare su larga scala proteine, attingendo a vasti database e ai continui progressi nel sequenziamento ad alto rendimento. Sulla base dei recenti progressi fatti nel Natural Language Processing con l'architettura Transformer, abbiamo implementato DomainSpanESM per l'identificazione di schemi nelle sequenze funzionalmente correlate, facilitando la classificazione di milioni di sequenze in gruppi definiti. Inoltre, grazie all'allineamento di sequenze, sono stati riconosciuti diversi amminoacidi altamente conservati, ritrovati frequentemente lungo le stesse posizioni nella matrice, e risultati cruciali per la definizione delle classi, offrendo spunti sulle sfumature funzionali di ognuna di esse. Sperimentato su famiglie proteiche di rilievo come Thioredoxin (TRX) e Cryptochrome/Photolyase (CPF), DomainSpanESM rappresenta un progresso nell'annotazione e classificazione funzionale su larga scala, migliorando la comprensione delle funzioni proteiche e delle loro relazioni evolutive.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

L'analisi e la classificazione delle sequenze proteiche rappresentano uno degli aspetti più importanti e cruciali in bioinformatica. L'attività biochimica alla base della vita è orchestrata dalle proteine, catene polimeriche composte da amminoacidi, detti residui (esempio tabella 1.1). Queste molecole possono operare autonomamente oppure in complessi multi-chain, svolgendo ruoli fondamentali attraverso interazioni e il controllo di numerosi processi nell'ambiente cellulare, dove convivono proteine e altre macromolecole. Le reazioni chimiche che avvengono in questo ambiente, guidate da fattori termodinamici e cinematici, richiedono che le proteine mantengano un insieme conformazionale funzionale stabile. La specificità della funzione proteica è determinata dalle proprietà fisiche e chimiche del suo ambiente e risulta codificata nella sua struttura primaria, la sequenza, fondamento del dogma centrale della biologia [7]. Il dogma stabilisce la dipendenza univoca della funzione alla struttura, e della struttura alla sequenza; in questo modo lo spostamento dell'informazione avviene solamente nella direzione sequenza-struttura-funzione. Tuttavia, questo paradigma risulta spesso troppo superficiale, in quanto descrivere la funzione di ogni proteina richiede un approccio dettagliato e personalizzato. La caratteristica comune responsabile per la funzione proteica è come queste interagiscono tra di loro. La forza del legame può essere forte o debole dipendentemente dall'ambiente in cui si trovano e dalle loro sequenze [8].

L'avvento delle tecnologie per il sequenziamento ad alta produttività, noto anche come Next Generation Sequencing, ha portato ad un incremento sostanziale del numero di sequenze proteiche depositate nei database pubblici, come UniProt e Swiss-Prot (www.uniprot.org). Questa grande disponibilità ha reso indispensabili strumenti computazionali avanzati per l'analisi e l'interpretazione delle numerose sequenze. In particolare, la bioinformatica si occupa proprio di affrontare le sfide legate alla gestione di big data biologici, includendo metodi statistici, algoritmi di machine learning e, più recentemente, tecniche di deep learning (DL), arrivando a risolvere i task più complessi. Un esempio è AlphaFold [9], che ha permesso di prevedere con elevata accuratezza la struttura tridimensionale delle proteine, e contribuito ad ottenere il premio nobel per la chimica nel 2024 ad alcuni dei suoi autori.

L'importanza dello studio delle sequenze proteiche, oltre per ricostruire la struttura delle proteine fondamentali per la comprensione della loro funzione, risiede nell'impatto diretto che tali analisi hanno sullo sviluppo di nuove terapie e la progettazione di farmaci. In questo scenario, la capacità di classificare e annotare accuratamente le proteine diventa un passaggio cruciale per tradurre la mole di dati in materiale utile e applicabile per i biologi. Nonostante la grande quantità di sequenze disponibili, solo una frazione di esse è accompagnata da informazioni riguardanti la struttura, e la funzione. Nelle ultime statistiche pubblicate di Uniprot, su un totale di circa 235M, solo lo 0.15% sono annotate funzionalmente, ancora meno lo 0.01% ha informazioni sul dominio (https://www.uniprot.org/uniprotkb/statistics). In particolare, il dominio è una porzione di una sequenza proteica, non necessariamente contigua, che ricorre frequentemente in proteine della stessa famiglia. Il suo riconoscimento avviene attraverso il confronto con altre sequenze proteiche, rivelando una struttura conservata spesso associata a una specifica funzione.

Questa disparità evidenzia l'importanza di estrarre queste informazioni da dati prevalentemente grezzi. Le proteine, inoltre, possono presentare diverse complessità come la presenza di domini multipli, variazioni evolutive (mutazioni) e interazioni con altre biomolecole. Queste informazioni sono

estratte analizzando la conservazione dei residui, ovvero gli amminoacidi ritrovati lungo le stesse posizioni all'interno delle sequenze prese in esame. Per esempio è possibile risalire alle cause di un problema genetico tramite l'albero evolutivo di una proteina osservando quali amminoacidi sono mutati e quali mantenuti. Per fare questo sono stati implementati metodi di allineamento e annotazione come BLAST [10] o i modelli di Markov. Questi strumenti, cosidetti tradizionali, hanno un limite che si basa su confronti di sequenza diretti o su modelli probabilistici che possono non essere sufficientemente sensibili per identificare pattern distanti o funzioni distinte in sequenze con identità elevata, note anche come omologhe.

La ricerca, spinta dal grande successo dell'intelligenza artificiale e dall'apprendimento automatico, è fortemente orientata all'utilizzo di tecniche di DL, che consentono di individuare autonomamente le parti cruciali da sequenze di lunghezza variabile, per estrarre importanti informazioni come la funzione. Questo avviene perchè sono in grado di considerare relazioni complesse non lineari tra i dati rispetto ai metodi tradizionali. Attualmente, gli approcci più diffusi si ispirano direttamente alle architetture e ai metodi impiegati nel campo del Natural Language Processing (NLP). Dai rinomati Transformer e Large Language Models, derivano i Protein Language Models [11], progettati per applicare le straordinarie prestazioni ottenute nell'elaborazione del linguaggio naturale all'analisi del "linguaggio" delle proteine, ovvero le loro sequenze di amminoacidi.

Sulla base di queste premesse, in questo studio ho sviluppato un'applicazione per la classificazione funzionale di sequenze proteiche, mediante tecniche di DL di tipo supervisionato. In particolare ho svolto una ricerca in collaborazione con il "Laboratory of Computational and Quantitative Biology" (LCQB) dell'università "Sorbonne" con la professoressa A. Carbone. Partendo dall'utilizzo di ProfileView, un framework sviluppato da LCQB [1], costui identifica in maniera efficacie classi e sottoclassi funzionali di sequenze proteiche, gestendo alcune migliaia di sequenze alla volta. Questo approccio non supervisionato evidenzia la necessità di annotare più sequenze contem-

poraneamente, anziché in isolamento, per facilitare la scoperta di nuove classi funzionali e l'identificazione di classi alternative tramite confronto. Inoltre, consente di estrarre informazioni cruciali sui singoli residui, essenziali per specifiche sottoclassi funzionali.

L'obiettivo di questo lavoro è ampliare la classificazione effettuata da ProfileView e ottenere una classificazione funzionale su larga scala di milioni di proteine, in linea con l'enorme quantità di dati disponibili nei database, prodotti dai continui sforzi di sequenziamento ad alto rendimento. Per affrontare analisi su larga scala, comprendendo centinaia di migliaia fino a milioni di sequenze all'interno di una famiglia proteica, abbiamo progettato DomSpanESM, un'architettura Transformer, in grado di riconoscere pattern ricorrenti nelle sequenze funzionalmente correlate. Queste sequenze, inizialmente discriminate in piccoli gruppi da ProfileView, vengono così classificate su larga scala in classi funzionali definite. Il nostro metodo rappresenta un progresso significativo nell'annotazione e classificazione funzionale delle proteine su larga scala, contribuendo a una migliore comprensione delle loro funzioni e delle relazioni evolutive. La sua validità è stata testata su due famiglie di proteine importanti in natura la Thioredoxin (TRX) e la famiglia Cryptochrome/Photolyase (CPF), in particolare la TRX è altamente diffusa in natura presente in quasi tutti gli organismi con funzioni diverse, candidandosi quindi come ottimo caso di test.

Basandosi su ProfileView, DomSpanESM può consentire di analizzare milioni di sequenze provenienti da database come MGnify [12] e di effettuare una prima classificazione dei loro domini proteici. In questo modo, il nostro approccio supera i limiti di database come CATH/FunFam [13], attualmente ristretti a poche centinaia di sequenze.

L'elaborato sarà così organizzato da un riassunto dei progressi fatto sullo Stato dell'Arte (Sezione 2), seguito da una presentazione dell'architettura del modello (Sezione 3) ed il suo addestramento (Sezione 4); per concludere con diversi esperimenti (Sezione 5) tra cui un'analisi comparativa con diverse tecniche di ML e DL per un confronto e la validazione delle performance

raggiunte (Sezione 6) .

Aminoacido	**Abbreviazione (3 lettere)**	**Abbreviazione (1 lettera)**
Alanina	Ala	A
Arginina	Arg	R
Asparagina	Asn	N
Acido aspartico	Asp	D
Cisteina	Cys	C
Acido glutammico	Glu	E
Glutammina	Gln	Q
Glicina	Gly	G
Istidina	His	H
Isoleucina	Ile	I
Leucina	Leu	L
Lisina	Lys	K
Metionina	Met	M
Fenilalanina	Phe	F
Prolina	Pro	P
Serina	Ser	S
Treonina	Thr	T
Triptofano	Trp	W
Tirosina	Tyr	Y
Valina	Val	V
Selenocisteina	Sec	U
Pirrolisina	Pyl	O

Tabella 1.1: Lista 22 amminoacidi e abbreviazioni

Capitolo 2

Stato dell'arte

Negli ultimi vent'anni, le tecniche di classificazione delle sequenze proteiche sono evolute significativamente, passando da metodi tradizionali, basati sulla manipolazione delle sequenze, ad approcci più avanzati di Machine Learning e Deep learning. Le prime si basano principalmente sull'allineamento delle sequenze e sull'inferenza per omologia, utilizzando strumenti come BLAST per identificare somiglianze tra proteine sconosciute e quelle con funzioni già note [10].

I limiti di queste tecniche, benchè pilastri della bioinformatica, appaiono chiari nei casi di scarsa omologia e similarità tra le sequenze, oppure dove sequenze molto simili hanno classi distinte. Questo ha portato ad adoperare tecniche di ML che estraessero features come il motivo degli amminoacidi e descrittori della struttura proteica, da classificare mediante Support Vector Machines e metodi 'kernel-based' in generale [14].

Nel caso in cui l'informazione da classificare non è conosciuta un altro tipo di algoritmi largamente adoperato è il clustering. In [15] vengono analizzati diverse tecniche per raggruppare per funzione reti di associazione tra proteine mediante l'informazione di interazione e del contesto genomico. Un altro esempio è proprio ProfileView sviluppato da LCQB, il quale adopera clustering gerarchico mediante varianza di Ward per il raggruppamento delle sequenze all'interno di uno spazio funzionale. Le sequenze infatti sono

rappresentate in questo spazio utilizzando un embedding ricco di informazione funzionale, con l'obbiettivo di individuare cluster di funzioni diverse. Maggiori informazioni sono descritte nel capitolo 3.

L'aumento della quantità di dati disponibili, insieme al miglioramento della potenza computazionale, ha portato allo sviluppo di numerosi approcci "ensemble" e basati su reti, capaci di integrare informazioni biologiche provenienti da fonti diverse, dal contesto genomico alle reti di interazione tra proteine, con l'obiettivo di aumentare l'accuratezza delle predizioni [16].

L'introduzione del deep learning ha ulteriormente rivoluzionato il campo: reti neurali convoluzionali (CNN) e reti neurali ricorrenti (RNN) sono state impiegate per catturare relazioni complesse e non lineari tra le variabili, obiettivi che in precedenza risultavano irraggiungibili. Ad esempio, i modelli DeepGO e DeepGOPlus hanno dimostrato che utilizzando un classificatore gerarchico con CNN possono prevedere con successo le annotazioni del Gene Ontology, un sistema di rappresentazione standard e strutturato per descrivere le funzioni di geni e proteine, apprendendo direttamente sia dalle sequenze che dai dati di interazione [16, 17]. Gli autori di SPOT-Disorder [18] hanno implementato con successo un'architettura Long-Short Term Memory (LSTM) bidirezionale per la predizione del disordine nelle strutture delle proteine; grazie alla capacità dei LSTM di catturare relazioni complesse tra posizioni distanti sono stati in grado di identificare le regioni interessate in diverse punti di diversa lunghezza.

Diversamente al nostro approccio sull'analisi di sequenze omologhe, HiFun [19] è una architettura indipendente dalla natura delle sequenze; inizialmente trasforma le sequenze proteiche in rappresentazioni numeriche utilizzando la combinazione di due metodi di embedding: uno basato sulle matrice di sostituzioni BLOSUM62 e un altro che utilizza FastText. I vettori basati su BLOSUM62 vengono elaborati da layer convoluzionali per estrarre le caratteristiche evolutive, mentre le matrici basate su FastText passano attraverso una sotto-architettura che combina CNN e BiLSTM con un meccanismo di self-attention. I layer CNN estraggono le caratteristiche n-gram dalle sequen-

ze, mentre i BiLSTM catturano le informazioni contestuali sia in avanti che all'indietro. Il meccanismo di self-attention si concentra maggiormente sugli amminoacidi rilevanti per la predizione della funzione proteica.

