



Università degli studi di Genova

---

---

DIPARTIMENTO DI MATEMATICA

Corso di Laurea Magistrale in Matematica - Curriculum  
Matematica Applicata

Tecniche di Deep Learning per la  
previsione di brillamenti solari

---

Anno Accademico 2023/2024

**Candidato:** Alessia Pellegrini

**Relatore:** Sabrina Guastavino  
**Correlatore:** Anna Maria Massone

---

*A chi ha imparato che il coraggio sta nell'affrontare ciò che non conosce,  
accettando l'incertezza come parte del viaggio.*



# Indice

<b>Introduzione</b>	<b>1</b>
<b>1 Metodi di Deep Learning e il Transformer</b>	<b>5</b>
1.1 Introduzione al Machine Learning . . . . .	5
1.1.1 Deep Learning . . . . .	8
1.2 Il Transformer . . . . .	9
1.2.1 Architettura del modello . . . . .	9
1.2.2 Il meccanismo dell'attenzione . . . . .	12
1.2.3 Residual connections e Layer normalization . . . . .	14
1.2.4 Convolutional layers . . . . .	18
1.2.5 Feed-forward Neural Network . . . . .	19
1.3 Loss functions . . . . .	25
1.3.1 Loss functions per la classificazione binaria . . . . .	26
1.3.2 Loss functions per la classificazione multiclasse . . . . .	28
1.4 Metriche di valutazione . . . . .	29
<b>2 Convergenza del Transformer</b>	<b>35</b>
2.1 Teorema di Convergenza . . . . .	35
2.2 Dimostrazione del Teorema di Convergenza . . . . .	38
2.3 Convergenza del Transformer . . . . .	56
2.4 Bias-Variance Tradeoff e Double-Descent Curve . . . . .	58
<b>3 Applicazione ai brillamenti solari</b>	<b>65</b>
3.1 Introduzione al problema . . . . .	65
3.2 Dataset . . . . .	67
3.2.1 Data Preprocessing e Normalizzazione . . . . .	71
3.3 Metodologia . . . . .	72
3.3.1 Il caso multiclasse . . . . .	72
3.3.2 Il caso binario . . . . .	75

---

3.4	Specifiche (o Condizioni) sul modello . . . . .	76
<b>4</b>	<b>Analisi dei risultati</b>	<b>79</b>
4.1	Il caso multiclasse . . . . .	79
4.1.1	Test 1 . . . . .	80
4.1.2	Test 2 . . . . .	84
4.1.3	Test 3 . . . . .	87
4.1.4	Test 4 . . . . .	89
4.1.5	Analisi dei risultati . . . . .	92
4.2	Il caso binario . . . . .	98
4.2.1	Test 1 . . . . .	99
4.2.2	Test 2 . . . . .	102
4.2.3	Test 3 . . . . .	105
4.2.4	Test 4 . . . . .	107
4.2.5	Analisi dei risultati . . . . .	109
4.3	Conclusioni . . . . .	115
	<b>Bibliografia</b>	<b>119</b>

# Introduzione

In questa trattazione, si intendono studiare i metodi di Machine Learning e Deep Learning applicati ai problemi legati alla meteorologia spaziale, nota come Space Weather. L'obiettivo è risolvere problemi di classificazione utilizzando tecniche avanzate per la previsione dei brillamenti solari.

I brillamenti solari, o *solar flares*, sono potenti esplosioni che avvengono sulla superficie del Sole, caratterizzati dal rilascio improvviso di una grande quantità di energia sotto forma di radiazione elettromagnetica, che include raggi X, raggi gamma, radiazioni ultraviolette e luce visibile. Questi eventi hanno origine a partire dalle regioni attive del Sole, dette *Active Regions* (ARs), anche se non tutte le regioni attive del Sole danno luogo a intensi flare.

La comprensione della probabilità che un flare si origini da una regione attiva è di fondamentale importanza per la nostra società. Questi eventi, infatti, se particolarmente intensi, possono rappresentare rischi significativi per le infrastrutture terrestri e spaziali, con potenziali implicazioni economiche e per la sicurezza. Riuscire a prevedere con precisione l'insorgenza di tali fenomeni potrebbe contribuire a mitigare i danni e a proteggere sistemi critici, come reti elettriche, satelliti di comunicazione e navigazione, e altre infrastrutture tecnologiche. Di conseguenza, al fine di ridurre gli effetti delle eruzioni solari e l'impatto economico che le loro conseguenze possono avere sulla Terra, negli ultimi anni la comunità eliofisica si è concentrata sempre più sulla previsione dei brillamenti solari. Questo è stato possibile attraverso l'analisi dei dati attuali e storici relativi al campo magnetico provenienti dalle regioni attive del Sole.

A seconda dell'intensità dell'evento, i brillamenti solari sono classificati in modo logaritmico in base al loro picco di flusso di raggi X come *A*, *B*, *C*, *M* e *X*. In generale, si è particolarmente interessati a prevedere soprattutto i flare di classe *M* e *X*, in quanto tali brillamenti possono innescare espulsioni di massa coronale (CME), che, se dirette verso la Terra, possono generare tempeste geomagnetiche intense che potrebbero impattare il nostro pianeta. Tuttavia, se

---

da una parte i brillamenti di classe M e X sono i più influenti sul nostro pianeta, allo stesso tempo questi eventi sono abbastanza rari. Poiché rappresentano una percentuale molto bassa rispetto alle altre classi, in questo lavoro di tesi andremo ad affrontare problemi di previsione multi-classe caratterizzati da un forte sbilanciamento tra le classi.

In particolare, in questa trattazione, come caso multiclasse, studieremo l'occorrenza di tali eventi suddivisi in tre classi ( $A/B$ ,  $C$ ,  $M/X$ ), prestando particolare attenzione alle percentuali di ciascuna classe nel dataset. D'altra parte, affronteremo uno studio di classificazione binaria in cui addestreremo il modello a prevedere le occorrenze di flares di classe  $\geq M$ . I dati a nostra disposizione sono forniti dallo strumento Helioseismic and Magnetic Imager (HMI) sul satellite NASA Solar Dynamic Observatory (SDO) e forniscono informazioni sulle caratteristiche, dette *features*, del campo magnetico delle *ARs* entro un intervallo di tempo fisso, da cui potrebbero avere origine i brillamenti solari.

Il Machine Learning e il Deep Learning, rami dell'intelligenza artificiale, sono ampiamente riconosciuti come strumenti fondamentali per la previsione degli eventi solari nell'ambito dello Space Weather. Sfruttando i dati storici, il Machine Learning consente ai computer di assimilare conoscenze e fare previsioni precise riguardo scenari futuri.

Il modello che utilizzeremo per la previsione dei brillamenti è il Transformer, un modello di Deep Learning sviluppato da Vaswani [19] nell'ambito dell'elaborazione del linguaggio naturale, il cui principale obiettivo è permettere ai computer di comprendere, interpretare e generare il linguaggio umano in modo utile e significativo. Nonostante il modello sia nato per gestire sequenze di parole, i suoi meccanismi possono essere generalizzati e sfruttati per la predizione su generiche sequenze di dati. La componente principale del Transformer è il meccanismo di attenzione, una tecnica che consente di identificare le relazioni più rilevanti all'interno delle sequenze di dati.

In questa trattazione studieremo la convergenza del Transformer in un caso particolare e la relazione tra convergenza e numero di parametri del modello. Da qui, ci collegheremo allo studio del bias-variance trade-off e della double descent curve per i modelli di Deep Learning, concetti che indicano il comportamento di un modello a seconda del numero di parametri che lo caratterizzano.

Infine, applicheremo tale tecnica di deep learning per la previsione dei bril-

---

lamenti solari, affrontando il problema sia come una classificazione multi-classe che come una previsione binaria, mirata a stabilire l'occorrenza di flare intensi superiori alla classe M, utilizzando dati di serie temporali relativi alle caratteristiche fisiche delle regioni attive sul Sole. In particolare, in questo lavoro confronteremo diversi esperimenti per analizzare come le performance variano modificando lo sbilanciamento delle classi, pur mantenendo percentuali coerenti con il contesto operativo. Nel caso della previsione binaria, adotteremo inoltre una loss function specifica, progettata per ottimizzare una metrica adeguata alla valutazione di previsioni su classi sbilanciate, migliorando così la capacità del modello di gestire eventi rari.

Il lavoro di Tesi è suddiviso come segue.

- Nel Capitolo 1 si introducono le principali tecniche di Machine Learning e Deep Learning per problemi di classificazione binaria e multiclasse. Dopo aver menzionato i principali modelli per la gestione di sequenze di dati, approfondiremo nello specifico il Transformer, analizzando la sua architettura e la sua struttura principale per l'elaborazione dei dati, il meccanismo dell'attenzione. Infine analizzeremo le funzioni di perdita e le metriche di valutazione per generici modelli, distinguendo tra il caso di classificazione binaria e multiclasse.
- Nel Capitolo 2 ci occupiamo di studiare un risultato sulla convergenza del Transformer; in particolare dimostreremo che, sotto determinate condizioni iniziali, un'overparametrizzazione di tipo cubico è sufficiente per la convergenza del modello. In seguito, passeremo all'analisi del comportamento del bias-variance trade-off per modelli di Machine Learning, per poi menzionare un nuovo comportamento osservato in modelli più recenti di Deep Learning, che è la double descent curve.
- Nel Capitolo 3 si studia il problema di previsione dei brillamenti solari, spiegando questi eventi e perchè è importante prevederli. In seguito, si introducono il dataset di riferimento che verrà adottato per la classificazione di questi eventi, evidenziando il significativo sbilanciamento presente nei dati analizzati. Infine si analizza la metodologia scelta per il caso binario e multiclasse.
- Nel Capitolo 4 si analizzano i risultati ottenuti applicando il modello del Capitolo 3 al problema proposto. Sia nel caso multiclasse che binario, si

---

addestrerà il modello su 4 dataset differenti, cercando di capire come le performance di apprendimento e di generalizzazione del modello cambino modificando il bilanciamento delle classi nei vari set di training.

# Capitolo 1

## Metodi di Deep Learning e il Transformer

In questo capitolo si fornisce, nella prima sezione, una descrizione dei metodi di Machine Learning e Deep Learning per problemi di classificazione binaria e multiclasse. Si passa poi allo studio più specifico di uno dei modelli più recenti e innovativi nell'ambito, il Transformer [19], approfondendone nel dettaglio l'architettura e la struttura principale, il meccanismo dell'attenzione. Infine, nell'ultima sezione, si introducono le principali funzioni di perdita utilizzate nel Deep Learning, distinguendo tra il caso di classificazione binaria e classificazione multiclasse.

### 1.1 Introduzione al Machine Learning

Il Machine Learning è una branca dell'Intelligenza Artificiale che comprende i metodi per permettere ad un sistema di apprendere in modo automatico dai dati, senza la necessità di essere programmata in modo esplicito. L'idea è quella di consentire alla macchina di adattarsi ed evolversi, mentre si accumulano le esperienze, ovvero i dati, in modo da imparare le relazioni dietro di essi. D'altra parte, una caratteristica fondamentale che vogliamo il sistema acquisisca è la capacità di generalizzazione dei processi assimilati su nuovi dati, in modo da fornire risultati anche su esempi che la macchina non ha mai visto. In generale, le tecniche del Machine Learning si suddividono in tre categorie, a seconda dei dati e dell'utilizzo richiesto:

- *Supervised learning* : al modello vengono forniti esempi nella forma di coppie di input e output desiderato e l'obiettivo è quello di imparare la

relazione che li lega;

- *Unsupervised learning* : in questo caso i dati forniti non sono etichettati e il sistema cerca di trovare una struttura particolare o pattern all'interno di essi;
- *Reinforcement learning* : il modello apprende l'obiettivo in un ambiente dinamico tramite ripetute interazioni di tipo "trial-and-error".

In questa trattazione ci occupiamo esclusivamente di supervised learning, dove il tipico setting è il seguente: si considera un *set di n campioni*

$$\{(x_i, y_i)\}_{i=1}^n$$

che supponiamo essere distribuiti in maniera *i.i.d.* (indipendenti e distribuiti in modo identico) da una distribuzione  $\rho$  scomponibile nella forma  $\rho(x, y) = \rho(x|y)\nu(x)$ . Gli input  $\{x_i\} \subset \mathcal{X}$  possono essere

- vettori di *features*  $\mathcal{X} = \mathbb{R}^d$ ,
- *serie temporali di features*  $\mathcal{X} = \mathbb{R}^{d \times T}$ ,
- immagini  $\mathcal{X} = \mathbb{R}^{n \times m}$
- serie temporali di immagini (video)  $\mathcal{X} = \mathbb{R}^{n \times m \times t}$ .

Invece gli output  $\{y_i\} \subset \mathcal{Y}$  hanno forme diverse a seconda del problema considerato; in particolare nel caso di *classificazione binaria* gli output possono assumere forma  $\mathcal{Y} = \{0, 1\}$ , nel caso di *classificazione multiclasse* avremo  $\mathcal{Y} = \{1, \dots, K\}$  dove  $K$  è il numero di classi totali, e infine nel caso di *regressione* lo spazio di output comprende valori continui  $\mathcal{Y} \subseteq \mathbb{R}$ .

Lo scopo del sistema è quello di trovare una funzione  $g : \mathcal{X} \rightarrow \mathcal{Y}$ , detta *stimatore*, che da una parte spieghi la relazione esistente tra gli input e gli output, e dall'altra sia in grado di generalizzare il problema, cioè dato un nuovo input  $x \in \mathcal{X}$ , con  $x \neq x_i$  per  $i = 1 \dots n$ , vorremmo che il valore  $g(x)$  sia una buona stima dell'output. Solitamente si considera una forma canonica per lo stimatore, cioè si prende in considerazione una funzione parametrizzata da dei pesi  $w$ ; in questo contesto, allenare il sistema significa proprio trovare i pesi ottimali  $w$  della funzione  $g$  tramite tre diverse fasi di apprendimento:

- nella *fase di allenamento* si mostrano i campioni al sistema, il quale cerca di imparare i pesi migliori;

- durante una *fase di validazione* monitoriamo il comportamento del modello presentando nuovi esempi, controllando la capacità di generalizzazione;
- finito l'apprendimento, segue la *fase di test* in cui si verifica il comportamento del modello su nuovi campioni.

Analizziamo più nel dettaglio le tre fasi.

**Fase di training.** L'addestramento di un modello nel caso di supervised learning consiste nello scegliere una funzione di perdita, detta *loss function*, che misuri l'errore tra l'output predetto e l'output effettivo, per poi andare a cercare di minimizzare tale errore. Di conseguenza, una loss function è una funzione

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, +\infty)$$

che misura la discrepanza tra il valore di output atteso  $y_i$  e il valore di output predetto  $g(x_i)$  per tutti gli  $n$  campioni. Con queste notazioni, addestrare la rete significa trovare i pesi  $w$  che minimizzano l'errore sul dataset, cioè

$$\min_w \frac{1}{n} \sum_{i=1}^n \ell(y_i, g(w, x_i)) \quad (1.1)$$

Solitamente, all'equazione 1.1 si aggiunge un termine di penalizzazione  $\lambda\phi(w)$  per regolarizzare il problema di minimo ed evitare il fenomeno di overfitting, cioè la possibilità che il modello impari a memoria dai dati, perdendo la capacità di generalizzazione. Il parametro  $\lambda$  viene scelto nell'intervallo  $[0, 1)$  e controlla l'entità della penalizzazione: una scelta più vicina a 1 può ridurre l'overfitting ma anche rendere il modello troppo semplice; una scelta più vicino allo zero può migliorare la capacità del modello di adattarsi ai dati di addestramento. Per risolvere questo problema di minimizzazione, si applicano tecniche di tipo iterativo che, una volta inizializzati i pesi, effettuano iterativamente i seguenti passaggi:

- (i) presentati gli input  $x_i \in \mathcal{X}$ , si calcolano gli output della rete  $\hat{y}_i = g(w, x_i)$ ;
- (ii) si effettua un confronto tra l'output reale e quello predetto tramite la funzione di perdita  $\ell(\hat{y}_i, y_i)$ ;
- (iii) si aggiornano i pesi del modello con le regole caratteristiche della loss function.

**Fase di validazione.** Come già accennato sopra, si vuole evitare l' *overfitting*, cioè il fenomeno per cui il modello si specializza sui campioni forniti e perde la capacità di generalizzare la predizione su dati mai visti. A tale scopo si suddividono i campioni etichettati in un training set e un validation set: durante la fase di addestramento si allena il modello sui dati forniti, mentre nella fase di validazione si controlla che effettivamente la rete stia imparando a generalizzare, studiando l'andamento della loss sul corrispondente set. Una tecnica utilizzata in questa fase è quella dell' *early stopping*, in cui si sceglie un numero di epoche  $N_p$ , detto *patience*, e si arresta il modello quando l'errore calcolato tramite la loss sul validation set non migliora dopo le  $N_p$  epoche scelte. I pesi ottimali scelti per la rete sono quelli corrispondenti all'epoca ottimale, in cui l'errore è il più basso sul validation set.

**Fase di test.** Infine, durante la fase di test, si sperimentano i pesi ottenuti su nuovi dati e si valuta la bontà della soluzione tramite una serie di metriche che dipendono dal problema affrontato e che introdurremo più avanti nella sezione 4 del seguente capitolo.