Ultima frontiera nel DL sono i Protein Language Models (PLMs) [20], implementati tramite Transformers, che hanno rivoluzionato la nostra abilità nel classificare le proteine tramite l'apprendimento di una rappresentazione interna ricca di informazione da enormi dataset. Questi modelli, come Evolutionary Scale Modeling (ESM) e Ankh [21, 22], riescono a catturare le relazioni intrinseche tra gli amminoacidi, codificando efficacemente informazioni biochimiche relative all'evoluzione e la struttura comprimendole in rappresentazioni ad alta dimensionalità. Grazie al meccanismo del self-attention, i transformers permettono ad ogni residuo presente nella sequenze di interagire con tutti gli altri definendo quindi un contesto per il residuo, rivelando così anche dipendenze distanti tra essi, critiche per determinare la funzione di una proteina.

I modelli apprendono con questa dinamica una rappresentazione interna dell'informazione, risultato di una compressione di pattern complessi derivanti dalla variazione e conservazione delle sequenze in input. Per esempio ESM, dimostra che un pretraining non supervisionato su milioni di sequenze può produrre una rappresentazione che generalizza molto bene tra le diverse famiglie proteiche; valorizzando così le successive task come l'annotazione funzionale e la predizione della struttura. Similmente, Ankh raffina ancora queste rappresentazioni, soprattutto per un utilizzo generale del modello dimostrando un miglioramento con un numero ridotto di parametri.

Nell'articolo ProtTrans [23], vengono confrontate sei diverse soluzioni, due auto-regressive e quattro auto-encoder, per un'analisi della bontà delle rappresentazioni interne apprese di questi modelli su diversi task. In particolare due principali architetture: un modello ispirato a BERT (ProtBERT) che tramite masked language modeling apprende rappresentazioni contestuali bidirezionali, e un modello basato su T5 (ProtT5) con denoising sequence-to-sequence, dove parti delle sequenze vengono corrotte con l'ob-

biettivo di ricostruirle. Entrambi sono stati allenati su UniRef [24] e BFD (https://bfd.mmseqs.com), dataset molto ampi con miliardi di dati, ottenendo ottimi risultati, in particolare ProtT5 raggiunge lo stato dell'arte per la predizione singola dei residui.

ProGen [25] è un PLM che rappresenta un approccio innovativo nel campo della generazione di sequenze proteiche. Utilizza lo stesso processo di pre-training adoperato dagli altri PLM, ispirato al language modeling e alla predizione del prossimo token. Il modello è composto da 1.2B parametri ed è stato addestrato su circa 280M di sequenze, condizionando la predizione con informazioni sulla funzione o i componenti cellulari; questo gli ha permesso di catturare pattern evolutivi e strutturali rilevanti. In particolare ProGen riesce a generare varianti della sequenza, o più, in input, ed utilizzata come contesto di partenza, preservando le sue caratteristiche funzionali e strutturali, un aspetto molto importante per la ingegnerizzazione di nuove proteine. Per valutare la qualità generativa del modello viene osservata la similarità sulla struttura primaria, la sequenza, e l'accuratezza della struttura secondaria, o tridimensionale.

Infine, ESM-2 [26] migliora ulteriormente le performance della prima versione del modello; arricchisce inoltre le sue capacità riuscendo a predire le coordinate atomiche della struttura proteica tramite un secondo transformer sopra il primo di ESM, presentandolo come ESMFold.

Capitolo 3

Modellazione ed Implementazione

Ci siamo concentrati sullo sviluppo di un'architettura per affrontare la limitata scalabilità di ProfileView nell'analisi di dataset di grandi dimensioni. Sebbene ProfileView sia efficace nella classificazione di insiemi di migliaia di sequenze, il nostro obiettivo è ampliarne la capacità per gestire dataset dell'ordine di milioni di sequenze.

Vogliamo sfruttare il successo dei modelli di machine learning nell'estrazione e nell'apprendimento di caratteristiche specifiche a partire da dataset complessi. A differenza di approcci algoritmici come ProfileView, i modelli di apprendimento supervisionato possono apprendere direttamente una funzione di discriminazione dai dati in input e sono in grado di identificare relazioni complesse non lineari tra le varie features.

Nel campo del deep learning, le reti neurali migliorano ulteriormente la capacità di astrarre caratteristiche chiave da dati non strutturati, come le sequenze proteiche. La loro capacità di apprendere autonomamente quali pattern riconoscere e come estrarli consente un livello di astrazione superiore e un'identificazione più efficace delle caratteristiche funzionali. Inoltre, i metodi di deep learning migliorano significativamente le prestazioni nella gestione di dataset di grandi dimensioni grazie all'uso della parallelizzazio-

ne e del calcolo su GPU. Negli ultimi anni, un'architettura specifica di rete neurale, il Transformer, ha raggiunto risultati all'avanguardia in compiti di elaborazione del linguaggio naturale (NLP), come la classificazione e la generazione di testo [27]. Large Language Models (LLMs), come quelli descritti da Vaswani et al. e Wei et al. (2022) [**?** 28], sono l'esempio più rilevante di questa architettura.

I Protein Language Models (PLMs) rappresentano un'applicazione diretta degli LLMs al campo delle proteine [29, 30, 31, 32], come dimostrato da modelli come ESM-2 [26], un modello altamente performante ispirato a BERT [33], che guida il settore. Per la nostra architettura, abbiamo scelto di sfruttare la tecnica del Transfer Learning, poiché addestrare un nuovo modello da zero è estremamente oneroso in termini di tempo e risorse, richiedendo dataset di grandi dimensioni, un'elevata capacità computazionale e numerose iterazioni. Inoltre, vogliamo sfruttare la conoscenza delle sequenze proteiche già appresa da ESM-2 durante la sua fase di pre-addestramento.

Nello specifico, per il nostro compito di classificazione funzionale, utilizziamo ESM-2 in due modi: 1. Per convertire le sequenze proteiche ad una rappresentazione interna al modello, che funge da input per la nostra architettura. 2. Come backbone nei vari modelli di deep learning che abbiamo testato.

3.1 Training dataset

Il dataset di addestramento di DomainSpanESM è determinato da ProfileView, che svolge il ruolo di annotatore dei dati per DomainSpanESM (SPIN). Il dataset di addestramento conterrà alcune migliaia di sequenze di domini, classificate da ProfileView in diverse classi. Le sequenze vengono prelevate dal database Uniprot (https://www.uniprot.org) e filtrate in base a un dominio proteico.

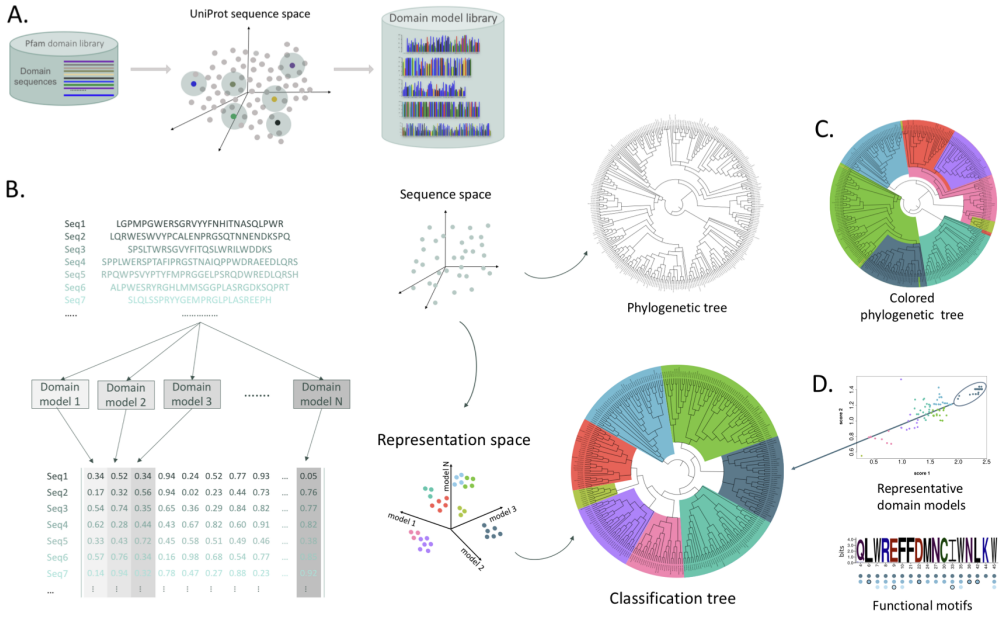

Figura 3.1: **Architettura ProfileView** (figura da [1]): **A**. Libreria dei modelli Hidden Markov Models (HMM) rappresentanti dei dominii della famiglia PFAM. **B**. Sequenze da classificare, punti nel Representation Space, mappate tramite score assegnato da ogni modello in libreria. **C**. Albero filogenetico in cui viene evidenziata ogni sottofamiglia in base alla classificazione di ProfileView. **D**. Albero di classificazione in output in cui si identifica il migliore modello rappresentativo per ogni sottoalbero e il motivo funzionale

3.1.1 ProfileView

ProfileView viene utilizzato per categorizzare gruppi di sequenze simili in base alle loro funzioni, con l'obiettivo di individuare i residui determinanti, ancora caso di studio irrisolto, alla base della diversità funzionale delle proteine e scoprire nuove funzioni biologiche. Prende in input un insieme di sequenze omologhe e un dominio proteico, restituendo una classificazione delle sequenze in sottogruppi funzionali, insieme ai motivi funzionali che caratterizzano ciascun sottogruppo.

La prima idea principale di ProfileView è estrarre schemi conservati, ov-

vero gruppi di residui ritrovati molto frequentemente, dallo spazio delle sequenze disponibili tramite la costruzione di numerosi modelli probabilistici per una famiglia di proteine. Questi modelli sono in grado di campionare la diversità tra sequenze omologhe disponibili e riflettere le differenti caratteristiche strutturali e funzionali, individuate localmente tra sequenze simili alle sequenze campionate (Figura 3.1A). Questi modelli, chiamati Clade-Centered Models (CCM) [34, 35] sono implementati come Hidden Markov Models (HMM) per identificare profili di conservazione tra le sequenze, chiamati anche come HMM Profile. In bioinformatica, il termine "clade" si utilizza spesso per identificare un gruppo di sequenze che condividono la stessa discendenza, per esempio all'interno di un albero evolutivo. Rispetto ai consensus models [36], evitano la perdita di segnali funzionali quando vengono considerate sequenze distanti.

Per costruire i CCM, viene considerato l'intero insieme di sequenze S_i provenienti dal dataset Pfam [37] associate a un dominio D_i. Seguendo la figura 3.1A, per ogni sequenza $s_j \in S_i$ (punti colorati nello spazio), viene eseguita una ricerca di sequenze omologhe, vicine a s_j, su UniProt (insiemi in grigio). Per ognuno di questi insiemi, o Clade, viene costruito un HMM Profile. Il motivo dei clade è per arricchire l'informazione della singola sequenza s_j. Questo modello mostra caratteristiche specifiche, possibilmente relative alla funzione di s_j, che potrebbero differire da altre sequenze $s_k \in Si$ del dominio D_i. Più s_j e s_k sono divergenti, più i CCM dovrebbero evidenziare caratteristiche differenti.

La seconda idea principale di ProfileView è utilizzare i CCM per incorporare le sequenze di input in uno spazio di rappresentazione multidimensionale, dove ogni dimensione è associata a un CCM (Figura 3.1B-D). Per ogni sequenza di input da classificare, ogni modello viene confrontato con la sequenza producendo uno score, che esprime la probabilità che il modello possa generla. Lo score viene utilizzato come feature per la rapprentazione vettoriale della sequenza. Così facendo le sequenze vengono proiettate in questo spazio denominato "spazio funzionale", che le organizza in base alla

loro funzione. Il vettore di embedding associato a una sequenza proteica ha dimensione $2 \times N$, dove N è il numero dei modelli CCM ed il numero di score assegnati: il bitscore e il bitscore medio di ciascun modello HMM.

Infine, ProfileView raggruppa le sequenze, trasformate in vettori, mediante clustering gerarchico considerando la minima varianza di Ward come criterio di selezione. In output viene prodotto un albero di classificazione funzionale. Come illustrato in figura 3.1C, la topologia dell'albero funzionale non corrisponde necessariamente a quella dell'albero filogenetico. Alcuni sottoalberi dell'albero di classificazione saranno associati a modelli probabilistici rappresentativi e motivi funzionali. Infatti, i modelli rappresentativi verranno utilizzati per suddividere i membri della famiglia o della sottofamiglia in gruppi più piccoli, al fine di catturare le differenze nelle caratteristiche funzionali della famiglia o creare gruppi che preferibilmente includano una sola funzione. I motivi sono pattern di residui da cui si analizzano informazioni sulla funzione o la struttura; si riconoscono attraverso l'allineamento di sequenze osservando le sezioni maggiormente conservate, possono essere parti di un dominio come combinazione di residui distanti.

La struttura gerarchica delle classi, ereditata dalla forma ad albero, consente di individuare sotto-alberi come gruppi di sequenze che condividono somiglianze funzionali. Più i sotto-alberi sono profondi, più specifica è la loro funzione proteica e maggiore è la misura di identità, trattandosi di sequenze omologhe.

Attendibilità della classificazione

Per evidenziare le sue capacità e la generalizzazione del modello, ProfileView è stato applicato su sette diverse famiglie di proteine. Le proteine appartenenti a queste famiglie sono distinte da una grande diversità funzionale, molti di questi sono ben annotati funzionalmente e le loro sottofamiglie sono state delineate e validate sperimentalmente insieme ai loro corrispettivi motivi funzionali. Nello specifico osservando l'albero funzionale costruito da ProfileView, si individuano tra le sue foglie le sequenze già annota-

te sperimentalmente. In questo modo si associa alle sequenze dello stesso sotto-albero la stessa funzione già annotata.

Le famiglie sui è stato collaudato ProfileView sono: la Cryptochrome/Photolyase (CPF), i domini WW, la famiglia degli enzimi GH30 e quattro sotto gruppi di proteine appartenenti a due superfamiglie di enzimi, l'Haloacid Dehydrogenase (HAD/β-PGM/Phosphatase) e la Radical SAM. Queste famiglie e sotto-gruppi permettono a ProfileView di dimostrare le sue potenzialità nell'estrazione di informazioni determinanti, oltre alla semplicità dell'interpretazione dei suoi risultati e della metodologia applicata. Numerose sequenze proteiche omologhe sono state così classificate per la prima volta con ProfileView.

3.1.2 Dataset pipeline

Per l'annotazione dei dati tramite ProfileView abbiamo bisogno di due tipologie di sequenze: la prima per la costruzione della libreria dei modelli, la seconda per la costruzione dell'albero funzionale per l'etichettatura delle sequenze (vedi paragrafo 3.1 e figura 3.1A,B).

Il dataset di addestramento di DomainSpanESM è progettato quindi in quattro fasi:

1. **Libreria dei modelli ProfileView:** Abbiamo bisogno delle sequenze dei dominii della famiglia proteica per la costruzione dei CCM. Dal database di domini PFAM, abbiamo scaricato in formato fasta [38] filtrando le sequenze per il loro codice rappresentativo della famiglia a cui appartengono (PF-XXXXX). Per la costruzione della libreria il numero di sequenze, se necessario, viene ridotto tramite clustering, rimozione delle sequenze altamente similari, con MMseqs [39]. Successivamente si esegue la funzionalità "build" di ProfileView che restituisce in output la libreria dei modelli.

2. **Creazione albero funzionale:** Data la grande disponibilità e la facilità di reperimento dei dati abbiamo scelto Uniprot (www.uniprot.org) come database per le sequenze da etichettare. Per la selezione abbiamo

filtrato sempre con il codice PFAM precedente e scaricato i relativi file fasta. Se necessario si applica sempre la riduzione tramite clustering. Infine eseguendo la funzione "tree" di ProfileView, passando in input la libreria dei modelli e le sequenze da classificare, viene stato restituito in output l'albero funzionale in formato Newick [40]; le foglie dell'albero sono per l'appunto le sequenze, mentre i nodi i vari cluster gerarchici. Puntualizziamo che ProfileView elimina le sequenze che ottengono uno score dai CCM inferiore ad un certa soglia durante la creazione dello spazio funzionale, significando che non è stato ritrovato nessun motivo caratterizzante.

3. **Classificazione:** L'albero di ProfileView viene suddiviso in sotto-alberi disgiunti, ciascuno rappresentante una distinta classe funzionale di sequenze. L'utente può scegliere i sotto-alberi sia manualmente sia fissando una soglia sulla profondità dell'albero a partire dalla radice. A ciascuna classe viene assegnato un ID, la label delle sequenze.

4. **Recupero informazioni sul dominio:** (Opzionale) recuperare dal database Uniprot le posizioni di inizio e fine del dominio di ciascuna sequenza, quando presenti.

Si noti che trovare un metodo generale e applicabile per determinare i sotto-alberi non è semplice, come discusso in [41]. Inoltre, l'utente potrebbe essere interessato a esaminare alcuni sotto-alberi specifici manualmente.

Un aspetto finale riguarda il forte squilibrio che i dataset presentano con questo approccio. È quasi impossibile ottenere una distribuzione omogenea delle classi costruendo il dataset come descritto precedentemente. Infatti, gli alberi di ProfileView sono fortemente sbilanciati, con dimensioni delle foglie differenti per ciascun sotto-albero. Abbiamo mitigato questo problema calcolando i pesi inversi della frequenza delle classi durante il training per il calcolo dell'errore con la funzione cross-entropy e delle metriche per la valutazione delle performance.

3.2 Architettura DomainSpanESM

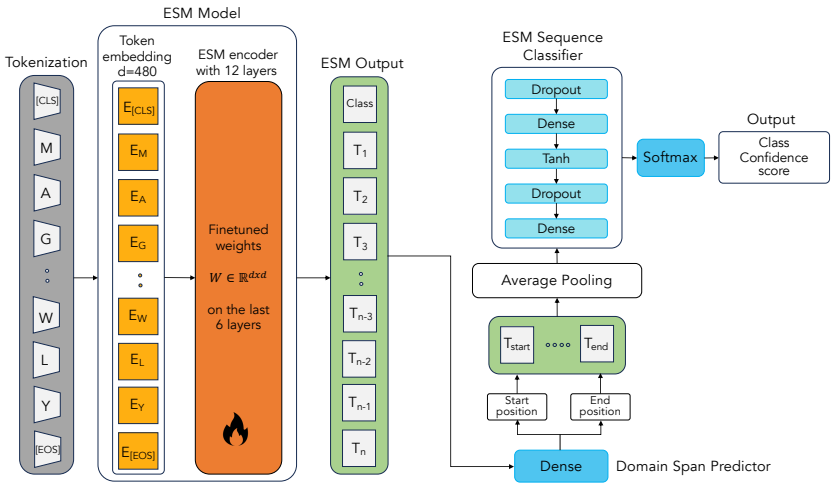

Figura 3.2: **Architettura del modello di DomainSpanESM**.La sequenza, ad esempio MAG...WLY, composta da n residui, viene trasformata in una lista ordinata di token (grigio) e processata da ESM-2 (blu), addestrando gli ultimi 6 layer su 12. I token della sequenza, codificati con ESM-2 tenendo conto del contesto, vengono processati da un Fully Connected layer per prevedere le posizioni di inizio e fine del dominio. Solo sui token selezionati viene applicato averaging pooling, che viene quindi elaborato dal classificatore. DomainSpanESM restitusce in output la classe predetta e un punteggio di confidenza.

Abbiamo esplorato diverse soluzioni per il nostro task di classificazione. La nostra sperimentazione ha spaziato dalle tecniche di machine learning classiche ai modelli di deep learning. Inizialmente, abbiamo valutato gli algoritmi di machine learning tradizionali, che sono stati ampiamente utilizzati per molti compiti di classificazione grazie alla loro interpretabilità e al relativamente basso costo computazionale. Successivamente, spinti dai recenti risultati ottenuti in Bioinformatica con modelli DL nella predizione della struttura proteica [9], è diventato evidente quanto queste tecniche possano apprendere una rappresentazione dei dati ricca di informazioni biologiche.

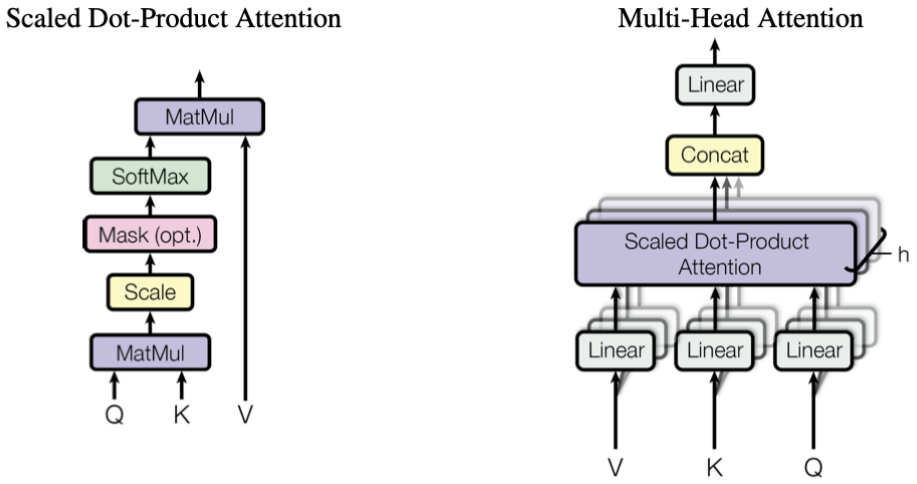

Figura 3.3: Self-Attention e Multi-head Attention, da [2]

La principale preoccupazione nel design dell'architettura di DomainSpanESM riguarda il problema dovuto all'alta probabilità di similarità tra le sequenze omologhe. Come introdotto precedentemente, il successo di ProfileView è legato alla sua capacità di classificare funzionalmente insiemi di sequenze omologhe, estremamente simili tra loro. Esse condividono lo stesso dominio proteico, che è una parte molto conservata della sequenza all'interno di una famiglia proteica.

Per l'architettura del modello di DomainSpanESM, illustrata in Figura 3.2, abbiamo scelto di partire da ESM-2. L'architettura di ESM-2 si basa su Transformer [2], fortemente ispirato a BERT [33], che ha rivoluzionato il settore del Natural Language Processing. In particolare, ESM-2 adotta un approccio di pre-training non supervisionato, simile a quello di BERT, in cui il modello viene addestrato a predire token mascherati all'interno delle sequenze proteiche, dove ogni token rappresenta un amminoacido. Questo metodo consente di costruire una rappresentazione interna dell'informazione ricca e contestuale, capace di catturare le relazioni a lungo raggio tra residui, essenziali per comprendere struttura e funzione delle proteine. Il nucleo dell'architettura risiede nel meccanismo di self-attention, descritto nel cele-

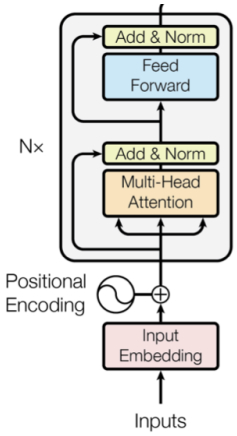

Figura 3.4: Transformer Encoder architecture, da [2]

bre articolo [42], che permette ad ogni token di aver uno peso differente in base all'influenza degli altri token presenti nella sequenza. In ESM-2, tale meccanismo viene implementato attraverso una serie di strati (layer) di encoder Transformer come in figura 3.4, ciascuno dei quali applica parallelamente self-attention (multi-head attention figura 3.3); seguiti da trasformazioni feed-forward, ovvero una combinazione di layer lineare e GeLU [43].

La nostra proposta si differenzia implementando un metodo ispirato al Question Answering task utilizzato su BERT [33]. Questo approccio si concentra sulla previsione dell'intervallo del dominio all'interno di una sequenza per isolare e classificare solo le parti rilevanti. Inizialmente ogni sequenza viene suddivisa a livello dell'amminoacido e rappresentata come una serie finita di token tramite il tokenizer associato a ESM-2, ad ogni amminoacido corrisponde un valore intero da 4 a 28, 0 ed 1 per il classification (cls) e pad token rispettivamente. Nel nostro caso la classificazione non è stata fatta sul cls, ma su una rappresentazione media della sequenza. Successivamente l'encoder di ESM-2 genera un embedding per ciascun token che cattura il contesto su tutta la sequenza. Gli embedding vengono quindi inviati ad un layer lineare (Span Dense Predictor in figura 3.2) per prevedere le posizioni di inizio e fine dell'intervallo. Questo restituisce due logits corrispondenti all'inizio e alla fine dell'intervallo, che vengono separati lungo le dimensioni

interne per ottenere predizioni separate per ciascuno. Infine, solo gli embedding all'interno dell'intervallo vengono estratti e approssimati tramite valore medio lungo le dimensioni interne per creare una rappresentazione vettoriale unidimensionale della sequenza, che viene inviata al classificatore in testa per il calcolo degli score delle classi.

Abbiamo selezionato la versione pre-addestrata del modello ESM-2 da HuggingFace (`https://huggingface.co`) utilizzando PyTorch (`https://pytorch.org`) come framework di DL. Questo modello ha 35 milioni di parametri e 12 Encoder layers. Il fine-tuning del modello è stato eseguito congelando solo i primi 6 layer per mantenere la conoscenza appresa sulle features a basso livello. I layer finali, essendo più specializzati e specifici per il task, si concentrano sull'acquisizione delle caratteristiche rilevanti per il task specificato. Anche la nostra capacità di memoria VRAM ha inficiato nel numero di layer che siamo riusciti ad addestrare. Poiché le sequenze proteiche variano significativamente in lunghezza, il preprocessing è essenziale. Inoltre, l'uso della memoria del meccanismo di attenzione cresce in modo quadratico con la lunghezza della sequenza, rendendo questo un aspetto importante da considerare. Si noti che la previsione dell'intervallo del dominio è opzionale; durante l'addestramento, se le posizioni di inizio e fine non sono disponibili, si considera l'intera sequenza per la rappresentazione finale della sequenza. In tal caso, l'architettura è identica a ESM-2, ovvero un Encoder con classificatore in cima.