### 1.1.1 Deep Learning

Il Deep Learning è un ramo del Machine Learning che coinvolge tecniche avanzate fondate sulle *reti neurali artificiali*, basate sull'emulazione del cervello umano e composte da molti *layers* nascosti (da cui il termine 'deep', ovvero profondo) che comprendono un numero arbitrario di *neuroni*. Tra questi modelli, menzioniamo i principali:

- le **Feed-forward Neural Networks (FNNs)** rappresentano il modello classico di rete neurale artificiale, in cui i dati si muovono in una sola direzione, dall'input all'output attraverso i vari layers;
- le **Convolutional Neural Networks (CNNs)** presentano, rispetto alla struttura precedente, layers convoluzionali, i quali permettono di identificare i tratti più rilevanti dei pattern nei dati;
- le **Recurrent Neural Networks (RNNs)** dispongono di una componente di memoria a breve termine grazie alle *residual connections*, le quali permettono di collegare tra loro diverse informazioni presenti nei dati nella gestione di input sequenziali;

- gli **LSTM (long-short term memory)** rappresentano un'evoluzione rispetto alle RNNs, disponendo di una memoria a breve termine e a lungo termine, le quali vengono portate avanti parallelamente durante l'addestramento della rete.

In termini di problemi di classificazione o regressione sui dati sequenziali, possiamo pensare alle quattro strutture precedenti come una il miglioramento dell'altra, poichè introducono progressivamente nuove tecniche per l'apprendimento dai dati. Tuttavia, anche gli LSTM presentano due problemi fondamentali: da una parte non si riescono a memorizzare le informazioni per periodi di tempo troppo lunghi; dall'altra sono modelli molto lenti da addestrare dal momento che non sono parallelizzabili.

In questa trattazione, andremo a studiare un modello che rappresenta un superamento di queste due problematiche, ovvero il **transformer**, architettura sviluppata da Vaswani nel 2017 che si basa sul *meccanismo dell'attenzione* [19]. Vedremo la struttura dettagliata del modello nella prossima sezione, mentre per ora ci accontentiamo di accennare al fatto che, nella sua struttura, esso comprende le quattro architetture sopra definite, in aggiunta al meccanismo dell'attenzione.

## 1.2 Il Transformer

Il **transformer** è un modello di Deep Learning sviluppato da Vaswani [19] per affrontare problemi di natura NLP (Natural Language Processing), ovvero per l'elaborazione del linguaggio naturale, il cui principale obiettivo è permettere ai computer di comprendere, interpretare e generare il linguaggio umano in modo utile e significativo. Il modello è nato nell'ambito delle traduzioni linguistiche, in particolare dall'inglese al tedesco, ed in seguito è stato utilizzato come modello generativo per la produzione di testi. In questo senso, lo scopo del modello è, dato un set di parole, prevedere la parola successiva nel testo. Tuttavia, anche se il modello è nato per gestire sequenze di parole, i suoi meccanismi possono essere generalizzati e sfruttati per la predizione su generiche sequenze di dati.

### 1.2.1 Architettura del modello

In questa sezione vediamo nel dettaglio l'architettura del transformer, cercando di spiegare le funzionalità di ciascuna componente e le formule matematiche

alla base di esse. In primo luogo vediamo la struttura con cui è nato il modello, per poi andare a considerare solo i meccanismi che abbiamo utilizzato nello sviluppo dei codici.

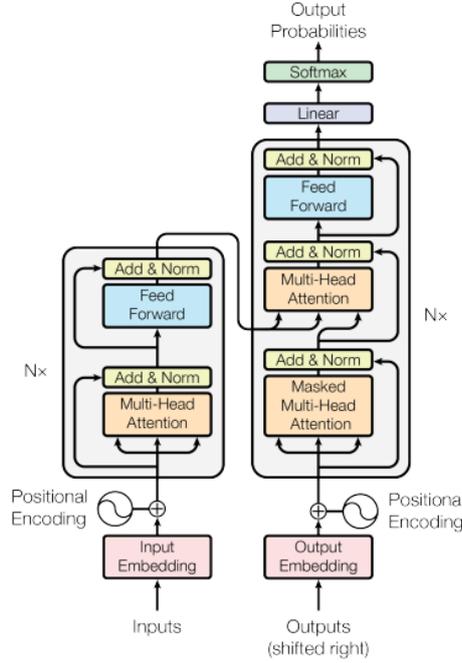


Figura 1.1: Architettura del Transformer [19]

Come la maggior parte dei modelli di trasduzione di sequenze, anche il transformer è composto dalle strutture principali di **encoder** e **decoder** (figura 1.1), entrambi composti da diversi layers che ricevono i dati attraverso un canale di embedding e positional encoding, che hanno il compito di tradurre le sequenze di parole in vettori matematici. Da una parte, l'encoder mappa la sequenza di input  $(x_1, \dots, x_n)$  in una rappresentazione continua del dato  $\mathbf{z} = (z_1, \dots, z_n)$  [19] attraverso uno strato di *multi-head attention*, uno strato di normalizzazione e connessione residua e infine una rete neurale *feed-forward*. D'altra parte il decoder, preso in input  $\mathbf{z}$  e l'output predetto per la sequenza precedente, genera una nuova sequenza di output  $(y_1, \dots, y_m)$  attraverso gli stessi meccanismi presenti nell'encoder. Di conseguenza, ad ogni passo, il modello è *auto-regressivo*, cioè la sequenza di output generata diventa un input aggiuntivo per la generazione dell'output successivo.

Come già accennato, in questa trattazione non utilizziamo la struttura con cui è nato il transformer, ma costruiamo un modello che utilizza solo la parte di encoder e sfrutta il meccanismo dell'attenzione per catturare eventuali dipendenze tra i dati e affrontare il problema di classificazione su un dataset

di brillanti solari. L'architettura del modello usato è presentata in figura 1.2

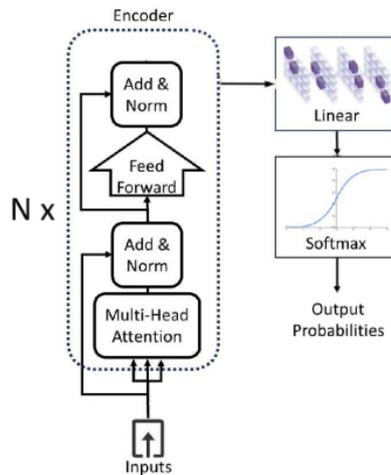


Figura 1.2: Architettura del modello basato sul Transformer [2]

Introduciamo brevemente le componenti principali che formano il modello e ne approfondiamo lo studio nelle prossime pagine. L'encoder è una pila di  $N$  strati identici e ciascun blocco è composto da:

- Il **multi-head attention layer**, composto da multipli blocchi che implementano il meccanismo dell'attenzione, il quale crea i cosiddetti *vettori dell'attenzione* che andranno a rappresentare le relazioni principali tra i dati, indicandone l'importanza a seconda del contesto;
- I layers **add & norm**, i quali servono a implementare la normalizzazione della sequenza e le *residual connections*, che permettono di propagare meglio l'informazione;
- Una **rete feed-forward** che introduce delle ulteriori non-linearità all'interno del modello. In particolare, il dato proveniente dai precedenti layers viene processato con dei *convolutional layers*, fondamentali per l'estrazione delle features nelle serie temporali.

Dopo l'encoder, si procede con l'applicazione di una **Feed-forward Neural Network** (FNN) finale, che rappresenta la parte del modello che si occupa di generare la previsione. Si implementa un tipo particolare di FNN, il Perceptrone Multi-Strato (MLP). L'ultimo strato applica una funzione di attivazione per trasformare gli output in previsioni per il modello.

## 1.2.2 Il meccanismo dell'attenzione

La prima struttura che incontriamo nel modello è il multi-head attention, ovvero il meccanismo di applicazione di diversi livelli di attenzione che corrono in parallelo. Ciascun livello, cioè ciascun *head-attention*, calcola una rappresentazione di attenzione degli input, per poi concatenare le rappresentazioni ottenute.

Il **meccanismo dell'attenzione** [19] è una tecnica che permette di trovare le relazioni tra i dati all'interno di una sequenza: data una sequenza di dati  $x = [x_1, \dots, x_T]$ , si vogliono trovare le possibili connessioni di ciascun elemento  $x_i$  con tutti gli altri elementi della sequenza. In questo contesto, si introducono i concetti di query, key e value.

- Le **queries** sono una rappresentazione degli input che il modello sfrutta per cercare le informazioni più rilevanti dentro le keys. In pratica ogni elemento della sequenza  $x$  genera una query, il cui scopo è trovare quali sono gli altri elementi rilevanti per essa, cioè le connessioni.
- Le **keys** sono una rappresentazione degli input che il modello confronta con le queries per determinare la sua rilevanza rispetto alla domanda proposta. Ogni elemento della sequenza genera una chiave, che rappresenta il suo riferimento ed è usata per valutare quanto l'elemento considerato è utile alla query.
- Infine i **values** rappresentano i valori corrispondenti alle keys che una query ha trovato e corrispondono alle informazioni effettive. Il modello, dopo aver determinato le keys rilevanti rispetto a una query, utilizza i values corrispondenti per costruire la rappresentazione finale.

Una funzione di attenzione può essere descritta come una mappa di una query e di un insieme di coppie chiave-valore su un output; nel caso del transformer, gli autori hanno chiamato il meccanismo utilizzato "Scaled Dot-Product Attention" [19].

Ciascun *head-attention*  $h$  associa ad ogni elemento  $x_i$  della sequenza una terna di vettori

$$(Q^h(x_i), K^h(x_i), V^h(x_i))$$

che rappresentano rispettivamente le queries, le keys e i values dell'elemento  $i$ . Denotiamo con  $d$  la dimensione di ciascun elemento della sequenza, cioè

$x \in \mathbb{R}^{d \times T}$ , e con  $k$  la dimensione dei vettori della terna. Ciascuno dei tre vettori si ottiene tramite una moltiplicazione matriciale dell'input  $x_i$  con una diversa matrice di pesi apprendibili, siano esse

$$W_q^h, W_k^h, W_v^h \in \mathbb{R}^{d \times k}$$

Dunque si ottengono, rispettivamente, i vettori

$$Q^h(x_i) = (W_q^h)^T x_i \in \mathbb{R}^k \quad (1.2)$$

$$K^h(x_i) = (W_k^h)^T x_i \in \mathbb{R}^k \quad (1.3)$$

$$V^h(x_i) = (W_v^h)^T x_i \in \mathbb{R}^k \quad (1.4)$$

Per ciascun elemento  $x_i$  si calcolano i pesi dell'attenzione tramite un prodotto vettore-matrice, una riscalatura e l'applicazione della funzione softmax che converte i valori scalati in probabilità, in modo che la somma dei pesi di attenzione per ogni elemento sia 1. Definiamo la matrice

$$K^h = [K^h(x_1), \dots, K^h(x_T)] \in \mathbb{R}^{k \times T}$$

da cui i pesi dell'attenzione  $A_i \in \mathbb{R}^{1 \times T}$  relativi all'elemento  $i$  si ottengono come

$$A_i = \text{softmax}\left(\frac{(Q^h(x_i))^T K^h}{\sqrt{k}}\right). \quad (1.5)$$

Detta  $V^h$  la matrice

$$V^h = [V^h(x_1), \dots, V^h(x_T)]^T \in \mathbb{R}^{T \times k}$$

l'attenzione relativa a  $x_i$  si calcola applicando i pesi dell'attenzione ai values

$$\text{Attention}(Q^h(x_i), K^h, V^h) = A_i V^h \in \mathbb{R}^{1 \times k} \quad (1.6)$$

Più in generale, detta

$$Q^h = [Q^h(x_1), \dots, Q^h(x_T)] \in \mathbb{R}^{k \times T}$$

la matrice delle queries, l'attenzione per l'  $h$ -esimo head attention è data dalla mappa

$$\text{head}_h = \text{Attention}(Q^h, K^h, V^h) \quad (1.7)$$

$$= \text{softmax}\left(\frac{(Q^h)^T K^h}{\sqrt{k}}\right) V^h \in \mathbb{R}^{T \times k}. \quad (1.8)$$

Nel caso del Multi-Head Attention, le operazioni sopra descritte vengono calcolate dal modello in parallelo per ogni testa di attenzione  $h = 1, \dots, H$ , generando una matrice di dimensione  $T \times k$ . Successivamente le  $H$  matrici vengono concatenate e moltiplicate con una matrice di pesi per formare l'output del Multi-Head Attention

$$MultiHead(Q, K, V) = W_o \text{concat}(head_1, \dots, head_H)^T \in \mathbb{R}^{d \times T} \quad (1.9)$$

dove  $W_o = [W_o^1, \dots, W_o^H] \in \mathbb{R}^{d \times kH}$  è una matrice di pesi apprendibili e ciascuna  $W_o^h \in \mathbb{R}^{d \times k}$ .

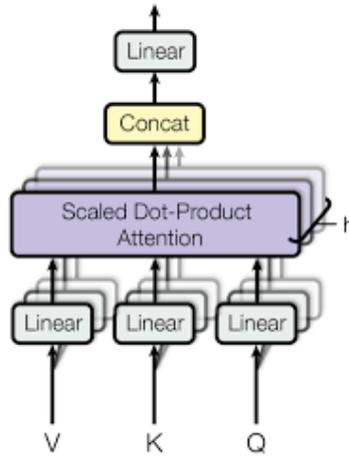


Figura 1.3: Meccanismo del multi-head attention [19]

Di conseguenza, guardando al risultato in componenti [18], questo meccanismo fornisce come output una matrice  $z = MultiHead(Q, K, V) \in \mathbb{R}^{d \times T}$  le cui  $T$  colonne sono i vettori  $d$ -dimensionali ottenuti come

$$\hat{y}_i = \sum_{h=1}^H (W_o^h) \left( \sum_{j=1}^T (A_i)_j V^h(x_j) \right)^T \in \mathbb{R}^d \quad (1.10)$$

per  $i = 1, \dots, T$ . Si noti che l'output generato dal blocco ha la stessa dimensione dell'input fornito al blocco stesso: questa osservazione è fondamentale per l'applicazione dei passaggi che seguiranno.

### 1.2.3 Residual connections e Layer normalization

Durante l'addestramento del modello, può verificarsi il fenomeno del **vani-  
shing/exploding gradient**, un problema per cui man a mano che si entra più in profondità nella rete, i gradienti calcolati per l'aggiornamento dei pesi

risultano essere troppo piccoli (o grandi, rispettivamente) per un’effettivo addestramento della rete. Questo comportamento implica, nel primo caso, una lenta convergenza o il mancato addestramento del modello, mentre, nel secondo caso, gli aggiornamenti possono diventare così grandi da far oscillare i pesi in maniera eccessiva, impedendo al modello di convergere verso una soluzione ottimale o portando alla divergenza della funzione di perdita.

Nell’architettura del Transformer sono stati introdotti due oggetti per la risoluzione di questo potenziale problema, ovvero le residual connections e la Layer Normalization [19].

Le **residual connection** [11] sono delle connessioni tra diversi strati, che collegano l’input di un blocco di strati all’output del blocco stesso. Se  $\mathcal{F}$  è una funzione che rappresenta un blocco di strati (come ad esempio il Multi-head attention), una residual connection consiste nella somma dell’input del blocco  $x$  con l’output generato, cioè

$$y = \mathcal{F}(x) + x.$$

Si noti che la dimensione dell’output del blocco  $\mathcal{F}$  e dell’input fornito devono essere compatibili. Queste “scorciatoie”, anche dette *skip connections*, implementano semplicemente un’identità, ma hanno diversi vantaggi nell’addestramento del modello [16]:

- (i) le skip connections consentono un flusso del gradiente senza ostacoli e questo ha importanti conseguenze sulla back-propagation, cioè l’algoritmo per l’aggiornamento dei pesi. Durante l’addestramento, infatti, tali aggiornamenti si trasmettono in modo più efficace, evitando il problema del vanishing/exploding gradient;
- (ii) queste connessioni creano percorsi di lunghezza variabile tra l’input e l’output. In questo contesto, i percorsi più brevi permettono un apprendimento più semplice e rapido, mentre quelli più lunghi forniscono dettagli più fini all’addestramento. Questo rende l’approccio più flessibile e il modello in grado di adattarsi nella gestione di input con diverse complessità, per cui gli input con una maggiore complessità potrebbero attraversare un numero maggiore di connessioni per estrarre le caratteristiche rilevanti. Tale approccio è anche detto *apprendimento residuo*, in cui l’apprendimento lungo i percorsi più lunghi è un affinamento dell’apprendimento lungo quelli più brevi.

Di conseguenza, le connessioni residuali migliorano l'efficienza del modello, permettono un apprendimento più flessibile e robusto e conferiscono alla rete una struttura simile a un insieme di reti più semplici.

L'utilizzo di **layers di normalizzazione** intermedi durante l'addestramento è progettato per stabilizzare e accelerare il training delle reti neurali [4]. Più un modello diventa profondo, più il suo allenamento risulta complicato: durante l'addestramento, i parametri (o pesi) dei layer vengono aggiornati continuamente attraverso il processo di back-propagation; ciò causa una variazione costante nella distribuzione degli input di ciascun layer. Questo fenomeno, noto come *internal covariate shift* [12], comporta un rallentamento nell'addestramento del modello, dal momento che i gradienti calcolati per l'aggiornamento dei pesi diventano meno affidabili, da cui si rende necessario l'utilizzo di valori più bassi per i *learning rates*, cioè i parametri di velocità per l'aggiornamento. Nel contesto del Transformer, si implementa la *Layer Normalization (LN)* [23], utilizzata per normalizzare gli output di ciascun sotto-strato, cioè dopo ciascuna multi-head attention e feedforward network, e che presenta diversi vantaggi [21, 23]:

- (i) Aiuta a controllare il problema del vanishing gradient, mantenendo gli output di ciascun sotto-strato in un range stabile, contribuendo a una back-propagation più stabile nel tempo.
- (ii) Agisce anche come meccanismo di regolarizzazione, aiutando a controllare la grandezza degli output degli strati. Questo può contribuire a ridurre l'overfitting durante l'addestramento, migliorando la generalizzazione del modello.
- (iii) Essendo applicata allo stesso modo sia durante l'addestramento che durante la fase di test, i parametri di normalizzazione (media e varianza) sono calcolati e fissati durante l'addestramento e utilizzati senza modifiche durante l'uso del modello.
- (iv) Data la natura del Transformer, che eccelle nel trattare sequenze lunghe grazie al meccanismo del multi-head attention, la layer normalization aiuta a mantenere la stabilità delle rappresentazioni lungo le diverse posizioni della sequenza.

Nel caso di serie temporali di features, la *LN* normalizza ciascun istante temporale separatamente rispetto alle sue features. Vengono calcolate:

- la media per ciascuna features  $i = 1, \dots, d$  su tutti i  $T$  tempi:

$$\mu_i = \frac{1}{T} \sum_{t=1}^T x_{i,t}; \quad (1.11)$$

- la varianza per ciascuna features  $i = 1, \dots, d$  su tutti i  $T$  tempi:

$$\sigma_i^2 = \frac{1}{T} \sum_{t=1}^T (x_{i,t} - \mu_i)^2; \quad (1.12)$$

- la normalizzazione per ciascuna features  $i = 1, \dots, d$  e ciascun tempo  $t = 1, \dots, T$

$$\hat{x}_{i,t} = \frac{x_{i,t} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}, \quad (1.13)$$

dove  $\varepsilon$  è una piccola costante aggiuntiva per evitare le divisioni per zero;

- infine si ha l'applicazione di una trasformazione lineare i cui parametri  $\beta, \gamma \in \mathbb{R}^d$  devono essere appresi durante l'addestramento del modello; di conseguenza l'output della  $LN$  è dato da

$$y_{i,t} = \gamma_i \hat{x}_{i,t} + \beta_i. \quad (1.14)$$

L'applicazione di questo metodo nell'architettura del Transformer si può dividere in due categorie a seconda della posizione in cui viene utilizzato nel modello [17, 21, 23]:

- la Pre Layer Normalization (Pre-LN, figura 1.4(b)) applica la normalizzazione direttamente all'input per ogni sub-layer  $\mathcal{F}$ , cioè

$$PreLN(x) = x + \mathcal{F}(LN(x)) \quad (1.15)$$

- la Post Layer Normalization (Post-LN, , figura 1.4(a)) applica la normalizzazione dopo ogni residual connection

$$PostLN(x) = LN(x + \mathcal{F}(x)); \quad (1.16)$$

Entrambe le posizione possiedono dei vantaggi per il modello, infatti la prima aiuta a prevenire il fenomeno del vanishing gradient, portando ad un addestramento più stabile, mentre la seconda, nonostante tenda a preservare valori più alti per la norma del gradiente, ha ottenuto prestazioni migliori rispetto alla Pre-LN. Il modello originale proposto da Vaswani [19] implementa la Post-LN, che tuttavia può portare a risultati più instabili [17], dunque nel nostro caso viene utilizzata la Pre-LN.

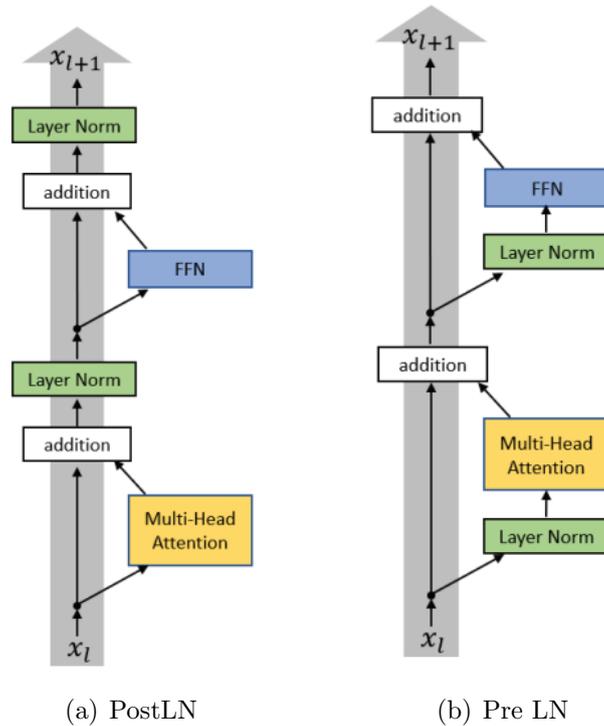


Figura 1.4: Posizioni della Layer Normalization nel Transformer;  $l$  rappresenta il layer del modello [23]

## 1.2.4 Convolutional layers

I convolutional layers sono particolari tipi di connessioni usate principalmente nel campo della computer vision, dal momento che permettono di gestire lunghe sequenze di input mantenendo un contenuto numero di parametri apprendibili, attraverso *interazioni sparse* [1]. Lo scopo di queste connessioni è cercare all'interno dei dati dei pattern, cioè caratteristiche, che si ripetono nell'immagine o nella sequenza. Queste features vengono estratte tramite l'applicazione di filtri convoluzionali, che scansionano il dato alla ricerca di specifiche proprietà. Questa tecnica prende ispirazione dalla biologia, in particolare dal modo in cui agiscono le reti neurali nella corteccia visiva.

Un filtro, o *kernel*, è una matrice di pesi apprendibile che scorre lungo il dato come mostrato in figura 1.5

Nella nostra trattazione utilizziamo la convoluzione 1-dimensionale, cioè la dimensione di profondità del filtro è unitaria. Nelle solite notazione, indichiamo con  $x_i \in \mathbb{R}^{d \times T}$  l' $i$ -esimo campione per  $i = 1, \dots, N$ , con  $d$  numero delle features e  $T$  lunghezza della serie temporale. Un filtro 1-dimensionale si rappresenta come una matrice  $K \in \mathbb{R}^{f \times f}$ , dove  $f$  è la dimensione del filtro. La *convoluzione* tra il campione  $x_i$  e il kernel  $K$  è una matrice  $s$ , spesso detta

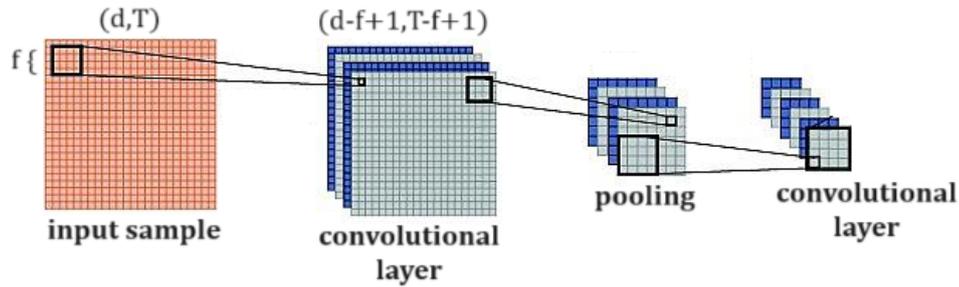


Figura 1.5: Visualizzazione dell'operatore di convoluzione

*feature map*, di dimensione  $(d - f + 1) \times (T - f + 1)$  di componenti

$$s_{n,m} = (x_i * K)(n, m) = \sum_{j=1}^f \sum_{k=1}^f x_i(n + j, m + k) K(j, k) \quad (1.17)$$

con  $n = 1, \dots, d - f + 1$  e  $m = 1, \dots, T - f + 1$ .

Spesso la convoluzione viene seguita da un operatore di *pooling*, che aiuta il modello nella gestione della dimensione del dato. Il pooling, infatti, riduce la dimensione delle feature map, diminuendo così il numero di parametri e, di conseguenza, il carico computazionale. L'idea è quella di definire una finestra, cioè una matrice, e di farla scorrere sulla feature map con un determinato passo, detto *stride*, ed eseguire un'operazione tra gli elementi selezionati dalla finestra. Tra le tecniche più utilizzate vi sono

- *average pooling*: scelta una finestra, si calcola la media dei valori all'interno di ogni matrice ottenuta scorrendo la finestra sulla feature map;
- *max pooling*: si scorre una finestra come nel caso precedente e si seleziona il massimo.

### 1.2.5 Feed-forward Neural Network

Le *feed-forward neural networks (FNNs)* rappresentano il modello più semplice nel campo del Machine Learning [6]. Queste reti si basano sull'emulazione del cervello umano e sono organizzate in strati, anche detti *layers*, composti da vari *neuroni* (o *nodii*) come mostrato in figura 2.3:

- l'**input layer** è il primo strato che riceve i dati di input ed è composto da tanti neuroni quante sono le features del dataset;

- gli **hidden layers** sono possibili strati nascosti compresi tra quello di input e quello di output. In una FNN possono esserci uno o più strati nascosti, che ne determinano la profondità. Ciascun nodo di un hidden layer riceve un input per ognuno dei neuroni del layer precedente e produce un output che viene passato al layer successivo;
- l'ultimo layer, anche detto **output layer**, produce l'output finale della rete. A seconda del problema considerato e della *activation function* utilizzata, l'ultimo strato può contenere uno o più nodi, e quindi generare uno o più output.

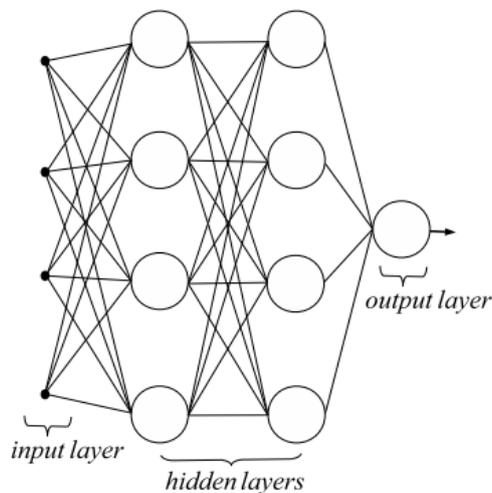


Figura 1.6: Feed-forward neural network con 2 hidden layers

Il termine “feed-forward” indica il fatto che i dati nella rete si muovono in un’unica direzione, dai nodi dello strato di input a quelli di output passando per gli hidden layers. In queste reti non sono presenti cicli o loop, da cui il dato non può muoversi all’indietro.

In particolare, nel nostro modello andremo a considerare un caso specifico di FNN, quello del *multi-layer perceptron (MLP)*, costituito da almeno tre strati di neuroni: uno di input, uno nascosto e uno di output. In questa struttura, tutti i neuroni di ciascun strato sono collegati a ogni neurone dello strato successivo, cioè l’architettura è detta *fully-connected*. L’addestramento di un MLP si divide in due fasi.

- **Forward Propagation:** il processo di training della rete inizia con la propagazione in avanti dei dati di input. Ogni nodo di ciascun layer calcola un output in base agli input forniti dai nodi dei layer precedenti,

applicando a esso una funzione di attivazione. Il procedimento continua fino al raggiungimento dell'output layer, che produce il risultato finale della rete.

- **Back-propagation:** La back-propagation è un algoritmo che addestra una rete, calcolando l'errore commesso sull'output del modello e propagando l'informazione all'indietro. L'informazione viene utilizzata aggiornando i pesi della reti in modo da ridurre l'errore sull'output attraverso metodi di ottimizzazione.

**Forward Propagation.** Vediamo nel dettaglio la formalizzazione matematica [5] del caso più semplice di multi-layer perceptron, ovvero quello in cui è presente un solo strato nascosto, che è il modello che effettivamente implementeremo nel codice. Supponiamo che la rete sia composta da  $d$  neuroni per lo strato di input (cioè gli input sono composti da  $d$  features),  $J$  neuroni per lo strato nascosto e  $M$  neuroni per lo strato di output. Indichiamo con

$$W_{j,i} : \begin{cases} i = 1, \dots, d & \text{indice di provenienza} \\ j = 1, \dots, J & \text{indice di arrivo} \end{cases} \quad (1.18)$$

il peso che collega l' $i$ -esimo neurone dello strato di input al  $j$ -esimo neurone dello strato nascosto, mentre

$$\omega_{k,j} : \begin{cases} j = 1, \dots, J & \text{indice di provenienza} \\ k = 1, \dots, M & \text{indice di arrivo} \end{cases} \quad (1.19)$$

indica il peso che collega il  $j$ -esimo neurone dello strato nascosto al  $k$ -esimo di quello di output (figura 1.7).

Dato un generico input  $x \in \mathbb{R}^d$ , la fase di Forward Propagation consiste nel portare avanti l'informazione del dato attraverso i pesi. Si calcolano i *potenziali di attivazione* del  $j$ -esimo neurone

$$h_j = \sum_{i=1}^d W_{j,i} x_i - W_{j,0} \quad (1.20)$$

dove  $W_{j,0}$  è un parametro apprendibile detto *bias*. Tale parametro può essere incluso nella rete considerando un neurone aggiuntivo per lo strato di input con valore fissato a  $x_0 = -1$ , da cui

$$h_j = \sum_{i=0}^d W_{j,i} x_i. \quad (1.21)$$

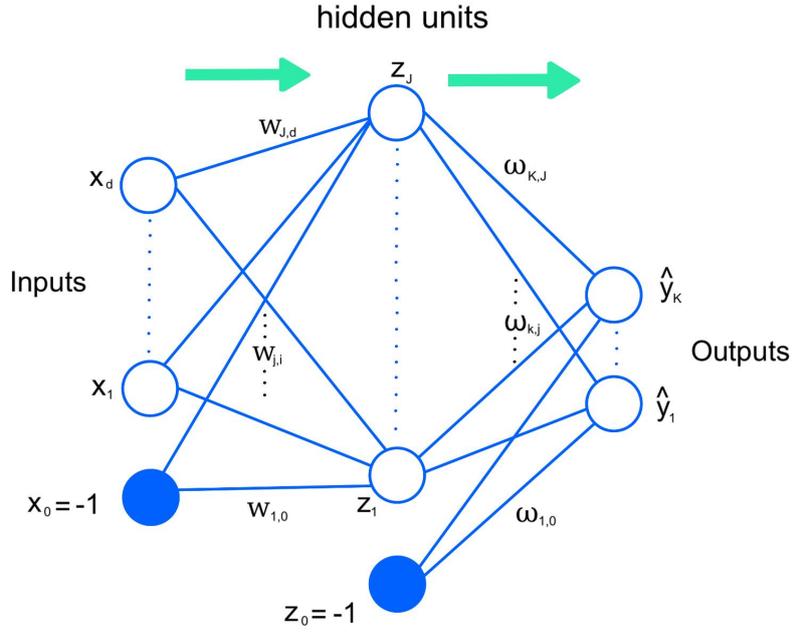


Figura 1.7: Multi-layer perceptron con un uno strato nascosto

Per imitare il comportamento delle soglie di attivazione del cervello umano, a ciascun nodo si associa una funzione di attivazione, che serve a propagare l'informazione quando quest'ultima supera una certa soglia. L'output del  $j$ -esimo neurone è dato dall'applicazione di questa funzione di attivazione  $\sigma$  al potenziale di attivazione, cioè

$$z_j = \sigma(h_j). \quad (1.22)$$

Lo stesso ragionamento si applica per propagare l'informazione dall'hidden layer a quello di output, in cui i nuovi input sono gli output  $z_j$  dello strato precedente. Il potenziale di attivazione del  $k$ -esimo neurone dell'ultimo layer è

$$a_k = \sum_{j=0}^J \omega_{k,j} z_j \quad (1.23)$$

dove, come prima, indichiamo con  $z_0 = -1$  il neurone aggiuntivo per il bias. Presa  $\bar{\sigma}$  funzione di attivazione per l'ultimo strato, l'output finale del  $k$ -esimo neurone è dato da

$$\hat{y}_k = \bar{\sigma}(a_k). \quad (1.24)$$

Sviluppando le sommatorie 1.21 e 1.23, otteniamo l'output  $k$ -esimo della rete come

$$\hat{y}_k = \bar{\sigma} \left( \sum_{j=1}^J \omega_{k,j} \sigma \left( \sum_{i=0}^d W_{j,i} x_i \right) - \omega_{k,0} \right). \quad (1.25)$$

**Backward Propagation.** Come già accennato nella sezione precedente, il passo successivo prevede la scelta di una funzione di perdita che rappresenti l'errore tra l'output calcolato dalla rete  $\hat{y}_k$  e il vero output  $y_k$ . L'algoritmo per l'aggiornamento dei pesi è detto back-propagation, in cui i pesi vengono aggiornati dallo strato finale, passando per gli hidden layers all'indietro per poi aggiornare i pesi del primo strato. Detto  $\mathcal{T} = \{(x_t, y_t) : t = 1, \dots, n\}$  l'insieme del training set, si definisce la funzione di errore relativa ai pesi  $\mathbf{w}, \boldsymbol{\omega}$  la seguente

$$E(\mathbf{w}, \boldsymbol{\omega}) = \sum_{t=1}^n l(y_k, \hat{y}_k) \quad (1.26)$$

e lo scopo è quello di trovare i pesi che realizzano il minimo di questa funzione. Per trovare tale minimo, si utilizza l'algoritmo di *discesa del gradiente*, in cui si segue la direzione opposta del gradiente dell'errore rispetto ai pesi. Di conseguenza l'aggiornamento dei pesi segue la regola

$$\begin{aligned} \omega_{k,j}^{NEW} &= \omega_{k,j}^{OLD} + \Delta_{k,j} \\ &= \omega_{k,j}^{NEW} - \eta \frac{\partial E}{\partial \omega_{k,j}} \end{aligned} \quad (1.27)$$

$$\begin{aligned} W_{j,i}^{NEW} &= W_{j,i}^{OLD} + \Delta_{j,i} \\ &= W_{j,i}^{NEW} - \eta \frac{\partial E}{\partial W_{j,i}} \end{aligned} \quad (1.28)$$

dove  $\eta$  è il parametro *learning rate*, e i pesi del secondo strato utilizzati per la equazione(1.28) sono quelli aggiornati calcolati con l'equazione (1.27).

Nel paragrafo che segue, andiamo a studiare le funzioni di attivazione più utilizzate a seconda del problema considerato e come vengono generati i conseguenti output dalla rete.