In inferenza DomainSpanESM accetta una o più sequenze come input, che possono essere fornite come file CSV o FASTA, o come una lista di stringhe. La descrizione di ciascuna sequenza nel file CSV comprende: l'ID della sequenza, la sequenza (come stringa), la posizione di inizio e fine dell'intervallo del dominio (numeri interi) e un'etichetta (ID del sotto-albero) per la fase di addestramento. Il file FASTA o una lista di sequenze, passate come stringhe, sono accettati esclusivamente per prevedere la classe. Se viene utilizzato un file CSV per l'inferenza, due colonne aggiuntive vengono inserite per la classe prevista e il relativo punteggio di probabilità.

Capitolo 4

Addestramento

Il dataset di training per DomainSpanESM è composto da sequenze omologhe provenienti da un'unica famiglia di proteine; ogni sample è etichettato per funzione, il loro numero dipende direttamente dalla conoscenza che abbiamo della famiglia e dalla specificità del livello di annotazione che vogliamo. Nel nostro caso questo dipende da ProfileView. Durante l'addestramento, abbiamo condotto un approccio grid-search per la ricerca degli iperparametri al fine di mitigare l'overfitting, che rappresentava una preoccupazione principale a causa dello sbilanciamento nel dataset. Abbiamo utilizzato il set di validation per misurare e confrontare le performance.

I risultati sono riportati utilizzando come caso di test la famiglia di proteine TRX, che utilizziamo in questo studio come approccio generale applicabile a tutte le famiglie proteiche.

Per DomainSpanESM abbiamo considerato 16 combinazioni di questi parametri:

- **Learning rate:** $1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$
- **Optimizer:** Adam, AdamW.
- **Weight decay:** 0.0, 0.01.
- **Scheduler:** Nessuno, Coseno con warmup.

La configurazione ottimale utilizza un learning rate di $1 \cdot 10^{-4}$, l'ottimizzatore AdamW con un weight decay di 0.01, un cosine learning rate scheduler con warmup i cui step sono impostati al 10% dei training step totali, early stopping con pazienza di 2 epoche sul F1 del validation. La funzione multi-loss sviluppata combina la cross-entropy per la classificazione e la previsione dell'intervallo, con un peso maggiore per la classificazione, pari al 70%:

$$\mathcal{L} = 0.7 \cdot \mathcal{L}_{\text{class}} + 0.3 \cdot \mathcal{L}_{\text{interval}}$$

,

$$\mathcal{L}_{\text{class}} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) w_i$$

. Questa scelta è in linea con l'obiettivo principale del compito, che è la classificazione. Un bilanciamento più equilibrato avrebbe penalizzato tale obiettivo. Per mitigare lo sbilanciamento del dataset abbiamo deciso di calcolare differenti pesi w_i per ogni classe, con valori inversamente proporzionali al numero di sequenze per classe. L'errore per la predizione dello span è la media tra il token di inizio e di fine:

$$\mathcal{L}_{\text{interval}} = \frac{1}{2}(\mathcal{L}_{\text{start}} + \mathcal{L}_{\text{end}})$$

dove $\mathcal{L}_{\text{start}}$ e $\mathcal{L}_{\text{end}}$ rappresentano sempre la funzione di cross-entropy.

L'addestramento è stato condotto per 10 epoche utilizzando tre seeds casuali diversi per valutare l'impatto delle fluttuazioni randomiche. Raccogliendo le performance per ogni epoca, abbiamo salvato i pesi del modello tenendo conto dello score di F1 maggiore, e al termine abbiamo selezionato il seed con il punteggio migliore.

La dimensione della batch è di 8 elementi per il set di addestramento, mentre 32 per la validazione e il test. Ogni epoca ha richiesto circa 25 minuti per essere completata.

Questo set di parametri è stato utilizzato anche per l'addestramento del modello sul caso della famiglia proteica CPF.

Capitolo 5

Esperimenti

Per validare la scelta architetturale e le performance di DomainSpanESM per la task di classificazione, è stata effettuata un'analisi comparativa con altre tipologie di modelli: tecniche tradizionali di ML e modelli moderni di DL. Due famiglie di proteine sono state prese in considerazione, la TRX e CPF; la prima è stata scelta come metro di giudizio per la validazione di ogni esperimento fatto, mentre la seconda come conferma della bontà del modello e in quanto è stata analizzata anche in ProfileView [1]. Parallelamente è stata portata avanti un'analisi sulle classi identificate per la TRX nella ricerca dei potenziali residui funzionali caratteristici per ognuna di esse.

5.1 Dati

Seguendo la pipeline presentata nel paragrafo 3.1.2:

1. Dal database di domini PFAM abbiamo scaricato in formato fasta filtrando le sequenze utilizzando il codice PF00085 per la TRX e PF00035 per CPF. Successivamente mediante clustering abbiamo ridotto del 50% il numero totale, per la CPF questo processo non è stato necessario in quanto la libreria è già fornita da ProfileView. Successivamente e soltanto per la TRX, abbiamo eseguito la funzionalità "build" di Profi-

leView che ha restituito in output la libreria dei modelli TRX. In totale sono stati generati 2827 modelli per la TRX mentre 4615 per la CPF.

2. Le sequenze per la classificazione scaricate da Uniprot sono in totale per la TRX 232500 e 44984 per la CPF, a cui è applicata sempre la riduzione tramite clustering portando le sequenze a 29000 e 20941, rispettivamente. Dopo la costruzione dell'albero con la funzionalità "tree", per la TRX il numero di sequenze rimanenti sono 28601 mentre per la CPF a 14295. Infine, anche il numero dei modelli viene ridotto tramite PCA, 202 per la TRX e 240 per la CPF.

3. Per entrambe le famiglie proteiche è stato scelto di individuare le classi funzionali ad altezza 3 dell'albero, in questo modo i cluster sono più ampi e le sequenze meno simili aumentando il grado di difficoltà di classificazione per DomainSpanESM.

4. Da Uniprot sono state scaricate le informazioni riguardanti l'inizio e la fine del dominio delle sequenze per l'addestramento alla predizione di DomainSpanESM, quando presenti.

Le prestazioni di DomainSpanESM sono state valutate quindi su 28.601 sequenze provenienti dalla famiglia proteica TRX ed etichettate da ProfileView in otto classi distinte come in figura 5.1, utilizzate come ground-truth. Le otto classi specifiche per funzione sono state identificate innanzitutto raggruppando le sequenze nello spazio multidimensionale definito da ProfileView, visibile in figura 5.2 in alto. La vicinanza delle sequenze in questo spazio è rappresentata nella figura 5.3A, che mostra un albero generato tramite clustering gerarchico. Le otto classi funzionali corrispondono agli otto sottoalberi individuati ad una profondità di 3 nell'albero. DomainSpanESM è stato addestrato sull'80% del set di sequenze, pari a 22.880 sequenze, e valutato sulle restanti 5.721 sequenze. La distribuzione delle sequenze nelle otto classi è mantenuta nella suddivisione.

Per la famiglia delle CPF è stato seguito lo stesso identico approccio, le statistiche sono riportate in tabella 5.1 destra.

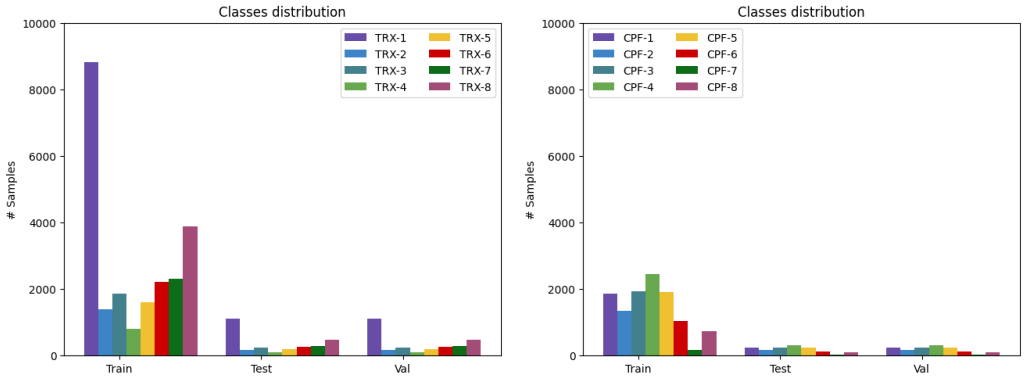

Figura 5.1: DomainSpanESM TRX e CPF dataset. L'istogramma descrive la distribuzione delle 8 classi di sequenze TRX (sinistra) e CPF (destra) per ogni set utilizzato: train, validation e test.

5.2 Metriche

Per la valutazione e la selezione della migliore architettura, è stato seguito un approccio standard che ha coinvolto il tuning degli iperparametri e l'aggregazione delle metriche di performance attraverso tre diversi seed, in particolare per i modelli influenzati dalle fluttuazioni casuali in fase di istanziamento, per esempio i modelli di DL.

Per valutare le prestazioni, definiamo una ground-truth di riferimento utilizzando i dataset di validation e test, e categorizziamo i risultati come segue: true positive (TP): etichette funzionali conosciute che sono identificate da DomainSpanESM; falsi positivi (FP): etichette identificate da DomainSpanESM che non sono conosciute; falsi negativi (FN): etichette che non vengono trovate da DomainSpanESM ma sono conosciute; veri negativi (TN): etichette che non sono conosciute e non vengono identificate da DomainSpanESM. Considerando lo sbilanciamento delle etichette, utilizziamo due metriche standard di valutazione delle prestazioni: accuratezza e score F1. Entrambe le metriche sono calcolate assegnando un peso ai campioni, bilanciato in base alla frequenza inversa delle classi.

- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

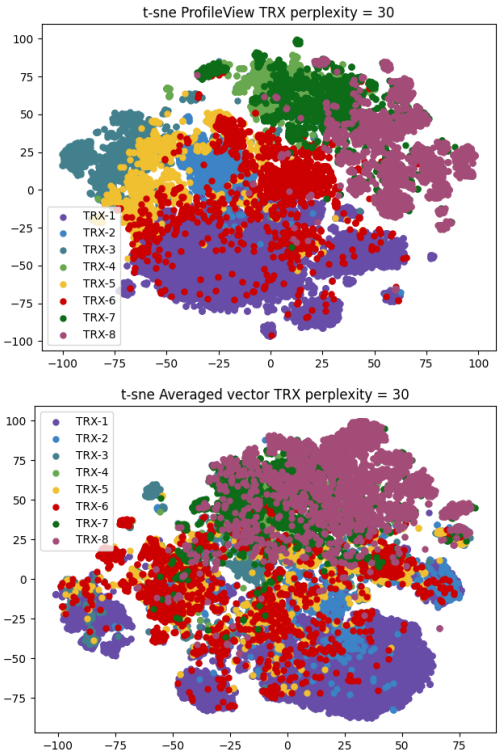

Figura 5.2: Rappresentazione visiva delle 28601 sequenze TRX, utilizzate per addestrare DomainSpanESM, con gli embedding prodotti da ProfileView (sopra) e ESM-2 (sotto)

- F1 score $= 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$

La scelta della configurazione migliore viene fatta considerando l'F1 score più alto. Il motivo della scelta è dovuto all'equilibrio che la formula assicura tra le metriche di Precision ($\frac{TP}{TP+FP}$) e Recall ($\frac{TP}{TP+FN}$), la prima indica la proporzione di istanze correttamente identificate come positive rispetto a tutte quelle classificate come positive, mentre la seconda la capacità del modello di identificare tutte le istanze positive effettive. Un valore alto di F1 è dovuto quindi a valori elevati per entrambi, in questo modo evitiamo di penalizzare una rispetto all'altra.

5.3 Analisi comparativa

Abbiamo confrontato l'architettura di DomainSpanESM con una serie di tecniche esistenti per compiti di classificazione, includendo sia tecniche di Machine Learning tradizionali che modelli di Deep Learning moderni.

Una differenza tra il nostro approccio ai modelli di Machine Learning (ML) e Deep Learning (DL): per i primi abbiamo eseguito una cross-validation con 5-Fold, mentre per i modelli DL abbiamo separato i parametri architetturali da quelli di addestramento, a causa dell'elevato numero di combinazioni possibili e dei tempi di addestramento prolungati, portando quindi a risultati sub-ottimali. In particolare, abbiamo testato diversi tipi di ottimizzatori specifici per ciascuna architettura. Abbiamo riscontrato che SGD ha dato buoni risultati per le architetture basate su CNN, mentre Adam e AdamW sono stati più efficaci per il fine-tuning di ESM-2 e LSTM. Ogni addestramento è stato eseguito per 5 epoche durante la grid search, mentre per 10 epoche durante la valutazione dei semi casuali.