**Funzioni di attivazione.** Tramite queste funzioni non solo si cerca di emulare il funzionamento del cervello, ma è anche possibile inserire delle non-linearità nel modello che permettendo di apprendere funzioni più complesse. La forma delle *activation functions* dipende dal layer considerato e dal tipo di problema studiato. Per gli hidden layer, si considerano solitamente le seguenti funzioni [5]:

- la funzione *Heaviside*

$$\sigma(x) = \begin{cases} 1 & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

fa passare l'informazione con stesso peso, è non differenziabile in  $x = 0$  e alcuni neuroni potrebbero non essere attivati;

- la funzione *ReLU (Rectified Linear Unit)*

$$ReLU(x) = \max\{0, x\} \in [0, +\infty)$$

è non differenziabile in  $x = 0$ , può non attivare alcuni neuroni ma mantiene l'importanza dell'informazione che passa;

- la funzione *tangente iperbolica*

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1)$$

è differenziabile e attiva tutti i neuroni, anche se può portare al problema del vanishing gradient perchè ammette valori molto piccoli.

Per quanto riguarda l'output layer, la funzione di attivazione viene scelta a seconda del problema che la rete deve affrontare [5]:

- per i problemi di classificazione binaria si è soliti utilizzare la funzione *sigmoide*, la cui formula è data da

$$\sigma(x) = \frac{1}{1 + e^{-x}} \in (0, 1). \quad (1.29)$$

Questa funzione ha il vantaggio di essere continua e differenziabile e di poter approssimare una funzione a gradino con qualsiasi grado di precisione. In un contesto di classificazione binaria, lo spazio degli output viene solitamente inteso come l'insieme delle etichette positive, rappresentate dal numero intero 1, e delle etichette negative, rappresentate con 0. Dunque la sigmoide ha una forte interpretazione probabilistica, in quanto rappresenta la probabilità della previsione di essere nella classe dei positivi;

- nel caso di classificazione multi-classe, l'ultimo strato della rete viene completato con la funzione *softmax*, che misura la probabilità che

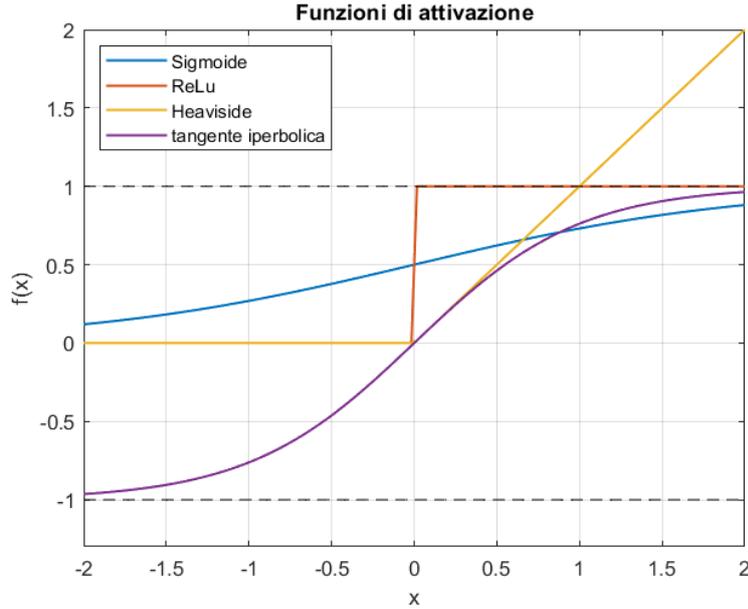


Figura 1.8: Funzioni di attivazione

un input appartenga a ciascuna delle classi possibili. Se il problema comprende  $K$  classi, per ciascuna classe  $i$  si calcola

$$\hat{y}_i = \textit{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (1.30)$$

dove  $z_i$  è il logit (o il punteggio non normalizzato) associato alla classe  $i$ . Si osserva che ciascuna di queste componenti rappresenta effettivamente una probabilità sulle classi, infatti  $\hat{y}_i \in (0, 1)$  per  $i = 1, \dots, k$  e

$$z_1 + \dots + z_k = 1.$$

### 1.3 Loss functions

In questa sezione, introduciamo le loss functions che vengono maggiormente utilizzate per l'addestramento del modello nel caso da noi studiato, cioè per i problemi di classificazione binaria e multiclasse. Queste funzioni misurano la discrepanza tra la predizione del modello e la real label del dato considerato. Il design di queste funzioni rappresenta un aspetto cruciale nella teoria del machine learning e tale scelta è legata al tipo di problema che la rete deve risolvere.

### 1.3.1 Loss functions per la classificazione binaria

Lo spazio degli output è indicato come l'insieme delle etichette positive, rappresentate dal numero intero 1, e delle etichette negative, rappresentate con 0. La funzione di attivazione maggiormente usata in questo problema è la *sigmoide* (1.29), che produce come output label dei numeri reali compresi tra 0 e 1 che rappresentano la probabilità delle previsioni di essere nella classe dei positivi. L'idea che vedremo dietro alle loss functions è quella di penalizzare le previsioni che non sono corrette.

**Binary cross-entropy.** La binary cross-entropy deriva dal concetto di entropia, che misura l'incertezza associata a una variabile casuale [5]. In questo senso, la cross-entropy estende questo concetto e misura l'incertezza tra due distribuzioni di probabilità, quella reale e quella prevista. Detta  $y \in \{0, 1\}$  l'etichetta reale e  $\hat{y} \in (0, 1)$  il label predetto, la binary cross-entropy si misura come

$$\ell_{bc}(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \quad (1.31)$$

e si osserva facilmente che

- se  $y = 0$ , la 1.31 si riduce alla funzione  $-\log(1 - \hat{y})$ , che penalizza i valori di  $\hat{y}$  più vicini a uno;
- se  $y = 1$ , la 1.31 si riduce alla funzione  $-\log(\hat{y})$ , che penalizza i valori di  $\hat{y}$  più vicini a zero.

**Score-Oriented Loss.** La *Score-Oriented Loss (SOL)* è stata introdotta per considerare, all'interno del problema di ottimizzazione dell'errore, il risultato della classificazione della rete attraverso determinate *metriche* o *skill scores* [14], punteggi associati al modello che permettono di dare una valutazione quantitativa della bontà del modello stesso nel task assegnato. La massimizzazione di questi scores non viene considerata durante il training dal momento che tali scores sono tipicamente rappresentati da funzioni discontinue rispetto alle predizioni del modello. L'idea dietro alla SOL function è quella di considerare un *threshold* che influenzi gli scores, in modo da trattarli non come valori fissi, ma piuttosto come variabili random. Consideriamo  $\mathcal{S} = \mathcal{S}(\mathbf{w}) = \{\hat{y}_i, y_i\}_{i=1}^n$  l'insieme predizioni-label, dove le predizioni  $\hat{y}_i$  sono calcolate dalla rete utilizzando il set di pesi  $\mathbf{w}$ , mentre  $y_i$  sono i veri label. Data una *soglia di threshold*

$\tau \in (0, 1)$ , la labels del campione  $i$ -esimo, che indichiamo con  $\bar{y}_i$ , è considerata positiva se  $\hat{y}_i \geq \tau$ , mentre è negativa se  $\hat{y}_i < \tau$ . Di conseguenza si definiscono

$$\begin{aligned} TN(\tau, \mathcal{S}) &= \sum_{i=1}^n (1 - y_i) \mathbb{1}_{\{\hat{y}_i < \tau\}} \\ &= \#\{i = 1 \dots n : y_i = 0 \wedge \hat{y}_i < \tau\} \end{aligned} \quad (1.32)$$

$$\begin{aligned} TP(\tau, \mathcal{S}) &= \sum_{i=1}^n y_i \mathbb{1}_{\{\hat{y}_i \geq \tau\}} \\ &= \#\{i = 1 \dots n : y_i = 1 \wedge \hat{y}_i \geq \tau\} \end{aligned} \quad (1.33)$$

$$\begin{aligned} FN(\tau, \mathcal{S}) &= \sum_{i=1}^n y_i \mathbb{1}_{\{\hat{y}_i < \tau\}} \\ &= \#\{i = 1 \dots n : y_i = 1 \wedge \hat{y}_i < \tau\} \end{aligned} \quad (1.34)$$

$$\begin{aligned} FP(\tau, \mathcal{S}) &= \sum_{i=1}^n (1 - y_i) \mathbb{1}_{\{\hat{y}_i \geq \tau\}} \\ &= \#\{i = 1 \dots n : y_i = 0 \wedge \hat{y}_i \geq \tau\} \end{aligned} \quad (1.35)$$

rispettivamente i True Negative, True Positive, False Negative e False Positive dipendenti dalla soglia  $\tau$ , con  $\mathbb{1}$  funzione indicatrice sul sottoinsieme definito. Inoltre chiamiamo **matrice di confusione** con threshold  $\tau \in (0, 1)$  la matrice

$$CM_\tau = CM(\tau, \mathcal{S}) = \begin{pmatrix} TN(\tau, \mathcal{S}) & FP(\tau, \mathcal{S}) \\ FN(\tau, \mathcal{S}) & TP(\tau, \mathcal{S}) \end{pmatrix} \quad (1.36)$$

Supponiamo che  $\tau$  sia una variabile casuale continua la cui densità di probabilità  $f$  abbia supporto in  $[a, b] \in [0, 1]$ . Inoltre denotiamo con  $F$  la funzione di densità cumulativa, cioè

$$F(x) = \int_a^x f(z) dz, \quad x \leq b \quad (1.37)$$

da cui

$$\mathbb{E}_\tau[\mathbb{1}_{x > \tau}] = F(x). \quad (1.38)$$

Sotto queste ipotesi, definiamo la **matrice di confusione attesa**

$$\overline{CM}_F = \overline{CM}_F(\mathcal{S}) = \begin{pmatrix} \overline{TN}_F(\mathcal{S}) & \overline{FP}_F(\mathcal{S}) \\ \overline{FN}_F(\mathcal{S}) & \overline{TP}_F(\mathcal{S}) \end{pmatrix} \quad (1.39)$$

con

$$\begin{aligned}
\overline{\text{TN}}_F(\mathcal{S}) &= \sum_{i=1}^n (1 - y_i)(1 - F(\hat{y}_i)) \\
\overline{\text{TP}}_F(\mathcal{S}) &= \sum_{i=1}^n y_i F(\hat{y}_i) \\
\overline{\text{FN}}_F(\mathcal{S}) &= \sum_{i=1}^n y_i (1 - F(\hat{y}_i)) \\
\overline{\text{FP}}_F(\mathcal{S}) &= \sum_{i=1}^n (1 - y_i) F(\hat{y}_i).
\end{aligned} \tag{1.40}$$

Si osserva che

$$\mathbb{E}_\tau[\text{CM}_\tau] = \overline{\text{CM}}_F \tag{1.41}$$

Definiamo **skill score** una mappa  $s : M_{2,2}(\mathbb{N}) \rightarrow \mathbb{R}$  costruita su  $\text{CM}(\tau, \mathcal{S})$  per un valore fissato di  $\tau \in (0, 1)$ , che indichiamo con  $s_\tau = s_\tau(\mathcal{S})$ . Detta  $\overline{s}_F = \overline{s}_F(\mathcal{S})$  la skill score corrispondente calcolata su  $\overline{\text{CM}}_F$ , la **Score-Oriented Loss** associata allo score  $s_\tau$  è la funzione

$$\ell_{\overline{s}}(F, \mathcal{S}) := -\overline{s}_F(\mathcal{S}) \tag{1.42}$$

dove ricordiamo  $\mathcal{S} = \mathcal{S}(\mathbf{w})$  dipende dal set di pesi della rete. Più avanti definiremo gli skill scores più utilizzati per la valutazione della bontà del modello a seconda del problema.

### 1.3.2 Loss functions per la classificazione multiclasse

Nel caso di classificazione multiclasse si propone una generalizzazione del caso binario. A partire dalla funzione di attivazione, l'ultimo strato della rete viene completato con la funzione *softmax* (1.30). L'output della rete è un vettore  $\hat{\mathbf{y}} \in (0, 1)^K$ , dove  $K$  è il numero delle classi totali, e ciascun elemento  $\hat{y}_i$  rappresenta la probabilità che l'elemento appartenga alla classe  $i$ . Per i label effettivi, si considera il one-hot encoding delle etichette che la rete deve imparare: supponendo dunque di avere  $K$  classi, ogni label reale  $y \in \{0, \dots, K\}$  viene trasformato in un vettore  $\mathbf{y} = [y_1, y_2, \dots, y_K]$  dove

$$y_i = \begin{cases} 1 & \text{se } y \text{ appartiene alla classe } i \\ 0 & \text{altrimenti} \end{cases}.$$

Detto  $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_k]$  il vettore delle probabilità previste dal modello per ciascuna classe, la categorical cross-entropy ha la seguente forma

$$\ell_{cc}(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=1}^k y_j \log(\hat{y}_j) \quad (1.43)$$

e cerca di calcolare la dissimilarità tra le probabilità predette dal modello  $\hat{\mathbf{y}}$  e quella reale  $\mathbf{y}$ . L'idea dietro la funzione è la stessa del caso binario: se l'input considerato appartiene alla classe  $i$ , l'unica entrata non nulla è  $y_i = 1$  e la (1.43) si riduce alla funzione  $-\log(\hat{y}_i)$ , dunque se il modello predice una bassa probabilità per la corrispondente classe, cioè  $\hat{y}_i$  è molto piccolo, la perdita sarà alta, penalizzando il modello. D'altra parte, se la previsione risulta corretta, la perdita sarà bassa.

## 1.4 Metriche di valutazione

Le metriche di valutazione sono strumenti fondamentali per misurare la bontà del modello nel compito prefissato e per studiare quanto le previsioni siano accurate sui dati di test o di validazione. Tali metriche, infatti, rappresentano un modo quantitativo per valutare la performance del modello rispetto agli obiettivi e alle aspettative del problema affrontato. Supponendo di avere un dataset con  $n$  campioni, consideriamo  $\mathcal{S} = \mathcal{S}(\mathbf{w}) = \{\bar{y}_i, y_i\}_{i=1}^n$  l'insieme predizioni-label, dove  $y_i$  sono i label reali, mentre gli output della rete  $\bar{y}_i$  sono calcolati utilizzando il set di pesi  $\mathbf{w}$  e una soglia fissata di threshold. Per calcolare la bontà della predizione si considera la **matrice di confusione**

$$\text{CM} = \begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix} \quad (1.44)$$

dove  $TN, TP, FN$  e  $FP$  sono rispettivamente i *true negative*, *true positive*, *false positive* e *false negative*, che si calcolano nel modo seguente

$$\begin{aligned} TN = TN(\mathcal{S}) &= \#\{i = 1, \dots, n : y_i = \bar{y}_i = 0\} \\ &= \sum_{i=1}^n (1 - y_i)(1 - \bar{y}_i) \end{aligned} \quad (1.45)$$

$$\begin{aligned} TP = TP(\mathcal{S}) &= \#\{i = 1, \dots, n : y_i = \bar{y}_i = 1\} \\ &= \sum_{i=1}^n y_i \bar{y}_i \end{aligned} \quad (1.46)$$

$$\begin{aligned}
FN &= FN(\mathcal{S}) = \#\{i = 1, \dots, n : y_i = 1, \bar{y}_i = 0\} \\
&= \sum_{i=1}^n y_i(1 - \bar{y}_i)
\end{aligned} \tag{1.47}$$

$$\begin{aligned}
FP &= FP(\mathcal{S}) = \#\{i = 1, \dots, n : y_i = 0, \bar{y}_i = 1\} \\
&= \sum_{i=1}^n (1 - y_i)\bar{y}_i.
\end{aligned} \tag{1.48}$$

Le metriche di valutazione sono funzioni definite sulla matrice di confusione che permettono di fornire una misura quantitativa sull'efficacia del modello nel predire gli output. Nel caso di classificazione binaria [20] si definiscono le seguenti:

- l' **Accuracy** è il rapporto tra il numero di previsioni corrette e il numero di previsioni totali, cioè

$$ACC(\mathcal{S}) = \frac{TP + TN}{n} \in [0, 1]; \tag{1.49}$$

- la **Recall** (Sensitivity o True Positive Rate) misura la capacità di un modello di identificare correttamente tutte le previsioni positive (veri positivi) rispetto a tutti i positivi

$$Recall(\mathcal{S}) = \frac{TP}{TP + FN} \in [0, 1]; \tag{1.50}$$

- la **Specificity** (True Negative Rate) misura la capacità di un modello di identificare correttamente tutte le previsioni negative rispetto a tutte le istanze negative

$$Specificity(\mathcal{S}) = \frac{TN}{TN + FP} \in [0, 1]; \tag{1.51}$$

- la **Precision** misura la precisione delle predizioni positive fatte dal modello ed è la proporzione di veri positivi rispetto a tutti i positivi predetti dal modello

$$Precision(\mathcal{S}) = \frac{TP}{TP + FP} \in [0, 1]; \tag{1.52}$$

- l'**F1-score** rappresenta un equilibrio tra precision e recall, in particolare rappresenta la media armonica tra precision e recall così definita

$$F1(\mathcal{S}) = 2 \frac{Precision * Recall}{Precision + Recall} \in [0, 1] \tag{1.53}$$

- la **True Skill Statistic** (TSS) è una metrica utile nel caso di dataset sbilanciati per la sua capacità di discriminazione, dal momento che tiene conto sia del true positive rate che del true negative rate del modello:

$$\begin{aligned} TSS(\mathcal{S}) &= \textit{sensitivity} + \textit{specificity} - 1 \\ &= \frac{TP}{TP + FN} + \frac{TN}{TN + FP} - 1 \in [-1, 1]; \end{aligned} \quad (1.54)$$

- anche la **Balanced accuracy** è utile quando le classi di output sono sbilanciate nel dataset ed è la media delle sensitivity (recall) e specificity.

$$\textit{balanced accuracy}(\mathcal{S}) = \frac{\textit{sensitivity} + \textit{specificity}}{2} \in [0, 1] \quad (1.55)$$

- l' **Heidke Skill Score** (HSS) è una metrica che valuta quanto un modello è migliore rispetto a un modello casuale

$$HSS(\mathcal{S}) = 2 \frac{TP * TN - FP * FN}{(TP + FN) * (FN + TN) + (TP + FP) * (FP + TN)} \in [-1, 1] \quad (1.56)$$

- il **Critical Success Index** (CSI) viene utilizzata soprattutto in contesti ambientali per valutare la capacità di un modello di previsione di identificare correttamente gli eventi positivi .

$$CSI(\mathcal{S}) = \frac{TP}{TP + FN + FP} \in [0, 1] \quad (1.57)$$

Questa metrica misura la proporzione di eventi positivi previsti correttamente rispetto a tutti gli eventi previsti o osservati come positivi, dunque, rispetto alla Recall, tiene conto anche dei falsi positivi, fornendo una misura più equilibrata delle prestazioni di un modello in contesti in cui la sovrastima dei positivi risulta un problema.