5.3.1 Rappresentazione dei dati

La rappresentazione dei dati è cruciale per il successo dei modelli di machine learning nell'apprendere le caratteristiche più rilevanti. Abbiamo distinto due modalità di embedding delle nostre sequenze, valutando come le diverse architetture performano con ciascuna di esse. La prima, più semplice e diretta, è il one-hot encoding di ciascun amminoacido della sequenza. Un vocabolario degli amminoacidi noti viene estratto dai dati di addestramento, considerando anche il caso degli amminoacidi sconosciuti, per un totale di 23. Il vettore avrà così valore uguale a 1. nella posizione corrispondente all'indice dell'amminoacido nel vocabolario, e valore 0. in tutte le altre. Così per tutti gli amminoacidi della sequenza. Questa rappresentazione è stata adoperata dalle architettura CNN e BiLSTM (vedi paragrafi 5.3.3 e 5.3.3) per esaminare le loro performance di apprendimento da dati puramente grezzi senza nessuna informazione già codificata come per gli embeddings di ESM-2, spinti

anche dal fatto che a differenza del linguaggio naturale il vocabolario è molto ridotto. Allo stesso modo per tutte le architettetture testate, anche per CNN e BiLSTM, abbiamo utilizzato una rappresentazione contestualizzata tramite Transformers come ESM-2, dove lo stesso amminoacido può essere rappresentato in modo diverso a seconda del suo contesto, portando a rappresentazioni molto più ricche e significative rispetto ai metodi statici precedenti.

5.3.2 Algoritmi Machine Learning

Gli algoritmi di ML tradizionali, noti per la loro interpretabilità e il basso costo computazionale, hanno avuto ampio successo in molte attività di classificazione. Abbiamo sperimentato con Support Vector Machine (SVM), Gradient Boosting, Random Forest e K-Nearest Neighbors (K-NN) per valutare fino a che livello questi metodi potessero rilevare pattern specifici delle classi nelle nostre sequenze. Abbiamo utilizzato Scikit-learn (www.scikit-learn.org) come framework per l'implementazione e addestramento dei modelli.

Support Vector Machine

Abbiamo adoperato il modello Support Vector Classifier di Scikit che implementa Support Vector Machines [44] per la classificazione supervisionata. Utilizza algoritmi per la ricerca di un iperpiano ottimale in grado di separare le diverse classi dei dati, massimizzando inoltre il margine di separazione tra essi. Nel caso in cui i dati non sono linearmente separabili nello spazio, le SVM utilizzano funzioni kernel per mappare i dati in uno spazio a dimensione superiore, dove è più probabile trovare una separazione lineare. Abbiamo testato diversi tipi di kernel: lineare, polinomiale e RBF; oltre a diversi valori per il parametro di tolleranza agli errori C (5,10,15,20,30), dovuti dal trade-off nella ricerca dell'iperpiano che ammette appunto misclassificazioni. La configurazione ottimale è composta dal kernel RBF con C uguale a 5. Risultati in tabella 6.2.

k-Nearest Neighbor

k-NN [45] è un algoritmo di apprendimento supervisionato che classifica un dato osservando i suoi vicini più prossimi. Particolarità è che non richiede una fase di addestramento esplicita, in quanto utilizza direttamente i dati presenti per effettuare le predizioni. I dati sono rappresentati come punti in uno spazio multidimensionale, dove ogni dimensione corrisponde a una feature. La distanza per determinare la vicinanza tra i punti viene calcolata tramite funzione Euclidea, Coseno o altre. Oltre alla metrica per la distanza un altro parametro fondamentale è il valore di k, che determina quanti punti considerare come vicini rispetto ad un dato target, quindi la sensibilità del modello nella classificazione. Infatti un valore troppo alto generalizza meggiormente appiattendo le differenza tra i dati, viceversa un valore troppo basso diventa troppo soggetto alle piccole variazioni. Come unici parametri abbiamo preso in considerazioni k, con valori tra 5 e 100, e la metrica di distanza tra funzione coseno, euclidea e minkowski. La configurazione migliore ha k uguale a 5 con distanza euclidea.

Random Forest

Random Forest [46] è un algortmo di ML basato sull'utilizzo di alberi decisionali per la classificazione dei dati. Una foresta è un insieme di alberi indipendenti, dove ognuno viene addestrato su un insieme casuale del dataset, processo noto come bagging. All'interno di ogni albero per il processo di split vengono scelte una o più variabili appartenenti ai dati: la qualità della scelta viene misurata da criteri che massimizzano la corretta suddivisione per classe tra gli insiemi, per esempio Entropia o coefficiente di Gini. Nelle foreste in più viene considerato un sottoinsieme casuale delle variabili, riducendo così la correlazione tra gli alberi. La predizione finale si ottiene aggregando i risultati di tutti gli alberi (mediante score di maggioranza o media). Questo metodo aiuta a ridurre il problema dell'overfitting tipico degli alberi decisionali. Come parametri abbiamo considerato per il calcolo della qualità degli split le funzioni Gini, Entropia e Log Loss; Per il numero

di alberi abbiamo spaziato tra 100 e 1000, con valori di profondità massima tra 10 e 100. Infine abbiamo valutato se utilizzare bagging o meno, quindi se utilizzare l'intero dataset per ogni albero. La configurazione ottimale misura la qualità dello split tramite coefficiente di Gini, con un numero di alberi pari a 75 e profondità massima 10, senza bagging.

Gradient Boosting

Sempre dalla libreria di Scikit abbiamo utilizzato Gradient Boosting Classifier, un algoritmo di boosting [47] che costruisce modelli di alberi decisionali in sequenza. Ad ogni iterazione, il modello corregge gli errori del modello precedente mediante l'ottimizzazione della funzione di loss, in questo caso cross-entropy . In più utilizza la tecnica del gradient descent per aggiornare i pesi associati alle features selezionate per i vari split negli alberi, migliorando la capacità predittiva complessiva. I parametri importanti da individuare sono il numero degli estimator (passaggi di boosting totali), abbiamo testato valori di 25 e 50; il numero massimo di nodi per ogni albero, con valori di 3,5 e 8; infine il learning rate a 0.05 e 0.01. La migliore configurazione individuata ha un numero di estimatore pari a 50, numero massimo di nodi uguale 5 e learning rate 0.01. Le metriche sui vari set in tabella 6.2.

5.3.3 Algoritmi Deep Learning

Dal lato DL, abbiamo valutato architetture che sfruttano la natura sequenziale dei dati per la classificazione, tra cui modelli Transformers, reti Long Short-Term Memory bidirezionali (BiLSTM) e modelli Convolutional Neural Networks (CNN).

Architettura Convoluzionale (1-D CNN)

Le CNN [48] sono efficaci nel rilevare pattern locali e motivi aggregando le informazioni dai token vicini utilizzando finestre mobili (kernel). La nostra architettura CNN è ispirata da [4] che ha ottenuto ottimi risultati

in vari compiti di classificazione per il linguaggio naturale, gestendo efficacemente le lunghezze variabili delle sequenze. L'architettura è composta da un solo layer convoluzionale unidimensionale (1-D CNN) con diversi filtri e dimensioni differenti: 3, 5 e 7 nel nostro caso; ottenendo così diverse features map, rappresentazione in figura 5.4. Successivamente, viene applicato un max-over-time pooling su tutti i filtri, concatenando tutte le features estratte. Infine, il vettore di features di lunghezza fissa risultante viene fornito come input ad un layer lineare preceduto da dropout per l'estrazione dei logits da classificare. In questa architettura abbiamo sperimentato diversi valori per la dimensione interna: 64,128; diversi valori di dropout: 0.1 e 0.2; infine diversi blocchi convoluzionali: 2,8,16. La configurazione migliore comprende dropout a 0.1, dimensione interna uguale a 64 e numero di blocchi convoluzionali 8.

Architettura Long-Short Term Memory (BiLSTM)

Le LSTM [49], invece, riescono a catturare dipendenze a lungo raggio all'interno della sequenza, una caratteristica importante poiché le funzioni delle proteine sono spesso determinate da residui che sono distanti nella struttura primaria (sequenza) ma vicini spazialmente nella struttura ripiegata (struttura secondaria); infatti i siti di interazione funzionali, dove hanno contatto due proteine, possono essere composti da amminoacidi molto distanti nella sequenza. La differenza con le architetture RNN classiche è l'utilizzo di una cella di memoria (Memory Cell), la quale è controllata da tre porte (gate) che decidono quali informazioni aggiungere, rimuovere o restituire dalla cella di memoria:

- **Input gate:** controlla quali informazioni verranno aggiunte alla cella di memoria.

- **Forget gate:** al contrario determina quali informazioni vengono rimosse dalla cella.

- **Output gate:** decide quali informazioni vengono restituire dalla cella di memoria.

Questo permette alle reti LSTM, figura 5.5 sinistra, di selezionare le informazioni rilevanti durante il processamento, consentendo di mantenere caratteristiche importanti anche a lungo termine, cosa non possibile precendentemente. Data la sua natura ricorrente lo stato corrente (h_t) funziona come memoria breve termine, dove questa viene aggiornata tramite la combinazione dell'input corrente, lo stato precedente (h_{t-1}) e lo stato attuale della cella di memoria. La nostra architettura BiLSTM cattura le dipendenze in entrambe le direzioni (forward e backward), vedi figura 5.5 destra. Il design assicura che la concatenazione degli stati interni in ogni step temporale della sequenza sia il valore medio lungo le dimensioni interne, terminando con due layer lineari con dropout per la classificazione. Per BiLSTM abbiamo sperimentato i seguenti parametri architetturali: dimensione interna con 64,128 e 256 con 2,4 o 8 layer bidirezionali. La configurazione migliore utilizza 4 layer con dimensione interna a 256.

Architettura Transformer

Tra i modelli più complessi, abbiamo valutato i Transformers e conseguenti architetture ibride, composte con CNN o BiLSTM. Infine, abbiamo anche considerato l'approccio "domain span", come descritto nella sezione dell'architettura di DomainSpanESM 3.2.

Con Transformer, abbiamo definito innanzitutto una baseline per la classificazione delle sequenze, per confrontare tutti i successivi miglioramenti che abbiamo testato. ESM-2 è servito come backbone dell'architettura, in testa al quale è stato costruito un classificatore, costituito da una combinazione di layer: Dropout, Lineare, attivazione Tanh, Dropout e un layer Lineare finale. Da HuggingFace (https://huggingface.co), abbiamo selezionato la versione del modello ESM-2 con 35 milioni di parametri, 12 Encoder layer e dimensione interna di 480. Il modello è stato testato sia congelato che

finetuned, allenando solo gli ultimi 6 layer. Dopo il passaggio nell'Encoder, la sequenza è rappresentata come la media lungo le dimensioni interne.

Successivamente sono stati eseguiti ulteriori test con ESM-2, sempre come backbone delle prime due architetture presentate sopra (CNN e BiLSTM), e per DomainSpanESM.

5.4 Scoperta dei Motivi TRX ed estrazione dei determinanti funzionali

Il motivo principale di questa analisi è la ricerca dei residui che sono caratteristici del gruppo funzionale di ciascuna classe, anche denominati come determinanti funzionali. Tramite il confronto dei vari domini, estratti attraverso l'allineamento multiplo delle sequenze di ogni sotto-albero, è stato possibile individuare quei residui altamente conservati che si differenziano tra le varie classi; candidati quindi con grande probabilità a determinare la funzione della proteina.

Nel dettaglio, Mafft (www.mafft.cbrc.jp/, versione 7.526) è stato utilizzato per la matrice di allineamento multiplo delle sequenze in 2 fasi:

1. Allineamento per la ricerca locale del dominio (–local-pair flag) all'interno delle sequenze appartenente allo stesso sotto-albero (figura 5.3A)

2. Allineamento globale di tutti i precedenti allineamenti tramite l'operazione di merge (–merge) per il confronto, risultato finale in figura 5.3B

Le posizioni più conservate nella matrice sono state calcolate ed estratte per ciascun sotto-gruppo mediante il calcolo dell'Entropia. Infine, la grafica dei motivi è stata realizzata tramite la libreria WebLogo (https://weblogo.berkeley.edu). Tramite differenza dei vari motivi si sono indivituati i potenziali determinanti funzionali.

Per la visualizzazione delle posizioni conservate, ed in particolare dei determinanti funzionali, nella struttura secondaria, è stato utilizzato PyMOL (https://www.pymol.org), un tool per la modellazione 3D delle proteine. Il procedimento è stato il sequente:

1. **Estrazione delle sequenza rappresentativa del sotto-albero**: Ad ogni posizione viene scelto il residuo più conservato nella matrice di allineamento, questa sequenza viene denominata "sequenza di consenso" (consensus sequence).

2. **Predizione della struttura secondaria**: La varie strutture secondarie delle consensus sequences viene predetta tramite Alphafold [9] in formato PDB.

3. **Visualizzazione in PyMOL** I vari file PDB sono poi visualizzati tramite il software PyMOL (https://www.pymol.org) dove è possibile evidenziare e localizzare i residui e le eliche, risultato in figura 5.3A lungo il contorno dell'albero, ed in E,F.

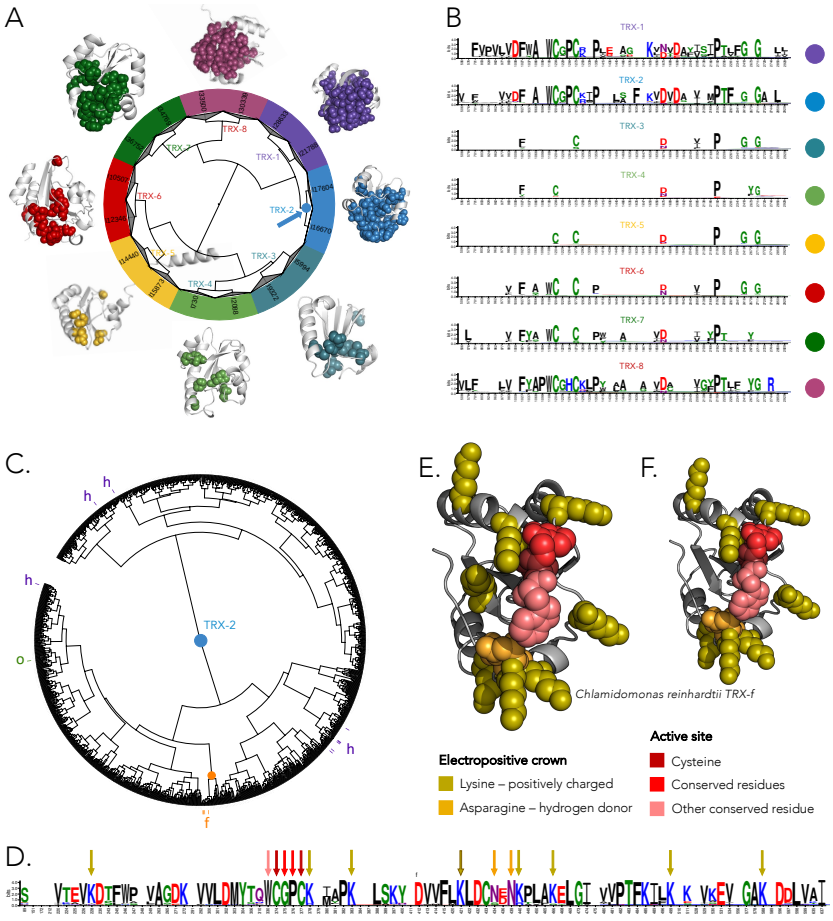

Figura 5.3: **Analisi di ProfileView sulla famiglia TRX, con un focus sulla tioredossina di tipo f, TRX-f2 e sottofamiglie**. **A.** Albero di ProfileView TRX. L'albero identifica 8 differenti sottoclassi a profondità 3. L'albero è stato disegnato utilizzato iTOL (www.itol.emble.de). **B.** I motivi funzionali calcolati tramite allineamento multiplo delle sequenze nei sottoalberi. **C.** Sotto-albero corrispondente alla classe TRX-2 (blu). In arancione sono identificate le posizioni delle sequenze di tipo funzionale f, le sequenze TRX-f2 provengono dall'organismo *Chlamidomonas reinhardtii*. **D.** Il motivo calcolato del sotto-albero TRX-2 in C. **E.** Struttura tridimensionale TRX-f2 (PDB 6i1c) riporta i residui appartanenti ai determinanti funzionali della sequenza *Chlamidomonas reinhardtii*: Il motivo esteso del legame disolfuro (WCGPC) è evidenziato in tonalità di rosso, mentre una corona carica positivamente che circonda il sito attivo comprende otto lisine (in verde oliva) e due asparagine (in arancione). Tutti gli altri residui sono lasciati in grigio.. **F.** Residui noti per svolgere un ruolo funzionale importante per TRX-f2 in [3]. Oltre al motivo esteso WCGPC (in tonalità di rosso), sono presenti due asparagine (in arancione) e sette lisine (in verde oliva), una in meno rispetto a quanto rilevato da ProfileView.

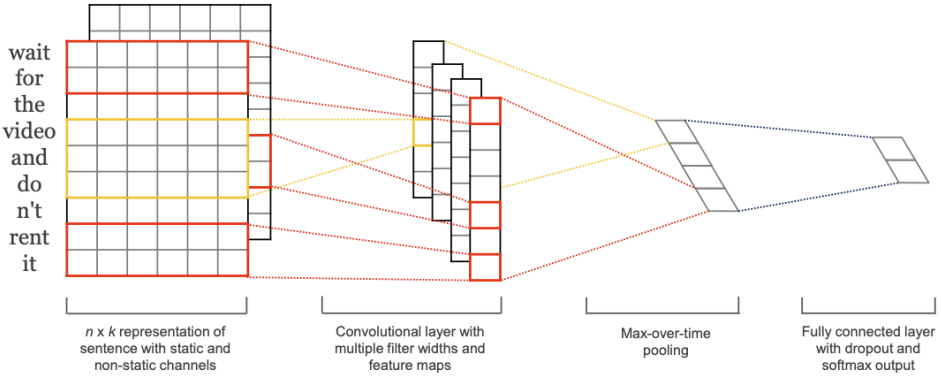

Figura 5.4: Architettura Convoluzionale utilizzata da [4]

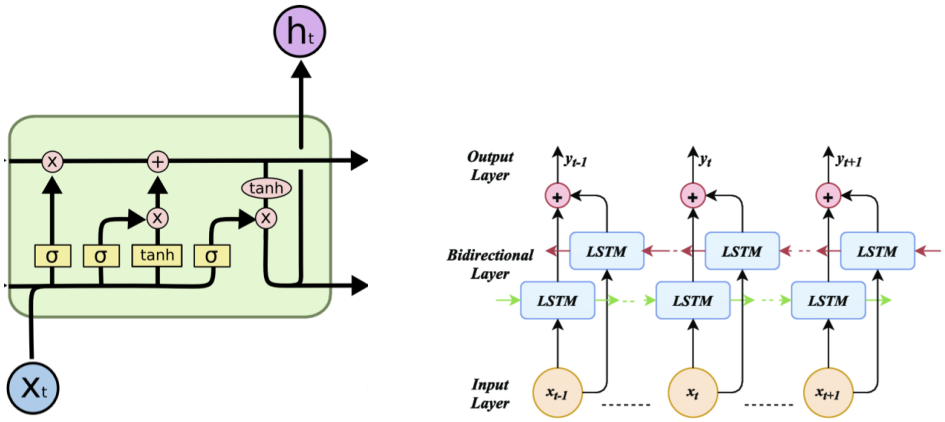

Figura 5.5: Architettura LSTM (sinistra) da Colah's blog ed applicazione Bidirezionale (destra) da [5]

Capitolo 6

Risultati

In questo capitolo presento i risultati ottenuti dagli esperimenti introdotti precedentemente, in particolare riguardo l'analisi comparativa dei modelli sperimentati sul dataset TRX, le performance dell'addestramento di DomainSpanESM su entrambi i dataset TRX e CPF, insieme ad un confronto con le prestazioni di ProfileView. Infine un'indagine sui residui caratteristici della funzione delle diverse classi TRX, denominati anche determinanti funzionali, insieme ad uno studio recentemente pubblicato.

In tabella 6.2 i risultati dell'analisi comparativa dove è subito chiaro come DomainSpanESM raggiunge le performance migliori in entrambe le metriche di accuratezza e F1. In tabella 6.1 invece i risultati per le due famiglie di proteine TRX e CPF, dove CPF ottiene i punteggi migliori, testimoniando probabilmente una migliore discriminazione dei sotto-gruppi funzionali da parte di ProfileView.

Nei grafici 6.26.1 sono riportate le metriche di addestramento per il seed più performante. Ricordo che la ricerca degli iperparametri, tra cui il seed, è stato svolto sulla TRX, e riutilizzati con la CPF. Per il dataset TRX, l'epoca con il F1 score più alto è la ottava, mentre per il dataset CPF il miglior F1 score si registra alla quarta epoca. In quest'ultimo caso, l'addestramento è stato interrotto alla settima epoca a causa dell'assenza di miglioramenti, superando così il valore di pazienza impostato per l'early stopping. Per

la TRX inoltre è evidente la presenza di overfitting rispetto alla CPF, che raggiunge punteggi sia sul train che sul validation prossimi al 100%.

Per un'analisi più dettagliata abbiamo preso in considerazione la matrice di confusione (confusion matrix) in figura 6.3. Lungo la diagonale i valori di recall, ovvero la frazione di sequenze rilevanti correttamente identificate per quella classe. Come ci aspettavamo, lo sbilanciamento dei due dataset utilizzati durante l'addestramento influenza le prestazioni del modello, con le classi meno rappresentate che ottengono valori di recall più bassi. Questi risultati derivano dal fatto che i metodi di deep learning tendono ad adattarsi e specializzarsi verso le classi più rappresentative, poiché vengono osservate più frequentemente, alterando le statistiche. Questo effetto persiste anche quando si applicano pesi di classe con frequenza inversa alla funzione di loss durante l'addestramento.

Per la previsione delle posizione di inizio e fine dello span del dominio, sono stati raggiunti punteggi di accuratezza molto soddisfacenti consentendo una tolleranza di ± 3 posizioni ai bordi del dominio. Senza questa tolleranza, le prestazioni diminuiscono significativamente. Per esempio nel caso della TRX con punteggi di accuratezza di 0,47 e 0,60 rispettivamente nelle posizioni di inizio e fine.

In generale, l'overfitting è un problema comune per tutti i modelli testati, di cui una causa certamente è il forte sbilanciamento presente nei due dataset. Come previsto, le limitazioni dei modelli di Machine Learning (ML) nella selezione autonoma delle features rilevanti per la classificazione influenzano negativamente le loro statistiche. Soltanto SVM riesce a raggiungere punteggi oltre l'80% avvicinandosi ai modelli di DL. Nel nostro caso di test, abbiamo confrontato lo spazio di rappresentazione delle sequenze utilizzando diversi tipi di embedding: ProfileView ed ESM-2 (figura 5.2. I risultati mostrano che ProfileView offre una rappresentazione più distinta, evidenziando le difficoltà di questi modelli nel discriminare tra le classi con gli embedding provenienti da ESM-2. Sorprendentemente, la codifica one-hot fornisce una rappresentazione robusta e semplice per le architetture CNN, ottenendo

	\bf ProfileView		\bf DomainSpanESM							
	Sequences	Classes	Training (80%)		Validation (10%)		Test (10%)		Test Span Accuracy (± 3)	
			Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Start	End
TRX	28601	8	0.957 ± 0.03	0.957 ± 0.03	0.906 ± 0.01	0.906 ± 0.01	0.895 ± 0.01	0.895 ± 0.01	0.823	0.819
CPF	14295	8	0.987 ± 0.004	0.987 ± 0.004	0.982 ± 0.002	0.982 ± 0.002	0.976 ± 0.004	0.976 ± 0.004	0.963	0.909

Tabella 6.1: Resultati training e testing di DomainSpanESM sui dataset di classificazione di ProfileView per le famiglie TRX e CPF. Performance sull'accuratezza della predizione dei residui di inizio e fine con tolleranza di errore 3

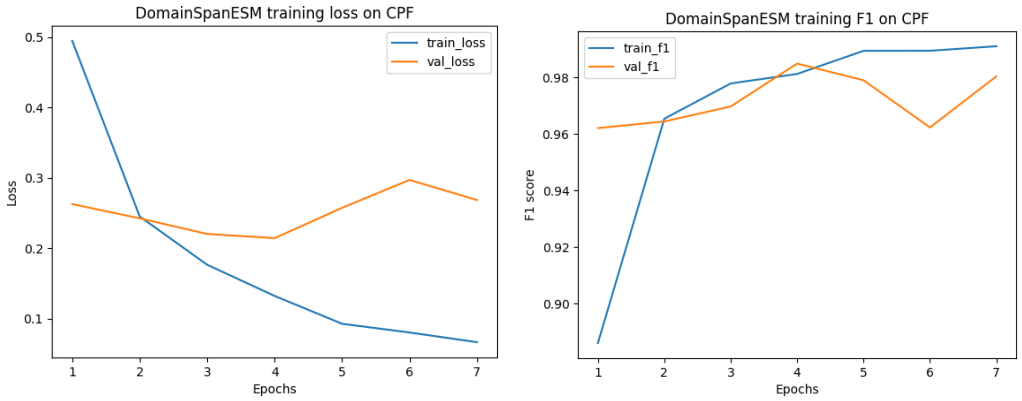

Figura 6.1: Loss (sinistra) e F1 score (destra) durante addestramento DomainSpanESM per il train e validation set CPF, sul seed migliore

ottime metriche di performance.

Per quanto riguarda ESM-2, sospettavamo che il fine-tuning dei layer dell'encoder ESM-2 avrebbe portato a risultati ancora migliori. Inaspettatamente, il modello ESM con convoluzioni non ha mostrato miglioramenti dopo il fine-tuning. L'operazione di smoothing dovuta alla convoluzione potrebbe aver causato una perdita di informazioni importanti sulle caratteristiche apprese. Al contrario, il modello ESM di base ha tratto il massimo beneficio dal fine-tuning. Per questo motivo è risultato chiaro che quest'ultimo è cruciale per ottenere migliori risultati, quindi non abbiamo considerato una versione congelata del modello Domain Span. Quest'ultimo infatti ha raggiunto le migliori statistiche complessive, sebbene molto vicino ai suoi omologhi LSTM e CNN.