Il caso multiclasse si tratta in modo molto simile, estendendo la modalità con cui si definisce la matrice di confusione [10], mentre le metriche su di essa possono essere definite in due modi distinti. Supponendo di avere  $K$  classi, si considerano sulle righe i label predetti e sulle colonne i real label, da cui la matrice di confusione  $CM$  è una matrice di dimensione  $K \times K$  le cui entrate sono

$$(CM)_{i,j} = \#\{k = 1, \dots, n : \bar{y}_k = i, y_k = j\}.$$

Si osserva facilmente che sulla diagonale si trovano i true positive per ciascuna classe, cioè i label predetti correttamente, mentre le altre entrate contengono

i conteggi dei label predetti non correttamente. Si converte questa matrice  $K \times K$  in  $K$  matrici di dimensione  $2 \times 2$  che rappresentano ciascuna la matrice di confusione per ciascuna classe, aggregando i true negative, false positive e false negative. Per ciascuna classe  $k = 1 \dots K$  si ottiene una matrice di confusione

$$CM_k = \begin{pmatrix} TN_k & FP_k \\ FN_k & TP_k \end{pmatrix} \quad (1.58)$$

con

$$\begin{aligned} TP_k &= (CM)_{k,k} \\ FP_k &= \#\{i = 1, \dots, n : \bar{y}_i = k, y_i \neq k\} = \sum_{\substack{j=1, \\ j \neq k}}^K (CM)_{k,j} \\ FN_k &= \#\{i = 1, \dots, n : \bar{y}_i \neq k, y_i = k\} = \sum_{\substack{i=1, \\ i \neq k}}^K (CM)_{i,k} \\ TN_k &= \sum_{\substack{i,j=1, \\ i,j \neq k}}^K (CM)_{i,j}. \end{aligned} \quad (1.59)$$

Possiamo definire i due metodi per il calcolo delle metriche:

- la **Macro-media** consiste nel calcolare una determinata metrica per ciascuna classe, per poi considerarne la media (media per tipologia); in particolare si calcola il punteggio  $score(k)$  relativo a ciascuna classe  $k = 1, \dots, K$  sulla matrice di confusione  $CM_k$  e se ne considera la media

$$macro - averaging\ score = \frac{1}{K} \sum_{k=1}^K score(k); \quad (1.60)$$

- la **Micro-media** aggrega i contributi di tutte le classi per poi calcolare le metriche globali. In questo caso si sommano tutti i true positive, false positive e false negative di tutte le classi e poi calcola le metriche basate su questi totali. Si definiscono

$$\begin{aligned} \Sigma TP &= \sum_{k=1}^K TP_k \\ \Sigma TN &= \sum_{k=1}^K TN_k \\ \Sigma FN &= \sum_{k=1}^K FN_k \\ \Sigma FP &= \sum_{k=1}^K FP_k \end{aligned} \quad (1.61)$$

e si calcolano le metriche definite per il caso binario con questi valori:

$$\text{micro-averaging score} = \text{score}(\Sigma\text{CM}), \quad (1.62)$$

dove  $\Sigma\text{CM}$  è la matrice

$$\Sigma\text{CM} = \begin{pmatrix} \Sigma TP & \Sigma FN \\ \Sigma FP & \Sigma TN \end{pmatrix}$$

Quindi, nel micro-averaging tutte le predizioni vengono trattate come un singolo insieme, aggregando true positive, true negative, false positive e false negative per calcolare le metriche sui totali: ne consegue che tale metodo è più sensibile alle classi più numerose, perché queste sono pesate in base al numero di campioni che le rappresenta. Tale metrica può essere utile nel caso in cui si vuole enfatizzare il comportamento del modello rispetto alla classe maggioritaria. D'altra parte, nel macro-averaging le metriche vengono calcolate per ciascuna classe separatamente, e poi si calcola la loro media aritmetica: in questo caso tutte le classi sono trattate allo stesso modo, senza considerare che alcune potrebbero essere maggiormente rappresentate. Questa metrica è consigliabile quando si vuole avere una visione complessiva del comportamento del modello su tutte le classi, anche quelle meno rappresentate. Di conseguenza, nel caso di dataset sbilanciati, si ottengono risultati migliori con la micro-averaging piuttosto che la macro-averaging, perché è sufficiente che il modello si adatti bene alla classe maggioritaria per ottenere alte performance. Nel nostro caso, quindi, andremo a considerare maggiormente la macro-averaging, per comprendere le abilità del modello a predire anche le classi minoritarie.

*Osservazione:* si segnala che la macro accuracy non sarà calcolata con la formula (1.60), ma con la sua definizione, cioè come rapporto delle previsioni corrette sulle previsioni totali. Riprendendo le notazioni della matrice di confusione del caso multiclasse, la macro accuracy è data dal rapporto

$$\frac{\sum_{i=1}^K CM_{i,i}}{\sum_{i,j=1}^K CM_{i,j}}$$



# Capitolo 2

## Convergenza del Transformer

In questo capitolo si vuole riportare un risultato di convergenza del Transformer [22] in un caso particolare, in cui il modello è costituito dal solo Encoder composto dai 3 layers: un primo livello di *self-attention*, un *feed-forward ReLU* layer e un layer finale di classificazione per il calcolo dell'output. In particolare, si fa riferimento al fatto che un' over-parametrizzazione di tipo cubico è sufficiente per la convergenza globale del modello sotto le ipotesi di inizializzazione dei pesi di He o LeCun [22], frequentemente adottate nella pratica. Infine si discute il fenomeno del *bias-variance trade-off* e si accenna brevemente al nuovo problema della *double-descent curve*.

### 2.1 Teorema di Convergenza

In primo luogo si vogliono riprendere le notazioni del capitolo precedente per le equazioni del Transformer e aggiungerne di nuove. Per un generico layer, siano

- $\mathbf{X} = \{x_n \in \mathbb{R}^{d \times T}\}_{n=1}^N$  l'insieme degli input;
- $\mathbf{Y} = \{y_n\}_{n=1}^N \in \mathbb{R}^N$  l'insieme degli output;
- $W_q \in \mathbb{R}^{d \times k}$  la matrice dei pesi delle query;
- $W_k \in \mathbb{R}^{d \times k}$  la matrice dei pesi delle keys;
- $W_v \in \mathbb{R}^{d \times k}$  la matrice dei pesi dei values.

Il modello di cui vogliamo mostrare la convergenza è un Transformer formato da un solo Encoder costituito dai seguenti layers [22]:

- (i) un primo layer che implementa il meccanismo dell'attenzione secondo l'equazione (1.8), il cui output e la cui funzione sono denominati come segue

$$\begin{aligned} A_n &:= \text{Attention}(x_n) = \\ &= \text{softmax}\left(\tau_0(x_n^T W_q)(W_k x_n)\right)(x_n W_v)^T \in \mathbb{R}^{T \times k} \end{aligned} \quad (2.1)$$

con  $\tau_0 = \frac{1}{\sqrt{k}}$ ,  $n = 1, \dots, N$ ;

- (ii) un layer di attivazione che implementa la *ReLU*,

$$Z_n := \tau_1 \text{ReLU}(A_n) \in \mathbb{R}^{T \times k} \quad (2.2)$$

con  $\tau_1$  coefficiente di scala;

- (iii) un layer che implementa l'*average pooling* lungo le colonne di  $Z_n$ ,

$$P_n := \varphi(Z_n) \in \mathbb{R}^{1 \times k}, \quad (2.3)$$

dove  $\varphi$  è la funzione così definita

$$\varphi(Z_n) = \begin{pmatrix} \sum_{i=1}^T (Z_n)_{i,1} \\ \vdots \\ \sum_{i=1}^T (Z_n)_{i,j} \\ \vdots \\ \sum_{i=1}^T (Z_n)_{i,k} \end{pmatrix} \quad (2.4)$$

- (iv) un layer finale che calcola l'output del modello con una matrice di pesi apprendibili  $W_o \in \mathbb{R}^{1 \times k}$ . Indichiamo con

$$f(x_n, \theta) = W_o P_n^T \in \mathbb{R}, \quad n = 1, \dots, N \quad (2.5)$$

la funzione che implementa l'intero modello e  $\theta$  è il vettore contenente tutti i pesi apprendibili.

Infine definiamo

$$\mathbf{f}(\theta) := \{f(x_n, \theta)\}_{n=1}^N \in \mathbb{R}^N \quad (2.6)$$

l'insieme degli output del modello calcolati con parametri  $\theta$  e con

$$\mathbf{F}_{pre}(\theta) := \{P_n(\theta)\}_{n=1}^N \in \mathbb{R}^{N \times k} \quad (2.7)$$

l'insieme degli output dell'ultimo hidden layer calcolato con parametri  $\theta$ .

La funzione di perdita che consideriamo è la *Squared-loss function*, che calcoliamo su tutto il set di campioni e indichiamo come

$$\ell(\theta) = \frac{1}{2} \|\mathbf{f}(\theta) - \mathbf{Y}\|^2. \quad (2.8)$$

L'*algoritmo di discesa del gradiente* si sviluppa con le seguenti regole di aggiornamento

- al passo  $t = 0$  si inizializzano i pesi

$$\theta^0 = \{W_q^0, W_k^0, W_v^0, W_o^0\};$$

- al passo  $t > 1$ , detto  $\gamma$  lo step-size dell'algoritmo, l'aggiornamento implementa le seguenti regole

$$\begin{aligned} W_q^{t+1} &= W_q^t - \gamma \cdot \nabla_{W_q} \ell(\theta^t) \\ W_k^{t+1} &= W_k^t - \gamma \cdot \nabla_{W_k} \ell(\theta^t) \\ W_v^{t+1} &= W_v^t - \gamma \cdot \nabla_{W_v} \ell(\theta^t) \\ W_o^{t+1} &= W_o^t - \gamma \cdot \nabla_{W_o} \ell(\theta^t) \end{aligned} \quad (2.9)$$

**Definizione 2.1.1.** Sia  $A \in \mathbb{R}^{n \times m}$  una matrice generica. Si definisce *Norma di Frobenius* di  $A$  la quantità

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |A_{i,j}|^2} \quad (2.10)$$

**Teorema 2.1.2** (Teorema di Convergenza[22]). Consideriamo il modello del Transformer sopra descritto con  $k \geq N$ . Supponiamo che l'insieme degli input  $\mathbf{X}$  sia limitato in *norma di Frobenius*, cioè che esista una costante positiva  $C_x$  tale che

$$\|\mathbf{X}\|_F \leq \sqrt{T} C_x \quad (2.11)$$

dove  $T$  è la lunghezza della sequenza.

Siano  $C_q, C_k, C_v$  e  $C_o$  costanti positive e definiamo le seguenti quantità al passo iniziale

$$\begin{aligned} \bar{\lambda}_q &:= \|W_q^0\|_2 + C_q \\ \bar{\lambda}_k &:= \|W_k^0\|_2 + C_k \\ \bar{\lambda}_v &:= \|W_v^0\|_2 + C_v \\ \bar{\lambda}_o &:= \|W_o^0\|_2 + C_o \end{aligned} \quad (2.12)$$

Definiamo

$$\rho := N^{\frac{1}{2}} T^{\frac{3}{2}} \tau_1 C_x; \quad (2.13)$$

$$z := \bar{\lambda}_o^2 (1 + 4\tau_o^2 C_x^4 T^2 \bar{\lambda}_v^2 (\bar{\lambda}_q^2 + \bar{\lambda}_k^2)) \quad (2.14)$$

Supponiamo inoltre che, detto  $\alpha$  il minimo valore singolare di  $F_{pre}^0$ , cioè  $\alpha := \sigma_{min}(F_{pre}^0)$ , dove  $F_{pre}^0$  è l'output del modello calcolati con parametri  $\theta^0$ , allora questo soddisfa

$$\alpha^2 \geq 8\rho \max\{\bar{\lambda}_v C_o^{-1}, \bar{\lambda}_o C_v^{-1}, 2\tau_o C_x^2 T \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o C_q^{-1}, 2\tau_o C_x^2 T \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o C_k^{-1}\} \quad (2.15)$$

$$\alpha^3 \geq 32\rho^2 z \frac{\sqrt{2\ell(\theta^0)}}{\bar{\lambda}_o}$$

Se lo step-size della discesa del gradiente  $\gamma$  soddisfa

$$\gamma \leq \frac{1}{C}$$

per una certa costante  $C$  che dipende da  $(\bar{\lambda}_q, \bar{\lambda}_k, \bar{\lambda}_v, \bar{\lambda}_o, \rho, \tau, \ell(\theta^0))$ , allora l'algoritmo di discesa del gradiente 2.9 converge ad un minimo globale come segue:

$$\ell(\theta^t) \leq \left(1 - \gamma \frac{\alpha^2}{2}\right)^t \ell(\theta^0) \quad \forall t \geq 0. \quad (2.16)$$

**Osservazione 2.1.3.** Il parametro  $\alpha$  controlla la velocità di convergenza del modello. Lo step-size  $\gamma$  è proporzionale a  $N^{-1/2}$  per come è definita la costante  $C$  in 2.63.

## 2.2 Dimostrazione del Teorema di Convergenza

**Lemma 2.2.1** (Bound per la norma euclidea della softmax.). Dato  $v = \text{softmax}(u) \in \mathbb{R}^T$ , allora vale

$$\frac{1}{\sqrt{T}} \leq \|v\|_2 \leq 1.$$

*Dimostrazione.* Per la disequazione della media aritmetica e geometrica

$$\frac{1}{\sqrt{T}} = \frac{\sum_{i=1}^T v_i}{\sqrt{T}} \leq \|v\|_2 = \sqrt{\sum_{i=1}^T (v_i)^2} \leq \sqrt{\left(\sum_{i=1}^T v_i\right)^2} = 1.$$

□

**Lemma 2.2.2** (Jacobiano della Softmax). Sia  $v = \text{Softmax}(u) \in \mathbb{R}^T$ , allora  $\frac{\partial v}{\partial u} = \text{diag}(v) - vv^T$ , dove  $\text{diag}(v)$  è la matrice diagonale  $T \times T$  con sulla diagonale il vettore  $v$ .

*Dimostrazione.* Il vettore  $v$  ha componenti:

$$v = \begin{bmatrix} \frac{\exp(u_1)}{\sum_{i=1}^T \exp(u_i)} \\ \vdots \\ \frac{\exp(u_T)}{\sum_{i=1}^T \exp(u_i)} \end{bmatrix}.$$

Di conseguenza, le derivate parziali si calcolano come

$$\begin{aligned} \frac{\partial v_j}{\partial u_k} &= \frac{\partial \frac{\exp(u_j)}{\sum_{i=1}^T \exp(u_i)}}{\partial u_k} = \begin{cases} \frac{-\exp(u_j) - \exp(u_k)}{(\sum_{i=1}^T \exp(u_i))^2} & \text{se } j \neq k \\ \frac{\exp(u_k) \sum_{i=1}^T \exp(u_i) - (\exp(u_k))^2}{(\sum_{i=1}^T \exp(u_i))^2} & \text{se } j = k \end{cases} \\ &= \begin{cases} -v_j v_k & \text{se } j \neq k \\ v_k - v_j v_k & \text{se } j = k \end{cases} \end{aligned}$$

Dunque

$$\frac{\partial v}{\partial u} = \text{diag}(v) - vv^T.$$

□

**Lemma 2.2.3.** Per due generici vettori  $v, u \in \mathbb{T}$ ,

$$\|\text{softmax}(v) - \text{softmax}(u)\|_1 \leq 2 \|u - v\|_\infty \quad (2.17)$$

Poiché la dimostrazione del lemma 2.2.3 non è essenziale per la discussione principale di questa tesi, essa viene omessa. Per una trattazione dettagliata, si rimanda all'articolo [9].

**Lemma 2.2.4.** Data una generica matrice  $A \in \mathbb{R}^{n \times m}$ , valgono le seguenti:

1.  $\|A\|_2 \leq \|A\|_F$ ;
2.  $\|A\|_2 \leq \|A\|_1$
3. se  $n = 1$  oppure  $m = 1$ , allora  $\|A\|_2 = \|A\|_F$ .

**Lemma 2.2.5.** Definiamo

$$\beta_i = \text{softmax}\left(\tau_0\left((\mathbf{X}^{(:,i)})^T W_q\right)(W_k^T \mathbf{X})\right) \in \mathbb{R}^{(1 \times T) \times N}$$

l' $i$ -esima riga dell'output della Softmax, dove  $\mathbf{X}^{(:,i)}$  indica un elemento di  $\mathbf{X}$  lungo la dimensione delle features, con  $i = 1, \dots, T$ . Siano  $t$  e  $t'$  due tempi diversi dell'algoritmo di discesa del gradiente e supponiamo che

$$\begin{aligned} \max\left(\|W_q^t\|_2, \|W_q^{t'}\|_2\right) &\leq \bar{\lambda}_q, \\ \max\left(\|W_k^t\|_2, \|W_k^{t'}\|_2\right) &\leq \bar{\lambda}_k, \\ \max\left(\|W_v^t\|_2, \|W_v^{t'}\|_2\right) &\leq \bar{\lambda}_v, \\ \max\left(\|W_o^t\|_2, \|W_o^{t'}\|_2\right) &\leq \bar{\lambda}_o \end{aligned}$$

Allora si dimostrano i seguenti limiti superiori:

(i) per la *softmax*:

$$\left\|\beta_i^t - \beta_i^{t'}\right\|_2 \leq 2\tau_0 C_x^2 T (\bar{\lambda}_q \left\|W_k^t - W_k^{t'}\right\|_2 + \bar{\lambda}_k \left\|W_q^t - W_q^{t'}\right\|_2) \quad (2.18)$$

(ii) per l' output dell'ultimo hidden layer:

$$\begin{aligned} \left\|\mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t\right\|_F &\leq \rho \left( \left\|W_v^{t'} - W_v^t\right\|_2 + \right. \\ &\quad \left. \bar{\lambda}_v 2\tau_0 C_x^2 k (\bar{\lambda}_q \left\|W_k^{t'} - W_k^t\right\|_2 + \bar{\lambda}_k \left\|W_q^{t'} - W_q^t\right\|_2) \right), \end{aligned} \quad (2.19)$$

dove  $\rho$  è definita in (2.13);

(iii) per l' output della rete:

$$\left\|\mathbf{f}^{t'} - \mathbf{f}^t\right\|_2 \leq \rho \bar{\lambda}_v \left\|W_o^t - W_o^{t'}\right\|_2 + \bar{\lambda}_o \left\|\mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t\right\|_F. \quad (2.20)$$

*Dimostrazione.*

$$\begin{aligned} (i) \quad \left\|\beta_i^{t'} - \beta_i^t\right\|_2 &\leq \left\|\beta_i^{t'} - \beta_i^t\right\|_1 \\ &\stackrel{(a)}{\leq} 2 \left\|\tau_0(\mathbf{X}^{(:,i)})^T W_q^{t'} (W_k^{t'})^T \mathbf{X} - \tau_0(\mathbf{X}^{(:,i)})^T W_q^t (W_k^t)^T \mathbf{X}\right\|_\infty \\ &= 2 \max_j \left| \tau_0(\mathbf{X}^{(:,i)})^T W_q^{t'} (W_k^{t'})^T \mathbf{X}^{(:,j)} - \tau_0(\mathbf{X}^{(:,i)})^T W_q^t (W_k^t)^T \mathbf{X}^{(:,j)} \right| \\ &\stackrel{(b)}{\leq} 2 \max_j \left( \tau_0 \left\|\mathbf{X}^{(:,i)}\right\|_2 \left\|W_q^{t'} (W_k^{t'})^T - W_q^t (W_k^t)^T\right\|_2 \left\|\mathbf{X}^{(:,j)}\right\|_2 \right) \\ &\stackrel{(c)}{\leq} 2\tau_0 C_x^2 T \left\|W_q^{t'} (W_k^{t'})^T - W_q^t (W_k^t)^T\right\|_2 \\ &\stackrel{(d)}{\leq} 2\tau_0 C_x^2 T \left( \left\|W_q^{t'}\right\|_2 \left\|W_k^{t'} - W_k^t\right\|_2 + \left\|W_k^t\right\|_2 \left\|W_q^{t'} - W_q^t\right\|_2 \right) \\ &\leq 2\tau_0 C_x^2 T (\bar{\lambda}_q \left\|W_k^{t'} - W_k^t\right\|_2 + \bar{\lambda}_k \left\|W_q^{t'} - W_q^t\right\|_2). \end{aligned}$$

dove (a) segue dal lemma 2.2.1, (b) per la disuguaglianza di Cauchy-Schwartz, (c) per l'ipotesi sul dato nel teorema 2.1.2, (d) dalla Cauchy-Schwartz e dalla disuguaglianza triangolare aggiungendo e togliendo la quantità  $W_q^{t'}(W_v^t)$ , mentre l'ultima segue dalle ipotesi.

(ii) Dimostriamo la disuguaglianza (2.19) per ogni singolo esempio, cioè per  $f_{pre}^n \in \mathbb{R}^{1 \times k}$ ,  $n = 1, \dots, N$ , con

$$\mathbf{F}_{pre} = \begin{bmatrix} f_{pre}^1 \\ \vdots \\ f_{pre}^N \end{bmatrix} \in \mathbb{R}^{N \times k} \quad (2.21)$$

e ciascun esempio è dato da

$$\begin{aligned} f_{pre}^n &= \varphi\left(\tau_1 ReLU(A_n)\right) \\ &= \sum_{i=1}^T \tau_1 ReLU((A_n)_i) \end{aligned}$$

essendo  $\varphi$  la funzione che rappresenta l'average pooling per colonne.

Per semplicità trascuriamo l'indice  $n$  sugli esempi; si osserva che

$$\begin{aligned} \left\| f_{pre}^{t'} - f_{pre}^t \right\|_2 &= \tau_1 \left\| \sum_{i=1}^T ReLU(A_i(\theta^{t'})) - \sum_{i=1}^T ReLU(A_i(\theta^t)) \right\|_2 \\ &= \tau_1 \left\| \sum_{i=1}^T ReLU(\beta_i^{t'} \mathbf{X}^T W_v^{t'})^T - \sum_{i=1}^T ReLU(\beta_i^t \mathbf{X}^T W_v^t)^T \right\|_2 \\ &\stackrel{(a)}{\leq} \tau_1 \sum_{i=1}^T \left\| (\beta_i^{t'} \mathbf{X}^T W_v^{t'})^T - (\beta_i^t \mathbf{X}^T W_v^t)^T \right\|_2 \\ &\stackrel{(b)}{\leq} \tau_1 \sum_{i=1}^T \left( \left\| \mathbf{X}^T \beta_i^{t'} (W_v^{t'} - W_v^t) \right\|_2 + \left\| W_v^t \mathbf{X}^T (\beta_i^{t'} - \beta_i^t) \right\|_2 \right) \\ &\stackrel{(c)}{\leq} \tau_1 \sum_{i=1}^T \left( \left\| W_v^{t'} - W_v^t \right\|_2 \|\mathbf{X}\|_2 \left\| \beta_i^{t'} \right\|_2 + \left\| W_v^t \right\|_2 \|\mathbf{X}\|_2 \left\| \beta_i^{t'} - \beta_i^t \right\|_2 \right) \\ &\stackrel{(d)}{\leq} \tau_1 C_x \sqrt{T} \sum_{i=1}^T \left( \left\| W_v^{t'} - W_v^t \right\|_2 + \bar{\lambda}_v \left\| \beta_i^{t'} - \beta_i^t \right\|_2 \right) \\ &\stackrel{(e)}{\leq} \tau_1 C_x T^{3/2} \left( \left\| W_v^{t'} - W_v^t \right\|_2 + \bar{\lambda}_v 2\tau_0 C_x^2 T (\bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 \right. \\ &\quad \left. + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2 \right) \end{aligned} \quad (2.22)$$

dove (a) segue dalla continuità Lipschitziana della  $ReLU$ , (b) segue dalla disuguaglianza triangolare aggiungendo e togliendo la quantità  $\beta_i^{t'} \mathbf{X} W_v^t$ , (c) dalla Cauchy-Schwartz, (d) dal lemma 2.2.1 e dall'ipotesi sul dato del teorema

2.1.2, mentre (e) segue dal punto (i) sopra dimostrato. Considerando quindi la differenza su tutto il dataset

$$\begin{aligned} \left\| \mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t \right\|_F &\leq \sqrt{N} \tau_1 C_x T^{3/2} \left( \left\| W_v^{t'} - W_v^t \right\|_2 \right. \\ &\quad \left. + \bar{\lambda}_v 2\tau_0 C_x^2 T (\bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2) \right) \\ &= \rho \left( \left\| W_v^{t'} - W_v^t \right\|_2 + \bar{\lambda}_v 2\tau_0 C_x^2 T (\bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2) \right), \end{aligned}$$

dove l'ultima equazione segue dalla definizione di  $\rho$ .

(iii) Per la disuguaglianza triangolare

$$\begin{aligned} \left\| f^{t'} - f^t \right\|_2 &= \left\| \mathbf{F}_{pre}^{t'} W_o^{t'} - \mathbf{F}_{pre}^t W_o^t \right\|_2 \\ &= \left\| \mathbf{F}_{pre}^{t'} W_o^{t'} + \mathbf{F}_{pre}^t W_o^t - \mathbf{F}_{pre}^{t'} W_o^t - \mathbf{F}_{pre}^t W_o^{t'} \right\|_2 \\ &\leq \left\| \mathbf{F}_{pre}^{t'} \right\|_2 \left\| W_o^{t'} - W_o^t \right\|_2 + \left\| W_o^t \right\|_2 \left\| \mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t \right\|_2, \end{aligned}$$

Il primo termine può essere limitato usando la lipschitzianità della *ReLU* e i soliti bound nel modo seguente

$$\begin{aligned} \left\| \mathbf{F}_{pre}^{t'} \right\|_2 &\leq \left\| \mathbf{F}_{pre}^{t'} \right\|_F = \left\| \begin{bmatrix} \tau_1 \sum_{i=1}^T ReLU(\beta_{i,1}' x_1^T W_v^{t'}) \\ \vdots \\ \tau_1 \sum_{i=1}^T ReLU(\beta_{i,N}' x_N^T W_v^{t'}) \end{bmatrix} \right\|_F \\ &= \tau_1 \left\| \begin{bmatrix} \left\| \sum_{i=1}^T ReLU(\beta_{i,1}' x_1^T W_v^{t'}) \right\|_2 \\ \vdots \\ \left\| \sum_{i=1}^T ReLU(\beta_{i,N}' x_N^T W_v^{t'}) \right\|_2 \end{bmatrix} \right\|_2 \\ &\leq \tau_1 \left\| \begin{bmatrix} \left\| \sum_{i=1}^T \beta_{i,1}' x_1^T W_v^{t'} \right\|_2 \\ \vdots \\ \left\| \sum_{i=1}^T \beta_{i,N}' x_N^T W_v^{t'} \right\|_2 \end{bmatrix} \right\|_2 \\ &\leq \tau_1 \left\| \begin{bmatrix} T \|x_1\|_2^T \|W_v^{t'}\| \\ \vdots \\ T \|x_N\|_2^T \|W_v^{t'}\| \end{bmatrix} \right\|_2 \\ &\leq \tau_1 \sqrt{N} T^{3/2} C_x \left\| W_v^{t'} \right\|_2 = \rho \left\| W_v^{t'} \right\|_2, \end{aligned}$$

dove l'ultima equazione segue dalla definizione di  $\rho$ . Allora per le ipotesi sulle norme dei pesi segue

$$\begin{aligned} \left\| \mathbf{F}_{pre}^{t'} \right\|_2 &\leq \rho \left\| W_v^{t'} \right\|_2 \left\| W_o^{t'} - W_o^t \right\|_2 + \left\| W_o^t \right\|_2 \left\| \mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t \right\|_2 \\ &\leq \rho \bar{\lambda}_v \left\| W_o^{t'} - W_o^t \right\|_2 + \bar{\lambda}_o \left\| \mathbf{F}_{pre}^{t'} - \mathbf{F}_{pre}^t \right\|_2 \end{aligned}$$

□

**Osservazione 2.2.6.** Per calcolare i gradienti della funzione che rappresenta il modello rispetto alle diverse matrici di pesi possiamo scrivere la funzione  $f$  in diversi modi.

- Scrivendo

$$f(x_n) = W_o \varphi(\tau_1 \text{ReLU}(A_n)) = W_o f_{pre}^n$$

si osserva facilmente che

$$\begin{aligned} \nabla_{W_o} \ell(\theta) &= - \sum_{n=1}^N (f(x_n) - y_n) \frac{\partial f(x_n)}{\partial W_o} \\ &= -\tau_1 \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T \text{ReLU}(\beta_{i,n} x_n^T W_v)^T. \end{aligned} \quad (2.23)$$

- Scrivendo

$$f(x_n) = \tau_1 W_o \sum_{i=1}^T \text{ReLU}(\beta_i x_n^T W_v)^T,$$

si osserva che

$$\begin{aligned} \nabla_{W_v} \ell(\theta) &= - \sum_{n=1}^N (f(x_n) - y_n) \frac{\partial f(x_n)}{\partial W_v} \\ &= -\tau_1 \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T \left( (W_o^t)^T \circ (\text{ReLU}(\beta_{i,n} x_n^T W_v)^T) \right) \beta_{i,n} x_n^T. \end{aligned} \quad (2.24)$$

- Scrivendo

$$f(x_n) = \tau_1 W_o \sum_{i=1}^T \text{ReLU} \left( \text{softmax} \left( \tau_0 (x_n^{(:,i)})^T W_q W_k^T x_n \right) \mathbf{X}^T W_v \right)^T,$$

si osserva che

$$\begin{aligned} \nabla_{W_q} \ell(\theta) &= - \sum_{n=1}^N (f(x_n) - y_n) \frac{\partial f(x_n)}{\partial W_q} \\ &= -\tau_0 \tau_1 \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T (W_k)^T x_n (\text{diag}(\beta_{i,n}) - \beta_{i,n} \beta_{i,n}^T) \\ &\quad (x_n)^T W_v \left( (W_o^t)^T \circ \text{ReLU}(\beta_{i,n} x_n^T W_v)^T \right) x_n^{(:,i)} \end{aligned} \quad (2.25)$$

e

$$\begin{aligned} \nabla_{W_k} \ell(\theta) &= - \sum_{n=1}^N (f(x_n) - y_n) \frac{\partial f(x_n)}{\partial W_k} \\ &= -\tau_0 \tau_1 \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T W_q^T x_n (\text{diag}(\beta_{i,n}) - \beta_{i,n} \beta_{i,n}^T) \\ &\quad x_n^T W_v \left( (W_o^t)^T \circ \text{ReLU}(\beta_{i,n} x_n^T W_v)^T \right) x_n^{(:,i)}. \end{aligned} \quad (2.26)$$

**Lemma 2.2.7** (Bound per la norma del gradiente della loss). Supponiamo che al passo  $t$  valgano

$$\begin{aligned}\|W_q^t\|_2 &\leq \bar{\lambda}_q, \\ \|W_k^t\|_2 &\leq \bar{\lambda}_k, \\ \|W_v^t\|_2 &\leq \bar{\lambda}_v, \\ \|W_o^t\|_2 &\leq \bar{\lambda}_o,\end{aligned}$$

allora il gradiente della loss rispetto a  $W_q, W_k, W_v, W_o$  è limitato in norma dalle seguenti quantità:

$$\|\nabla_{W_q} \ell(\theta^t)\|_{\text{F}} \leq 2\rho\tau_0 \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o T C_x^2 \|\mathbf{f}^t - \mathbf{y}\|_2, \quad (2.27)$$

$$\|\nabla_{W_k} \ell(\theta^t)\|_{\text{F}} \leq 2\rho\tau_0 \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o T C_x^2 \|\mathbf{f}^t - \mathbf{y}\|_2, \quad (2.28)$$

$$\|\nabla_{W_v} \ell(\theta^t)\|_{\text{F}} \leq \rho \bar{\lambda}_o \|\mathbf{f}^t - \mathbf{y}\|_2, \quad (2.29)$$

$$\|\nabla_{W_o} \ell(\theta^t)\|_2 \leq \rho \bar{\lambda}_v \|\mathbf{f}^t - \mathbf{y}\|_2. \quad (2.30)$$

*Dimostrazione.* Per semplicità nella notazione omettiamo l'indice del passo temporale  $t$ . Ricordiamo che la funzione di perdita è la *squared loss function*

$$\ell(\theta^t) = \frac{1}{2} \|\mathbf{f}(\theta^t) - \mathbf{y}\|_2^2$$

e che

$$\mathbf{f}(\theta^t) = \left\{ f(x_n) := W_o \left( \varphi \left( \tau_1 \text{ReLU}(A_n) \right) \right)^T \right\}_{n=1 \dots N}.$$

Per primo consideriamo il gradiente rispetto a  $W_v$ ; per l'equazione (2.24)

$$\begin{aligned}\|\nabla_{W_v} \ell(\theta)\|_{\text{F}} &= \tau_1 \left\| \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T \left( (W_o^t)^T \circ (\text{ReLU}) \left( \beta_{i,n} x_n^T W_v \right)^T \right) \beta_{i,n} x_n^T \right\|_{\text{F}} \\ &\stackrel{(a)}{\leq} \tau_1 \sum_{n=1}^N |(f(x_n) - y_n)| T^{3/2} \bar{\lambda}_o C_x \\ &\leq \tau_1 \sqrt{N \sum_{n=1}^N |(f(x_n) - y_n)|^2} T^{3/2} \bar{\lambda}_o C_x \\ &= \tau_1 T^{3/2} \bar{\lambda}_o C_x \sqrt{N} \|\mathbf{f} - \mathbf{y}\|_2 \\ &= \rho \bar{\lambda}_o \|\mathbf{f} - \mathbf{y}\|_2,\end{aligned}$$

dove (a) segue dal fatto che la derivata della *ReLU* è maggiorata da 1 e dai bound sulle norme di  $\mathbf{X}$  e  $W_o$  dalle ipotesi, mentre l'ultima equazione segue dalla definizione di  $\rho$ .