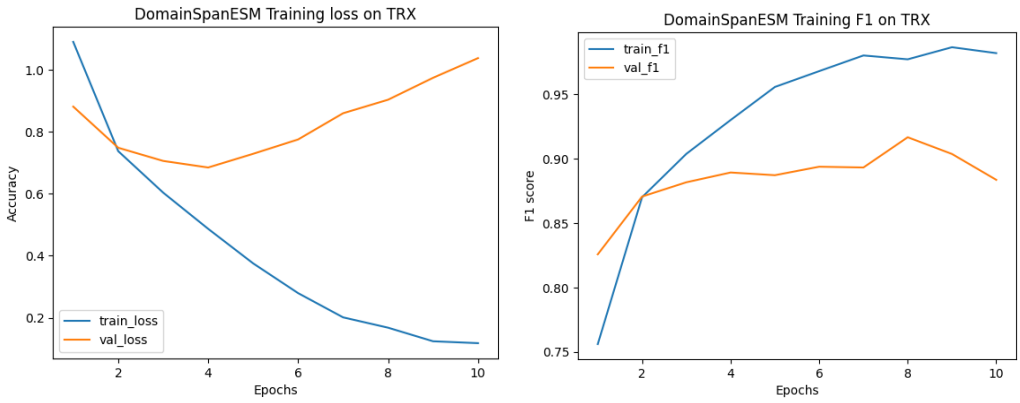

Figura 6.2: Loss (sinistra) e F1 score (destra) durante addestramento DomainSpanESM per il train e validation set TRX, sul seed migliore

Model Classifier	Validation		Test	
	Accuracy	F1	Accuracy	F1
One-hot encoding				
1-D CNN	0.84 ± 0.0	0.84 ± 0.0	0.837 ± 0.1	0.837 ± 0.1
LSTM	0.654 ± 0.27	0.63 ± 0.30	0.639 ± 0.25	0.618 ± 0.28
ESM - Average along hidden dimension				
SVM	0.822	0.821	0.827	0.826
Gradient Boosting	0.704 ± 0.0	0.702 ± 0.0	0.690 ± 0.0	0.688 ± 0.0
Random Forest	0.71 ± 0.0	0.706	0.690 ± 0.0	0.686 ± 0.01
k-NN	0.693	0.685	0.705	0.699
Baseline ESM Freezed	0.717 ± 0.01	0.715 ± 0.01	0.694 ± 0.01	0.691 ± 0.01
Baseline ESM Finetuned	0.893 ± 0.01	0.893 ± 0.01	0.887 ± 0.01	0.887 ± 0.01
ESM Freezed + 1-D CNN	0.878 ± 0.0	0.878 ± 0.0	0.876 ± 0.01	0.875 ± 0.01
ESM Finetuned + 1-D CNN	0.878 ± 0.01	0.879 ± 0.01	0.874 ± 0.02	0.874 ± 0.02
ESM Freezed + LSTM	0.888 ± 0.01	0.888 ± 0.01	0.887 ± 0.01	0.885 ± 0.01
ESM Finetuned + LSTM	0.891 ± 0.0	0.891 ± 0.0	0.887 ± 0.0	0.887 ± 0.0
DomainSpanESM (ESM Finetuned + Domain Span)	**0.906 ± 0.01**	**0.906 ± 0.01**	**0.896 ± 0.01**	**0.895 ± 0.01**

Tabella 6.2: Performance delle Architetture testate aggregate su tre diversi seed

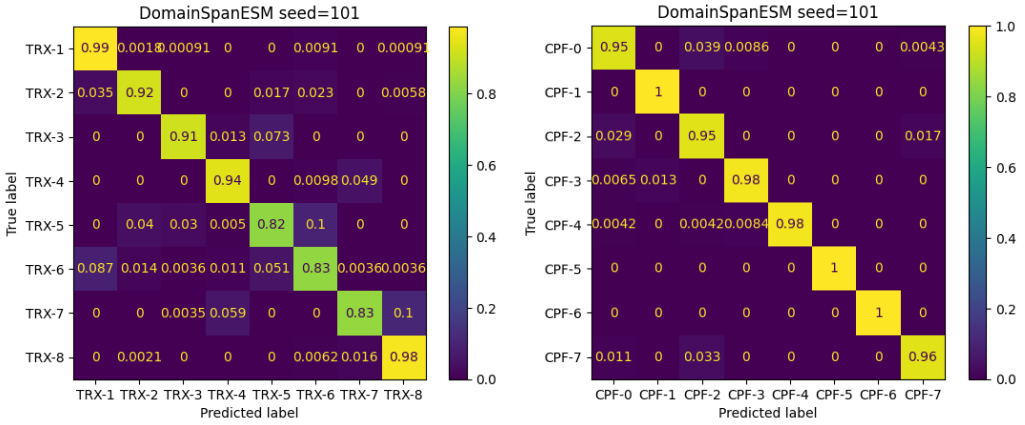

Figura 6.3: Confusion Matrix DomainSpanESM per TRX (sinistra) e CPF (destra)

6.0.1 Analisi sull'importanza delle dimensioni del Train set

La tabella 6.3 dimostra che DomainSpanESM ottiene prestazioni migliori quando viene addestrato su set di dati più ampi. La curva delle prestazioni si stabilizza su un'accuratezza e un F1-score di 0,90 sul validation e 0,89 sul test quando il 60% delle 28.601 sequenze TRX viene utilizzato per l'addestramento, avvicinandosi allo stesso livello di prestazioni raggiunto anche addestrando l'80% delle sequenze, come discusso in precedenza, in figura 6.1.

6.0.2 Similarità sequenze TRX

Data la natura omologa delle sequenze, abbiamo ipotizzato potenziali difficoltà per i modelli nel discriminare le classi su dati di input molto simili. Nella tabella 6.4 abbiamo calcolato un punteggio di similarità per classe, ovvero la media dell'identità di sequenza calcolata a coppie tra due classi. I risultati ottenuti sono in linea con le aspettative, poiché le sequenze erano state precedentemente raggruppate utilizzando mmseqs [39], per ridurre il numero di sequenze tramite clustering, con un'identità di sequenza minima di 0.5 per ridurne il numero per ProfileView.

Training set size	Validation		Testing	
	Accuracy	F1-score	Accuracy	F1-score
286 (1%)	0.6949	0.6827	0.6997	0.6859
2,860 (10%)	0.8508	0.8494	0.8291	0.8275
5,720 (20%)	0.8599	0.8584	0.8624	0.8621
11,440 (40%)	0.8943	0.8937	0.8874	0.8868
17,159 (60%)	0.9074	0.9076	0.8925	0.8918

Tabella 6.3: DomainSpanESM performance sul training al variare delle dimensioni del train set per le TRX. L'incremento delle dimensione tra parentesi rispetto al totale.

	TRX-1	TRX-2	TRX-3	TRX-4	TRX-5	TRX-6	TRX-7	TRX-8
TRX-1	1.0							
TRX-2	0.51 ± 0.10	1.0						
TRX-3	0.50 ± 0.11	0.49 ± 0.11	1.0					
TRX-4	0.56 ± 0.14	0.53 ± 0.13	0.50 ± 0.12	1.0				
TRX-5	0.51 ± 0.09	0.50 ± 0.12	0.49 ± 0.11	0.52 ± 0.13	1.0			
TRX-6	0.50 ± 0.09	0.49 ± 0.10	0.48 ± 0.10	0.51 ± 0.12	0.49 ± 0.12	1.0		
TRX-7	0.59 ± 0.12	0.55 ± 0.12	0.51 ± 0.12	0.49 ± 0.11	0.53 ± 0.12	0.52 ± 0.12	1.0	
TRX-8	0.61 ± 0.13	0.57 ± 0.12	0.53 ± 0.11	0.49 ± 0.10	0.54 ± 0.12	0.54 ± 0.12	0.49 ± 0.10	1.0

Tabella 6.4: Identità delle sequenze con valore medio di match score (= 1.0) per le classi TRX

6.0.3 ProfileView embeddings vs ESM-2 embeddings

Abbiamo confrontato gli embedding di ProfileView delle sequenze TRX di addestramento (figura 5.2 in alto) con i loro corrispondenti embedding di ESM-2 (figura 5.2 in basso), calcolati i vettori con i valori medi lungo la dimensione interna. Gli embedding di ProfileView risiedono in uno spazio multidimensionale di alcune centinaia di dimensioni, specificamente progettato per estrarre informazioni funzionali dalle sequenze, similmente gli embedding di ESM-2 operano in uno spazio ad alta dimensionalità di 480. La rappresentazione di ProfileView risulta leggermente più distinta rispetto a quella di ESM-2, evidenziando le difficoltà nel discriminare efficacemente tra le classi.

6.0.4 Performance temporali

ProfileView partendo dalla creazione della libreria dei modelli, finendo alla costruzione dell'albero può impiegare mediamente dalle 5 alle 8 ore. Alternativamente, se si è già in possesso della libreria la seconda parte impiega tra i 60 e i 90 minuti, con un limite massimo di circa 30 o 40 mila sequenze con 64 GB di RAM. Invece, la complessità di un'architettura Transformer, guidata dal parallelismo del meccanismo di Multi-Head Attention, cresce linearmente con il numero di sequenze e quadraticamente con la lunghezza delle sequenze a causa del calcolo della Self-Attention [**?**]. Per evitare di superare i limiti di memoria della GPU, ESM-2 è stato testato solo con sequenze fino a 1024 token [26]. Durante i test, i tempi di inferenza registrati sono stati di 33 ms per una singola sequenza e 1,07 secondi per un batch di 32 sequenze. L'addestramento è stato condotto su una GPU NVIDIA A4000 con 16 GB di VRAM, con un tempo medio di addestramento di circa 25 minuti per epoca.

6.0.5 Analisi segnali funzionali

Segnali provenienti dalle classi TRX

Un'analisi delle otto classi funzionali TRX ha rivelato profili di conservazione specifici per ciascuna classe, vedi figura 5.3B. L'allineamento del motivo mostra un ruolo funzionale svolto da due cisteine globalmente conservate e una prolina (figura 5.3B) all'interno delle classi TRX-1 e TRX-2. In particolare, queste due classi includono sequenze della microalga *Chlamydomonas reinhardtii* e di altre piante, dove è stato identificato il motivo del legame disolfuro CGPC, noto in questa specie per la sua importanza funzionale. Questo motivo è localizzato nelle posizioni 31-34 nella sequenza TRX-f2 di *C. reinhardtii* e corrisponde ai residui evidenziati in rosso scuro nella struttura PDB associata (vedi figura 5.3EF).

Segnali all'interno delle classi TRX

Per la specifica proteina TRX-f2 di *C. reinhardtii*, che appartiene al sottoalbero TRX-2 composto da 1699 sequenze, studi precedenti [3] hanno riportato un'estensione conservata del motivo del legame disolfuro, WCGPCK. Questo motivo esteso presenta un triptofano in posizione 30 e una lisina in posizione 35, ed è circondato da una "corona" di sei ulteriori lisine (Lys7, Lys43, Lys60, Lys63, Lys72, Lys93; evidenziate in verde oliva nella figura 5.3F) e due asparagine (Asn59, Asn62; mostrate in arancione).

Basandosi sull'analisi fatta sugli allineamenti, abbiamo identificato automaticamente il motivo esteso WCGPCK come altamente significativo (in tonalità di rosso, e Lys35 in verde oliva, figura 5.3F) per il sottoalbero TRX-2. Questo risultato supporta le osservazioni precedenti secondo cui residui aggiuntivi sono coinvolti nella formazione di un motivo tridimensionale esteso del sito attivo [50].

Inoltre, approfondendo ulteriormente l'albero e selezionando le sequenze vicine alla TRX-f2 di *C. reinhardtii* (corrispondente al sottoalbero con la radice arancione nella figura 5.3C), abbiamo costruito un profilo TRX-f2, figura **??**D) che arricchisce la nostra comprensione della proteina TRX-f2 di *C. reinhardtii*.

Innanzitutto, conferma le stesse sei lisine aggiuntive e le due asparagine descritte in [3]. In secondo luogo, incrociando le lisine conservate nel profilo TRX-f2 con quelle presenti nella sequenza di *C. reinhardtii*, siamo stati in grado di identificare una lisina extra con un potenziale ruolo biologico significativo, arricchendo ulteriormente la "corona" di lisine precedentemente osservata.

L'analisi evidenzia il valore dell'esplorazione dello spazio funzionale delle sequenze, poiché può rivelare regioni precedentemente non identificate della proteina che potrebbero avere un'importanza biologica.

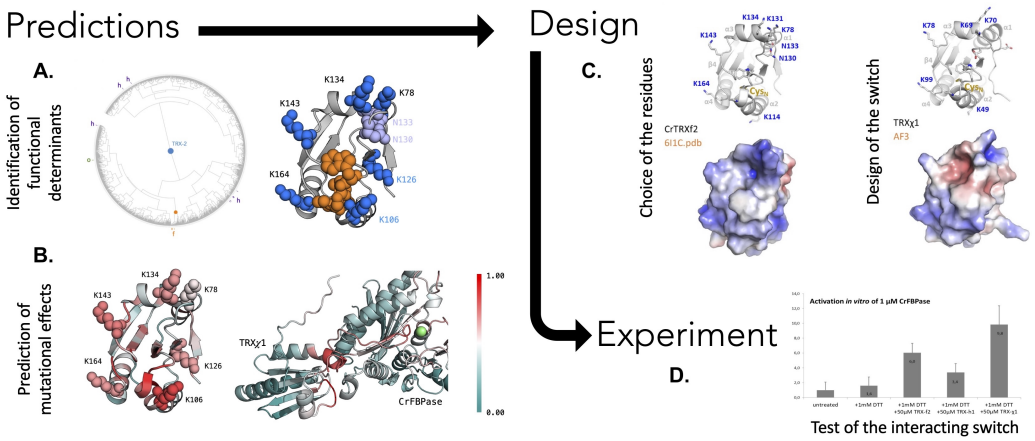

Figura 6.4: Pipeline dello studio condotto in [6] per recodifica di una funzione TRX-h in TRX-f. **A.**: individuazione dei determinanti funzionali con ProfileView. **B.**: Analisi con MuLAN sull'effetto delle mutazioni sui residui individuati. **C.**: Visualizzazione nella struttura tridimensionale (PDB 6I1C) dei residui scelti (sinistra) e della mutazione da apportare per la recodifica (destra). **D.** Esperimento *in vitro* per la verifica della recodifica

Studio sulla recodifica della funzione proteica

Partendo dall'intuizione che esistono dei residui specifici associati alla funzione della proteina (determinanti funzionali), insieme al lavoro di ricercatori del LCQB, è stata esplorata con successo l'idea di modifica dei determinanti funzionali individuati con ProfileView per ricodificare la funzione di una proteina. Nell'articolo [6] sono state prese in considerazione le sequenze della classe TRX-2 rappresentate in figura 6.4A. All'interno della classe sono presenti 301 sequenze già annotate sperimentalmente che si distinguono in 3 tipi funzionali, etichettate come "f", "h" e "o". Nello specifico, le 34 sequenze di tipo f sono raggruppate in maniera compatta, segno della precisione di ProfileView. Tramite allineamento di queste sequenze sono stati estratti i residui più conservati, trovando diverse lisine importanti.