Considerando il gradiente rispetto a  $W_o$ , per l'equazione (2.23)

$$\begin{aligned}
\|\nabla_{W_o}\ell(\theta)\|_2 &= \tau_1 \left\| \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T \text{ReLU}(\beta_{i,n} x_n^T W_v)^T \right\|_2 \\
&\stackrel{(a)}{=} \tau_1 \left\| \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T (\beta_{i,n} x_n^T W_v)^T \right\|_2 \\
&\stackrel{(b)}{\leq} \tau_1 \sum_{n=1}^N |f(x_n) - y_n| T^{3/2} \bar{\lambda}_v C_x \\
&\leq \tau_1 \sqrt{N \sum_{n=1}^N |f(x_n) - y_n|^2 T^{3/2} \bar{\lambda}_v C_x} \\
&= \rho \bar{\lambda}_v \|\mathbf{f} - \mathbf{y}\|_2,
\end{aligned} \tag{2.31}$$

dove (a) segue come al solito dalla Lipshitzianità della  $\text{ReLU}$  e (b) per le ipotesi sui dati. Per quanto riguarda il gradiente rispetto a  $W_q$ , applicando in ordine il lemma 2.2.2, il fatto che la derivata della  $\text{ReLU}$  è maggiorata da 1 e i bound sui dati all'equazione (2.25) si ottiene

$$\begin{aligned}
\|\nabla_{W_q}\ell(\theta)\|_{\mathbb{F}} &= \tau_0 \tau_1 \left\| \sum_{n=1}^N (f(x_n) - y_n) \sum_{i=1}^T (W_k)^T x_n (\text{diag}(\beta_{i,n}) - \beta_{i,n} \beta_{i,n}^T) \right. \\
&\quad \left. (x_n)^T W_v \left( (W_o^t)^T \circ \text{ReLU}(\beta_{i,n} x_n^T W_v)^T \right) x_n^{(:,i)} \right\|_{\mathbb{F}} \\
&\leq 2\tau_0 \tau_1 \sum_{n=1}^N |f(x_n) - y_n| T \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o (C_x \sqrt{T})^3 \\
&\leq 2\tau_0 \tau_1 \sqrt{N \sum_{n=1}^N |f(x_n) - y_n|^2 T \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o (C_x \sqrt{T})^3} \\
&= 2\rho \tau_0 \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o T C_x^2 \|\mathbf{f} - \mathbf{y}\|_2.
\end{aligned} \tag{2.32}$$

In modo simile si calcola il bound per il gradiente rispetto a  $W_k$  applicando lo stesso ragionamento a (2.26)

$$\|\nabla_{W_k}\ell(\theta)\|_{\mathbb{F}} \leq 2\rho \tau_0 \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o T C_x^2 \|\mathbf{f} - \mathbf{y}\|_2. \tag{2.33}$$

□

**Lemma 2.2.8.** Se per un certo passo temporale  $t$ , valgono

$$\begin{aligned}
\|W_q^t\|_2 &\leq \bar{\lambda}_q, \\
\|W_k^t\|_2 &\leq \bar{\lambda}_k, \\
\|W_v^t\|_2 &\leq \bar{\lambda}_v, \\
\|W_o^t\|_2 &\leq \bar{\lambda}_o,
\end{aligned}$$

allora

$$\|\nabla_{\theta} \mathbf{f}^t\|_2 \leq c_2,$$

con

$$c_2 := \rho \sqrt{\bar{\lambda}_o^2 + \bar{\lambda}_v^2 + (2\tau_0 \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o T C_x^2)^2 + (2\tau_0 \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o T C_x^2)^2}. \quad (2.34)$$

*Dimostrazione.* Come prima, per semplicità omettiamo l'indice  $t$ . Per il lemma 2.2.4

$$\begin{aligned} & \|\nabla_{\theta} \mathbf{f}\|_2 \leq \|\nabla_{\theta} \mathbf{f}\|_{\mathbb{F}} \\ = & \sqrt{\sum_{n=1}^N \left( \|\nabla_{W_o} f(x_n; \theta)\|_2^2 + \|\nabla_{W_q} f(x_n; \theta)\|_{\mathbb{F}}^2 + \|\nabla_{W_k} f(x_n; \theta)\|_{\mathbb{F}}^2 + \|\nabla_{W_v} f(x_n; \theta)\|_{\mathbb{F}}^2 \right)}. \end{aligned} \quad (2.35)$$

Ciascun termine può essere limitato superiormente come fatto nella dimostrazione precedente

$$\begin{aligned} \sum_{n=1}^N (\|\nabla_{W_v} f(x_n; \theta)\|_{\mathbb{F}}^2) & \leq (\rho \bar{\lambda}_o)^2, \\ \sum_{n=1}^N (\|\nabla_{W_o} f(x_n; \theta)\|_2^2) & \leq (\rho \bar{\lambda}_v)^2, \\ \sum_{n=1}^N (\|\nabla_{W_q} f(x_n; \theta)\|_{\mathbb{F}}^2) & \leq (2\rho\tau_0 \bar{\lambda}_k \bar{\lambda}_v \bar{\lambda}_o T C_x^2)^2, \\ \sum_{n=1}^N (\|\nabla_{W_k} f(x_n; \theta)\|_{\mathbb{F}}^2) & \leq (2\rho\tau_0 \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o T C_x^2)^2. \end{aligned}$$

Sommando i bound appena ottenuti si conclude la dimostrazione. □

**Lemma 2.2.9.** Supponiamo che

$$\begin{aligned} \max(\|W_q^t\|_2, \|W_q^{t'}\|_2) & \leq \bar{\lambda}_q, \\ \max(\|W_k^t\|_2, \|W_k^{t'}\|_2) & \leq \bar{\lambda}_k, \\ \max(\|W_v^t\|_2, \|W_v^{t'}\|_2) & \leq \bar{\lambda}_v, \\ \max(\|W_o^t\|_2, \|W_o^{t'}\|_2) & \leq \bar{\lambda}_o, \end{aligned}$$

e definiamo

$$z := 2\tau_0 C_x^2 T (\bar{\lambda}_q + \bar{\lambda}_k).$$

Allora

$$\left\| \nabla_{\theta} \mathbf{f}^{t'} - \nabla_{\theta} \mathbf{f}^t \right\|_2 \leq c_3 \left\| \theta^{t'} - \theta^t \right\|_2, \quad (2.36)$$

con

$$\begin{aligned}
(c_3)^2 &:= N(\tau_1 C_x T^{3/2}(1 + \bar{\lambda}_v z))^2 \\
&+ N \left\{ \tau_1 C_x T^{3/2} \left[ \bar{\lambda}_o z + \bar{\lambda}_o C_x \sqrt{T}(1 + \bar{\lambda}_v z) + 1 \right] \right\}^2 \\
&+ N \left\{ \tau_0 \tau_1 C_x T \left\{ 2\bar{\lambda}_k T C_x^2 \left[ \bar{\lambda}_v \left( \bar{\lambda}_o C_x \sqrt{T}(1 + \bar{\lambda}_v z) + 1 \right) + \bar{\lambda}_o \right] + \bar{\lambda}_v \bar{\lambda}_o [C_x^2 T + 3\bar{\lambda}_k z] \right\} \right\}^2 \\
&+ N \left\{ \tau_0 \tau_1 C_x T \left\{ 2\bar{\lambda}_q T C_x^2 \left[ \bar{\lambda}_v \left( \bar{\lambda}_o C_x \sqrt{T}(1 + \bar{\lambda}_v z) + 1 \right) + \bar{\lambda}_o \right] + \bar{\lambda}_v \bar{\lambda}_o [C_x^2 T + 3\bar{\lambda}_k z] \right\} \right\}^2.
\end{aligned} \tag{2.37}$$

*Dimostrazione.* Osserviamo che

$$\begin{aligned}
&\left\| \nabla_{\theta} \mathbf{f}^{t'} - \nabla_{\theta} \mathbf{f}^t \right\|_2^2 \\
&\leq \sum_{n=1}^N \left( \underbrace{\left\| \nabla_{W_o} f(x_n; \theta^{t'}) - \nabla_{W_o} f(x_n; \theta^t) \right\|_2^2}_{(1)} + \underbrace{\left\| \nabla_{W_v} f(x_n; \theta^{t'}) \nabla_{W_v} f(x_n; \theta^t) \right\|_{\mathbb{F}}^2}_{(2)} \right. \\
&\quad \left. + \underbrace{\left\| \nabla_{W_q} f(x_n; \theta^{t'}) - \nabla_{W_q} f(x_n; \theta^t) \right\|_{\mathbb{F}}^2}_{(3)} + \underbrace{\left\| \nabla_{W_k} f(x_n; \theta^{t'}) \nabla_{W_k} f(x_n; \theta^t) \right\|_{\mathbb{F}}^2}_{(4)} \right),
\end{aligned} \tag{2.38}$$

e consideriamo ciascun termine separatamente.

Per definizione,  $f(x_n) = W_o f_{pre}^n$ , da cui

$$\begin{aligned}
(1) \quad &\left\| \nabla_{W_o} f(x_n; \theta^{t'}) - \nabla_{W_o} f(x_n; \theta^t) \right\|_2 = \left\| \mathbf{f}_{pre}^{t'} - \mathbf{f}_{pre}^t \right\|_2 \\
&\leq \tau_1 C_x T^{3/2} \left( \left\| W_v^{t'} - W_v^t \right\|_2 + \bar{\lambda}_v 2\tau_0 C_x^2 T \left( \bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2 \right) \right) \\
&\leq \tau_1 C_x T^{3/2} (1 + \bar{\lambda}_v 2\tau_0 C_x^2 T (\bar{\lambda}_q + \bar{\lambda}_k)) \left\| \theta^{t'} - \theta^t \right\|_2 \\
&= \tau_1 C_x T^{3/2} (1 + \bar{\lambda}_v z) \left\| \theta^{t'} - \theta^t \right\|_2,
\end{aligned} \tag{2.39}$$

dove la prima disequazione segue dall'equazione (2.22), la seconda dal fatto che  $\theta$  indica l'insieme di tutti i pesi apprendibili, mentre l'ultima equazione segue per la definizione di  $z$ .

Per il secondo termine, applicando l'equazione (2.24) si ottiene

$$(2) \quad \left\| \nabla_{W_v} f(x_n; \theta^{t'}) - \nabla_{W_v} f(x_n; \theta^t) \right\|_{\mathbb{F}}$$

$$\begin{aligned}
&= \tau_1 \left\| \sum_{i=1}^T \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right) \beta_{i,n}^{t'} x_n^T \right. \\
&\quad \left. - \sum_{i=1}^T \left( (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right) \beta_{i,n}^t x_n^T \right\|_{\mathbb{F}} \\
&\stackrel{(a)}{\leq} \tau_1 C_x \sqrt{T} \sum_{i=1}^T \left\| \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right) \beta_{i,n}^{t'} \right. \\
&\quad \left. - \left( (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right) \beta_{i,n}^t \right\|_{\mathbb{F}} \\
&\stackrel{(b)}{\leq} \tau_1 C_x \sqrt{T} \sum_{i=1}^T \left[ \left\| \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right) \right\|_2 \left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_2 \right. \\
&\quad \left. + \left\| \beta_{i,n}^t \right\|_2 \left\| \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T - (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right\|_2 \right] \\
&\stackrel{(c)}{\leq} \tau_1 C_x \sqrt{T} \sum_{i=1}^T \left[ \bar{\lambda}_o \left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_2 \right. \\
&\quad \left. + \left\| \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T - (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right\|_2 \right]. \tag{2.40}
\end{aligned}$$

dove (a) segue per il bound sulla norma del dato iniziale, (b) per la disuguaglianza triangolare aggiungendo e togliendo la quantità

$$\left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right) \beta_{i,n}^t$$

e usando la disuguaglianza di Cauchy-Schwartz, mentre (c) segue dal lemma 2.2.1 e dal bound sulla matrice dei pesi considerata. Per il punto (i) del lemma (2.2.5), il primo termine dell'equazione 2.40 è limitato da

$$\begin{aligned}
\left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_2 &\leq 2\tau_0 C_x^2 T \left( \bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2 \right) \\
&\leq z \left\| \theta^{t'} - \theta^t \right\|_2. \tag{2.41}
\end{aligned}$$

mentre un bound per il secondo termine si trova applicando la disuguaglianza triangolare aggiungendo e togliendo un termine misto, la disuguaglianza di Cauchy-Schwartz e il metodo usato nell'equazione (2.22) come segue

$$\left\| \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T - (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right\|_2$$

$$\begin{aligned}
&\leq \left\| W_o^{t'} \right\|_2 \left\| \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T - \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right\|_2 \\
&\quad + \left\| \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \circ \left( W_o^{t'} - W_o^t \right) \right\|_2 \\
&\leq \bar{\lambda}_o \left\| \beta_{i,n}^{t'} x_n^T W_v^{t'} - \beta_{i,n}^t x_n^T W_v^t \right\|_2 + \left\| W_o^{t'} - W_o^t \right\|_2 \\
&\leq \bar{\lambda}_o C_x \sqrt{T} \left( \left\| W_v^{t'} - W_v^t \right\|_2 + \bar{\lambda}_v 2\tau_0 C_x^2 T \left( \bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2 \right) \right) \\
&\quad + \left\| W_o^{t'} - W_o^t \right\|_2 \\
&\leq \left[ \bar{\lambda}_o C_x \sqrt{T} (1 + \bar{\lambda}_v z) + 1 \right] \left\| \theta^{t'} - \theta^T \right\|_2. \tag{2.42}
\end{aligned}$$

Sommando i termini precedenti, otteniamo

$$\begin{aligned}
&\left\| \nabla_{W_v} f(x_n; \theta^{t'}) - \nabla_{W_v} f(x_n; \theta^t) \right\|_{\mathbb{F}} \\
&\leq \tau_1 C_x T^{3/2} \left[ \bar{\lambda}_o z + \bar{\lambda}_o C_x \sqrt{T} (1 + \bar{\lambda}_v z) + 1 \right] \left\| \theta^{t'} - \theta^T \right\|_2. \tag{2.43}
\end{aligned}$$

Per quanto riguarda il terzo termine nell'equazione (2.40), possiamo ridenominare i termini dell'equazione 2.25 come segue

$$\begin{aligned}
U_{i,n}^{t'} &= (W_k^{t'})^T x_n \left( \text{diag}(\beta_{i,n}^{t'}) - \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T \right) x_n^T \in \mathbb{R}^{k \times d}, \\
U_{i,n}^t &= (W_k^t)^T x_n \left( \text{diag}(\beta_{i,n}^t) - \beta_{i,n}^t \beta_{i,n}^t{}^T \right) x_n^T \in \mathbb{R}^{k \times d}, \\
h_{i,n}^{t'} &= W_v^{t'} \left( (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right) \in \mathbb{R}^{d \times T}, \\
h_{i,n}^t &= W_v^t \left( (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right) \in \mathbb{R}^{d \times T}.
\end{aligned}$$

Di conseguenza

$$\begin{aligned}
(3) \quad &\left\| \nabla_{W_q} f(x_n; \theta^{t'}) - \nabla_{W_q} f(x_n; \theta^t) \right\|_{\mathbb{F}} = \tau_0 \tau_1 \left\| \sum_{i=1}^T U_{i,n}^{t'} h_{i,n}^{t'} x_n^{(:,i)} - \sum_{i=1}^T U_{i,n}^t h_{i,n}^t x_n^{(:,i)} \right\|_{\mathbb{F}} \\
&\leq \tau_0 \tau_1 C_x \left\| \sum_{i=1}^T U_{i,n}^{t'} h_{i,n}^{t'} - \sum_{i=1}^T U_{i,n}^t h_{i,n}^t \right\|_2 \\
&\leq \tau_0 \tau_1 C_x \sum_{i=1}^T \left( \underbrace{\left\| U_{i,n}^{t'} \right\|_2}_{(i)} \underbrace{\left\| h_{i,n}^{t'} - h_{i,n}^t \right\|_2}_{(iii)} + \underbrace{\left\| h_{i,n}^t \right\|_2}_{(ii)} \underbrace{\left\| U_{i,n}^{t'} - U_{i,n}^t \right\|_2}_{(iv)} \right). \tag{2.44}
\end{aligned}$$