In seguito con l'utilizzo di MuLAN (Mutation-driven Light Attention Networks) [51], un modello di DL per la predizione degli effetti delle mutazioni

dei residui nelle interazione tra proteine e l'identificazione dei siti di interazione, è stato possibile confermare questi determinanti funzionali come critici per l'attività della funzione f e la costruzione dei legami tra proteine nella famiglia TRX (figura 6.4B).

Sei lisine del gruppo individuato precedentemente sono risultate cruciali per la funzione f e altamente sensibili a mutazioni, quindi a determinare la funzione. Dopo un'analisi visualizzando graficamente questi residui nella struttura (figura 6.4C) e da precedenti studi è stato scelto di mutare cinque citosine per una sequenza dell'organismo *C. reinhardtii* di tipo h, caratteristiche invece per la funzione h, in lisine. Solo dopo esperimento *in vitro* in laboratorio, ovvero al difuori dell'organismo vivente opposto di *in vivo*, è stata accertata che la nuova funzione ricodificata simula perfettamente la funzione f (figura 6.4D). Questo studio è un importante passo in avanti per la ingegnerizzazione e progettazione delle proteine, che ci aiuta a comprendere meglio la relazione tra sequenza, struttura e funzione, come in campo medico per nuove cure e farmaci.

Capitolo 7

Conclusioni

Avere un metodo di classificazione robusto come DomainSpanESM è essenziale per migliorare la nostra capacità di rilevare e raggruppare sequenze che condividono la stessa funzione, un compito sempre più importante man mano che affrontiamo la sfida di classificare milioni di proteine. Basandosi su metodologie come ProfileView, che identifica in modo efficiente classi e sottoclassi funzionali da migliaia di sequenze, DomainSpanESM sfrutta queste informazioni per arricchire la rappresentazione di sequenze con ruoli simili anche su set di dati di dimensioni arbitrariamente grandi.

Tuttavia, questo approccio aggira la principale sfida computazionale di gestire direttamente milioni di sequenze in un processo di classificazione dinamico. Attualmente, il numero di sottoclassi è fisso fin dall'inizio, basato sull'analisi di ProfileView di un dataset relativamente piccolo. Idealmente, un sistema di classificazione dovrebbe essere in grado di elaborare milioni di sequenze e determinare autonomamente le sottoclassi sfruttando le sottili differenze nei loro segnali codificati.

L'affidamento su ProfileView è cruciale, poiché fornisce un dataset di addestramento sufficientemente ampio e rappresentativo. Come indicato dalla nostra analisi, i modelli di deep learning tendono ad adattarsi e specializzarsi verso le classi più frequentemente osservate. Garantire che ogni classe sia ben rappresentata durante l'addestramento è quindi fondamentale per ottenere

un'elevata accuratezza di classificazione. Questo lavoro non solo evidenzia il potenziale di DomainSpanESM nel migliorare la classificazione delle funzioni proteiche, ma sottolinea anche la necessità di sviluppare metodi futuri che possano catturare dinamicamente l'intero spettro della diversità funzionale in dataset di grande scala.

7.0.1 Lavori Futuri

ProfileView, oltre a fungere da strumento di classificazione, è un potente tool di analisi delle sequenze grazie all'albero funzionale che costruisce per identificare le classi. Tuttavia, per garantire una buona visualizzazione grafica, il numero di dati in input raramente supera il migliaio. Questo rappresenta un limite rispetto all'obiettivo di addestramento di DomainSpanESM che, come mostrato in tabella 6.3, richiede decine di migliaia di campioni per raggiungere performance ottimali. Una possibile strategia per superare questo ostacolo è l'adozione del training self-supervised tramite Pseudo-Labeling. Questa tecnica prevede l'utilizzo di un piccolo dataset etichettato, nell'ordine delle centinaia di sequenze, insieme a un vasto dataset non etichettato, che può contenere decine o centinaia di migliaia di sequenze. L'addestramento aggiunge così un processo iterativo: il modello viene prima allenato sul dataset etichettato. Successivamente, vengono predette le classi per le sequenze del dataset non etichettato. Solo le sequenze con un punteggio di predizione che supera una soglia di confidenza prestabilita vengono aggiunte al dataset etichettato per l'addestramento all'iterazione successiva, arricchendolo con le pseudo-etichette inferite.

Ulteriori potenziali analisi sono nel suo utilizzo con diverse famiglie proteiche da ProfileView, per dimostrare il suo comportamento al variare dei dati. Sarebbe importante osservare se il fenomeno dell'overfitting è sempre presente, come per la TRX e la CPF. Provare ad utilizzare altre tecniche per mitigarne l'effetto, come label smoothing, bilanciamento del dataset tramite oversampling con dati sintetici, per esempio utilizzando ProGen [25].

Parallelamente sarebbe fondamentale analizzare le matrici di attenzione di DomainSpanESM per constatare la possibilità di individuare i residui determinanti della funzione direttamente dagli score di attenzione. Questo ridurebbe notevolmente le tempistiche rispetto agli allineamenti multipli utilizzati finora. Un primo tentativo è già stato fatto aggregando gli score delle varie head e layer del modello, ma con la necessità di allineare sempre le sequenze per classe per un confronto globale degli score.

Infine, sarebbe interessante confrontare diversi modelli come backbone per DomainSpanESM per confrontare i risultati al variare della conoscenza appresa da questi nella diverse tecniche di pre-training. Per esempio con Ankh e ProtBERT [22, 23].

Bibliografia

[1] R Vicedomini, JP Bouly, E Laine, A Falciatore, and A Carbone. Multiple profile models extract features from protein sequence data and resolve functional diversity of very different protein families. *Molecular biology and evolution*, 39(4):msac070, 2022.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Stéphane D Lemaire, Daniele Tedesco, Pierre Crozet, Laure Michelet, Simona Fermani, Mirko Zaffagnini, and Julien Henri. Crystal structure of chloroplastic thioredoxin f2 from chlamydomonas reinhardtii reveals distinct surface properties. *Antioxidants*, 7(12):171, 2018.

[4] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014.

[5] Isibor Ihianle, Augustine Nwajana, Solomon Ebenuwa, Richard Otuka, Kayode Owa, and Mobolaji Orisatoki. A deep learning approach for human activities recognition from multimodal sensing devices. *IEEE Access*, 8:179028–179038, 10 2020.

[6] Stéphane D. Lemaire, Gianluca Lombardi, Andrea Mancini, Alessandra Carbone, and Julien Henri. Functional recoding of chlamydomonas reinhardtii thioredoxin type-h into photosynthetic type-f by switching selectivity determinants. *Frontiers in Plant Science*, 16, 2025.

[7] FRANCIS CRICK. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[8] Chris Avery, John Patterson, Tyler Grear, Theodore Frater, and Donald J. Jacobs. Protein function analysis through machine learning. *Biomolecules*, 12(9), 2022.

[9] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[10] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[11] Jeffrey A. Ruffolo and Ali Madani. Designing proteins with language models. *Nature Biotechnology*, 42(2):200–202, 2024.

[12] Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, et al. Mgnify: the microbiome analysis resource in 2020. *Nucleic acids research*, 48(D1):D570–D578, 2020.

[13] Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P Waman, Paul Ashford, Harry M Scholes, Camilla SM Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, et al. Cath: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[14] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 538–543, New York, NY, USA, 2002. Association for Computing Machinery.

[15] Jianxin Wang, Min Li, Youping Deng, and Yi Pan. Recent advances in clustering methods for protein interaction networks. *BMC Genomics*, 11(3):S10, 2010.

[16] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 10 2017.

[17] Maxat Kulmanov and Robert Hoehndorf. Deepgoplus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, 07 2019.

[18] Jack Hanson, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. *Bioinformatics*, 33(5):685–692, 12 2016.

[19] Jun Wu, Haipeng Qing, Jian Ouyang, Jiajia Zhou, Zihao Gao, Christopher E Mason, Zhichao Liu, and Tieliu Shi. Hifun: homology independent protein function prediction by a novel protein-language self-attention model. *Briefings in Bioinformatics*, 24(5):bbad311, 08 2023.

[20] Serbulent Unsal, Heval Atas, Muammer Albayrak, Kemal Turhan, Aybar C. Acar, and Tunca Doğan. Learning functional properties of proteins with language models. *Nature Machine Intelligence*, 4(3):227–245, 2022.

[21] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[22] Ahmed Elnaggar, Hazem Essam, Wafaa Salah-Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, and Burkhard Rost. Ankh: Optimized protein language model unlocks general-purpose modelling, 2023.

[23] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, 2022.

[24] Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H. Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 03 2007.

[25] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation, 2020.

[26] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level

protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[27] Zhongwei Wan. Text classification: A perspective of deep learning methods. *arXiv preprint arXiv:2309.13761*, 2023.

[28] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[29] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. *Biorxiv*, pages 2020–12, 2020.

[30] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

[31] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell systems*, 12(6):654–669, 2021.

[32] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] Juliana Bernardes, Gerson Zaverucha, Catherine Vaquero, and Alessandra Carbone. Improvement in protein domain identification is reached

by breaking consensus, with the agreement of many profiles and domain co-occurrence. *PLOS Computational Biology*, 12(7):1–39, 07 2016.

[35] Ari Ugarte, Riccardo Vicedomini, Juliana Bernardes, and Alessandra Carbone. A multi-source domain annotation pipeline for quantitative metagenomic and metatranscriptomic functional profiling. *Microbiome*, 6(1):149, 2018.

[36] S R Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 01 1998.

[37] Robert D et al. Finn. Pfam: the protein families database. *Nucleic acids research*, 42(9), 2014.

[38] David J. Lipman and William R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[39] Maria Hauser, Martin Steinegger, and Johannes Söding. Mmseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330, 01 2016.

[40] Gabriel Cardona, Francesc Rosselló, and Gabriel Valiente. A perl package and an alignment tool for phylogenetic networks. *BMC Bioinformatics*, 9(1):175, 2008.

[41] David A Lee, Robert Rentzsch, and Christine Orengo. Gemma: functional subfamily classification within superfamilies of predicted protein structural domains. *Nucleic acids research*, 38(3):720–737, 2010.

[42] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[43] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[44] Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[45] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[46] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[47] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

[48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[50] Despoina AI Mavridou, Emmanuel Saridakis, Paraskevi Kritsiligkou, Erin C Mozley, Stuart J Ferguson, and Christina Redfield. An extended active-site motif controls the reactivity of the thioredoxin fold. *Journal of Biological Chemistry*, 289(12):8681–8696, 2014.

[51] Gianluca Lombardi and Alessandra Carbone. Mulan: Mutation-driven light attention networks for investigating protein-protein interactions from sequences. *bioRxiv*, 2024.

Ringraziamenti

Ci tengo a ringraziare per la realizzazione di questo elaborato il mio relatore il Prof. Andrea Asperti, per la sua ampia conoscenza nel campo ed avermi seguito con i suoi preziosi consigli nella stesura. La mia correlatrice Prof.ssa Alessadra Carbone per la sua eccezionale competenza e disponibilità, oltre alla grande ospitalità del suo laboratorio a Parigi facendomi sentire subito a mio agio. Voglio ringraziare infatti tutti i dottorandi e i ricercatori del laboratorio che mi hanno supportato per questo lavoro, in particolare Gianluca, Marialuce e Vinh-son, insieme a tutti gli altri che mi hanno accompagnato in questa esperienza unica che ricorderò sempre e con cui ho potuto condividere momenti spensierati in giro per Parigi. Per il supporto morale e finanziario devo ringraziare la mia famiglia che è stata sempre al mio fianco senza farmi mai mancare nulla, come la mia ragazza Julie che mi ha sostenuto in questi anni con il suo amore e tanta pazienza. A Luca e Matteo per aver condiviso nel bene e nel male questo lungo percorso universitario. Ai miei amici di sempre, i Nonmeloposso, con cui condivido tutto dalla nascita e definiscono per me il valore dell'amicizia. Ai diversi coinquilini a Bologna per avermi fatto sentire sempre a casa.