Di nuovo, limitiamo separatamente ogni termine della precedente:

$$(i) \quad \left\| U_{i,n}^{t'} \right\|_2 \leq \left\| W_k^{t'} \right\|_2 \left\| x_n \right\|_2 \left\| \text{diag}(\beta_{i,n}^{t'}) - \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T \right\|_2 \leq 2\bar{\lambda}_k T C_x^2,$$

$$(ii) \quad \|h_{i,n}^t\|_2 \leq \|W_v^t\|_2 \|W_o^t\|_2 \leq \bar{\lambda}_v \bar{\lambda}_o,$$

per le disuguaglianze dell'equazione (2.42),

$$(iii) \quad \begin{aligned} \|h_{i,n}^{t'} - h_{i,n}^t\|_2 &\leq \|W_v^{t'}\|_2 \left\| (W_o^{t'})^T \circ \text{ReLU} \left( \beta_{i,n}^{t'} x_n^T W_v^{t'} \right)^T \right. \\ &\quad \left. - (W_o^t)^T \circ \text{ReLU} \left( \beta_{i,n}^t x_n^T W_v^t \right)^T \right\|_2 + \|W_v^{t'} - W_v^t\|_2 \|W_o\|_2 \\ &\leq \left[ \bar{\lambda}_v \left( \bar{\lambda}_o C_x \sqrt{T} (1 + \bar{\lambda}_v z) + 1 \right) + \bar{\lambda}_o \right] \|\theta^{t'} - \theta^t\|_2, \end{aligned}$$

$$(iv) \quad \begin{aligned} &\|U_{i,n}^{t'} - U_{i,n}^t\|_2 \\ &\leq \left\| W_k^{t'} x_n^T \left( \text{diag}(\beta_{i,n}^{t'}) - \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T \right) - W_k^t x_n^T \left( \text{diag}(\beta_{i,n}^t) - \beta_{i,n}^t \beta_{i,n}^t{}^T \right) \right\|_2 \|x_n\|_2 \\ &\leq C_x \sqrt{T} \left[ \|W_k^{t'} - W_k^t\|_2 \|x_n\|_2 \left\| \text{diag}(\beta_{i,n}^{t'}) - \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T \right\|_2 \right. \\ &\quad \left. + \|W_k^t\|_2 \|x_n\|_2 \left\| \text{diag}(\beta_{i,n}^{t'}) - \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T - \text{diag}(\beta_{i,n}^t) - \beta_{i,n}^t \beta_{i,n}^t{}^T \right\|_2 \right] \\ &\leq C_x^2 T \left[ \|\theta^{t'} - \theta^t\|_2 + \bar{\lambda}_k \left( \left\| \text{diag}(\beta_{i,n}^{t'}) - \text{diag}(\beta_{i,n}^t) \right\|_2 + \left\| \beta_{i,n}^{t'} \beta_{i,n}^{t'}{}^T - \beta_{i,n}^t \beta_{i,n}^t{}^T \right\|_2 \right) \right] \\ &\leq C_x^2 T \left[ \|\theta^{t'} - \theta^t\|_2 + \bar{\lambda}_k \left( \left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_\infty + \left( \left\| \beta_{i,n}^{t'} \right\|_2 + \left\| \beta_{i,n}^t \right\|_2 \right) \left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_2 \right) \right] \\ &\leq C_x^2 T \left[ \|\theta^{t'} - \theta^t\|_2 + 3\bar{\lambda}_k \left\| \beta_{i,n}^{t'} - \beta_{i,n}^t \right\|_2 \right] \\ &\leq C_x^2 T \left[ \|\theta^{t'} - \theta^t\|_2 + 3\bar{\lambda}_k 2\tau_0 C_x^2 T \left( \bar{\lambda}_q \left\| W_k^{t'} - W_k^t \right\|_2 + \bar{\lambda}_k \left\| W_q^{t'} - W_q^t \right\|_2 \right) \right] \\ &\leq [C_x^2 T + 3\bar{\lambda}_k z] \|\theta^{t'} - \theta^t\|_2, \end{aligned} \tag{2.45}$$

dove la penultima disequazione segue dal punto (i) del lemma 2.2.5. Sommando i termini trovati, otteniamo

$$\begin{aligned} \left\| \nabla_{W_q} f(x_n; \theta^{t'}) - \nabla_{W_q} f(x_n; \theta^t) \right\|_F &\leq \tau_0 \tau_1 C_x T \left\{ 2\bar{\lambda}_k T C_x^2 \right. \\ &\quad \left. \left[ \bar{\lambda}_v (\bar{\lambda}_o C_x \sqrt{T} (1 + \bar{\lambda}_v z) + 1) + \bar{\lambda}_o \right] + \bar{\lambda}_v \bar{\lambda}_o [C_x^2 T + 3\bar{\lambda}_k z] \right\} \|\theta^{t'} - \theta^t\|_2. \end{aligned} \tag{2.46}$$

In modo analogo, applicando lo stesso ragionamento all'equazione (2.26), il quarto termine dell'equazione (2.40) è limitato da

$$\begin{aligned} \left\| \nabla_{W_k} f(x_n; \theta^{t'}) - \nabla_{W_k} f(x_n; \theta^t) \right\|_F &\leq \tau_0 \tau_1 C_x T \left\{ 2\bar{\lambda}_q T C_x^2 \right. \\ &\quad \left. \left[ \bar{\lambda}_v (\bar{\lambda}_o C_x \sqrt{T} (1 + \bar{\lambda}_v z) + 1) + \bar{\lambda}_o \right] + \bar{\lambda}_v \bar{\lambda}_o [C_x^2 T + 3\bar{\lambda}_k z] \right\} \|\theta^{t'} - \theta^t\|_2, \end{aligned} \tag{2.47}$$

e sommando tutti i bound trovati si conclude la dimostrazione.  $\square$

**Lemma 2.2.10** ([15]). Sia  $g : \mathbb{R}^m \rightarrow \mathbb{R}$  una funzione di classe  $C^2$ . Dati  $x, y \in \mathbb{R}^m$ , supponiamo che  $\|\nabla g(z) - \nabla g(x)\|_2 \leq C \|z - x\|_2$  per ogni  $z = x + t(y - x)$  con  $t \in [0, 1]$ , con  $C > 0$ . Allora

$$g(y) \leq g(x) + \langle \nabla g(x), y - x \rangle + \frac{C}{2} \|x - y\|_2^2.$$

*Dimostrazione.* Definiamo

$$h(t) := g(x + t(y - x)) \quad t \in [0, 1];$$

allora

$$\begin{aligned} g(y) - g(x) &= h(1) - h(0) \\ &= \int_0^1 h'(t) dt \\ &= \int_0^1 \langle \nabla g(x + t(y - x)), y - x \rangle dt \\ &= \langle \nabla g(x), y - x \rangle + \int_0^1 \langle \nabla g(x + t(y - x)) - \nabla g(x), y - x \rangle dt \\ &\leq \langle \nabla g(x), y - x \rangle + \int_0^1 Ct \|y - x\|_2^2 \\ &= \langle \nabla g(x), y - x \rangle + \frac{C}{2} \|y - x\|_2^2 \end{aligned}$$

□

**Dimostrazione del Teorema di Convergenza 2.1.2.** Possiamo riscrivere l'equazione (2.5) come

$$f(\mathbf{X}) = \tau_1 W_o \sum_{i=1}^T \text{ReLU}(\beta_i \mathbf{X}^T W_v)^T = W_o \mathbf{f}_{\text{pre}} \in \mathbb{R}^N,$$

con  $\beta_i = \text{softmax}(\tau_0 (\mathbf{X}^{(:,i)})^T W_q W_k^T \mathbf{X}) \in \mathbb{R}^{(1 \times T) \times N}$ .

Mostriamo per induzione che per ogni  $t \geq 0$

$$(i) \begin{cases} \|W_q^s\|_2 \leq \bar{\lambda}_q, & s \in [0, t], \\ \|W_k^s\|_2 \leq \bar{\lambda}_k, & s \in [0, t], \\ \|W_v^s\|_2 \leq \bar{\lambda}_k, & s \in [0, t], \\ \|W_o^s\|_2 \leq \bar{\lambda}_o, & s \in [0, t], \end{cases} \quad (2.48)$$

$$(ii) \sigma_{\min}(\mathbf{F}_{\text{pre}}^s) \geq \frac{1}{2} \alpha, \quad s \in [0, t], \quad (2.49)$$

$$(iii) \ell(\theta^s) \leq \left(1 - \gamma \frac{\alpha^2}{2}\right)^s \ell(\theta^0), \quad s \in [0, t]. \quad (2.50)$$

I tre punti sono veri per ipotesi al passo  $t = 0$ . Assumiamo dunque che siano valide per il generico passo  $t$  e le verifichiamo per il passo successivo.

(i) Per la disuguaglianza triangolare e le definizioni delle matrici dei pesi nell'algoritmo del gradiente (2.9), vale

$$\begin{aligned}
\|W_q^{t+1} - W_q^0\|_F &\leq \sum_{s=0}^t \|W_q^{s+1} - W_q^s\|_F \\
&= \gamma \sum_{s=0}^t \|\nabla_{W_q} \ell(\theta^s)\|_F \\
&\stackrel{(a)}{\leq} 2\gamma\rho\tau_0\bar{\lambda}_k\bar{\lambda}_v\bar{\lambda}_oTC_x^2 \sum_{s=0}^t \|\mathbf{f}^s - \mathbf{y}\|_2 \\
&\stackrel{(b)}{=} 2\gamma\rho\tau_0\bar{\lambda}_k\bar{\lambda}_v\bar{\lambda}_oTC_x^2 \sum_{s=0}^t \sqrt{2\ell(\theta^s)} \\
&\stackrel{(c)}{\leq} 2\gamma\rho\tau_0\bar{\lambda}_k\bar{\lambda}_v\bar{\lambda}_oTC_x^2 \sum_{s=0}^t \left(1 - \gamma\frac{\alpha^2}{2}\right)^{s/2} \sqrt{2\ell(\theta^0)},
\end{aligned} \tag{2.51}$$

dove (a) segue dal bound superiore trovato nel lemma 2.2.7, (b) segue dal fatto che

$$\|\mathbf{f}^s - \mathbf{y}\|_2 = \sqrt{\sum_{n=1}^N |f(x_n) - y_n|^2} \stackrel{def.}{=} \sqrt{2\ell(\theta^s)}$$

mentre (c) deriva dall'ipotesi induttiva. Detto

$$u := \sqrt{1 - \gamma\frac{\alpha^2}{2}} \in (0, 1) \tag{2.52}$$

vale che

$$\begin{aligned}
&\rho 2\tau_0\bar{\lambda}_k\bar{\lambda}_v\bar{\lambda}_oTC_x^2 \frac{2}{\alpha^2} (1 - u^2) \frac{1 - u^{t+1}}{1 - u} \sqrt{2\ell(\theta^0)} \\
&\leq \rho 2\tau_0\bar{\lambda}_k\bar{\lambda}_v\bar{\lambda}_oTC_x^2 \frac{4}{\alpha^2} \sqrt{2\ell(\theta^0)} \\
&\leq C_q.
\end{aligned} \tag{2.53}$$

Per la disuguaglianza di Weyl, segue

$$\|W_q^{t+1}\|_2 \leq \|W_q^0\|_2 + C_q = \bar{\lambda}_q. \tag{2.54}$$

In modo analogo si mostra che

$$\begin{aligned}
\|W_o^{t+1} - W_o^0\|_2 &\leq \sum_{s=0}^t \|W_o^{s+1} - W_o^s\|_2 \\
&= \gamma \sum_{s=0}^t \|\nabla_{W_o} \ell(\theta^s)\| \\
&\leq \gamma \rho \bar{\lambda}_v \sum_{s=0}^t \sqrt{2\ell(\theta^s)} \\
&\leq \gamma \rho \bar{\lambda}_v \sum_{s=0}^t \left(1 - \gamma \frac{\alpha^2}{2}\right)^{s/2} \sqrt{2\ell(\theta^0)} \\
&\leq \rho \bar{\lambda}_v \frac{2}{\alpha^2} (1 - u^2) \frac{1 - u^{t+1}}{1 - u} \sqrt{2\ell(\theta^0)} \\
&\leq \rho \bar{\lambda}_v \frac{4}{\alpha^2} \sqrt{2\ell(\theta^0)} \\
&\leq C_o
\end{aligned} \tag{2.55}$$

e per la disuguaglianza di Weyl, segue

$$\|W_o^{t+1}\|_2 \leq \|W_o^0\|_2 + C_o = \bar{\lambda}_o. \tag{2.56}$$

Allo stesso modo, si ottengono i bound superiori per le matrici dei values e delle keys:

$$\|W_k^{t+1} - W_k^0\|_F \leq 2\rho\tau_0 \bar{\lambda}_q \bar{\lambda}_v \bar{\lambda}_o T C_x^2 \frac{4}{\alpha^2} \sqrt{2\ell(\theta^0)} \implies \|W_k^{t+1}\|_2 \leq \bar{\lambda}_k, \tag{2.57}$$

$$\|W_v^{t+1} - W_v^0\|_F \leq \rho \bar{\lambda}_o \frac{4}{\alpha^2} \sqrt{2\ell(\theta^0)} \implies \|W_v^{t+1}\|_2 \leq \bar{\lambda}_v. \tag{2.58}$$

(ii) Si osserva che, per il punto (ii) del lemma 2.2.5

$$\begin{aligned}
\|\mathbf{F}_{\text{pre}}^{t+1} - \mathbf{F}_{\text{pre}}^0\|_F &\leq \rho \left\{ \|W_v^{t+1} - W_v^0\|_2 + \|W_v^0\|_2 2\tau_0 C_x^2 \right. \\
&\quad \left. \left( \|W_q^{t+1}\|_2 \|W_k^{t+1} - W_k^0\|_2 + \|W_k^0\|_2 \|W_q^{t+1} - W_q^0\|_2 \right) \right\} \\
&\leq \sum_{s=0}^t \gamma \rho \left\{ \rho \bar{\lambda}_o \sqrt{2\ell(\theta^s)} + \bar{\lambda}_v 2\tau_0 C_x^2 T [2\rho\tau_0 (\bar{\lambda}_k^2 + \bar{\lambda}_q^2) \bar{\lambda}_v \bar{\lambda}_o C_x^2 T \sqrt{2\ell(\theta^s)}] \right\} \\
&\leq \rho^2 \bar{\lambda}_o [1 + 4\tau_0^2 C_x^4 T^2 \bar{\lambda}_v^2 (\bar{\lambda}_Q^2 + \bar{\lambda}_K^2)] \frac{16}{\alpha^2} \sqrt{2\ell(\theta^0)} \\
&= \rho^2 \frac{z}{\bar{\lambda}_o} \frac{16}{\alpha^2} \sqrt{2\ell(\theta^0)} \\
&\leq \frac{1}{2} \alpha.
\end{aligned}$$

Questo implica, per la diseguazione di Weyl, che

$$\sigma_{\min}(\mathbf{F}_{\text{pre}}^{t+1}) \geq \frac{1}{2} \alpha.$$

(iii) Vogliamo dimostrare la disuguaglianza applicando il lemma precedente 2.2.10, dunque dobbiamo verificare che le ipotesi sono soddisfatte. Definiamo  $\theta^{t+\phi} := \theta^t + \phi(\theta^{t+1} - \theta^t)$ , per  $\phi \in [0, 1]$ . Per la disuguaglianza triangolare e la disuguaglianza di Cauchy-Schwartz,

$$\begin{aligned}
& \|\nabla_{\theta}\ell(\theta^{t+\phi}) - \nabla_{\theta}\ell(\theta^t)\|_2 \\
&= \|\nabla_{\theta}\mathbf{f}^{t+\phi} \cdot (\mathbf{f}^{t+\phi} - \mathbf{y}) - \nabla_{\theta}\mathbf{f}^t \cdot (\mathbf{f}^t - \mathbf{y})\|_2 \\
&= \|\nabla_{\theta}\mathbf{f}^{t+\phi} \cdot (\mathbf{f}^{t+\phi} - \mathbf{y}) - \nabla_{\theta}\mathbf{f}^{t+\phi}(\mathbf{f}^t - \mathbf{y}) + \nabla_{\theta}\mathbf{f}^{t+\phi}(\mathbf{f}^t - \mathbf{y}) - \nabla_{\theta}\mathbf{f}^t \cdot (\mathbf{f}^t - \mathbf{y})\|_2 \\
&= \|\nabla_{\theta}\mathbf{f}^{t+\phi} \cdot (\mathbf{f}^{t+\phi} - \mathbf{f}^t) + (\nabla_{\theta}\mathbf{f}^{t+\phi} - \nabla_{\theta}\mathbf{f}^t) \cdot (\mathbf{f}^t - \mathbf{y})\|_2 \\
&\leq \|\mathbf{f}^{t+\phi} - \mathbf{f}^t\|_2 \|\nabla_{\theta}\mathbf{f}^{t+\phi}\|_2 + \|\nabla_{\theta}\mathbf{f}^{t+\phi} - \nabla_{\theta}\mathbf{f}^t\|_2 \|\mathbf{f}^t - \mathbf{y}\|_2 \\
&\leq \|\mathbf{f}^{t+\phi} - \mathbf{f}^t\|_2 \|\nabla_{\theta}\mathbf{f}^{t+\phi}\|_2 + 2\|\nabla_{\theta}\mathbf{f}^{t+\phi} - \nabla_{\theta}\mathbf{f}^t\|_2 \ell(\theta^0),
\end{aligned} \tag{2.59}$$

dove l'ultima disuguaglianza segue per ipotesi induttiva.

Vogliamo trovare dei limiti per ciascun termine; per quanto riguarda il primo termine osserviamo che:

$$\begin{aligned}
& \|\mathbf{f}^{t+\phi} - \mathbf{f}^t\|_2 \\
&\stackrel{(a)}{\leq} \rho \|W_v^{t+\phi}\|_2 \|W_o^{t+\phi} - W_o^t\|_2 + \|W_o^t\|_2 \|\mathbf{F}_{\text{pre}}^{t+\phi} - \mathbf{F}_{\text{pre}}^t\|_2 \\
&\stackrel{(b)}{\leq} \rho \|W_v^{t+\phi}\|_2 \|W_o^{t+\phi} - W_o^t\|_2 \\
&\quad + \rho \|W_o^t\|_2 \left[ \|W_v^{t+\phi} - W_v^t\|_2 \right. \\
&\quad \left. + \|W_v^t\|_2 2\tau_0 C_x^2 T \left( \|W_q^{t+\phi}\|_2 \|W_k^{t+\phi} - W_k^t\|_2 + \|W_k^t\|_2 \|W_q^{t+\phi} - W_q^t\|_2 \right) \right]
\end{aligned} \tag{2.60}$$

dove (a) e (b) seguono dal punto (iii) e (ii) del lemma 2.2.5. Per quanto riguarda le norme delle matrici dei pesi, si osserva che

$$\begin{aligned}
& \|W_q^{t+\phi} - W_q^0\|_{\text{F}} \leq \|W_q^{t+\phi} - W_q^t\|_{\text{F}} + \sum_{s=0}^{t-1} \|W_q^{s+1} - W_q^s\|_{\text{F}} \\
&= \phi\gamma \|\nabla_{W_q}\ell(\theta^t)\|_{\text{F}} + \gamma \sum_{s=0}^{t-1} \|\nabla_{W_q}\ell(\theta^s)\|_{\text{F}} \\
&\leq \gamma \sum_{s=0}^t \|\nabla_{W_q}\ell(\theta^s)\|_{\text{F}}.
\end{aligned}$$

Seguendo passo per passo lo stesso procedimento come nelle equazioni (2.53) e (2.54), si ottiene

$$\|W_q^{t+\phi}\|_2 \leq \bar{\lambda}_q$$

e applicando lo stesso ragionamento alle altre matrici di pesi, si ottiene:

$$\begin{aligned}\|W_k^{t+\phi}\|_2 &\leq \bar{\lambda}_k, \\ \|W_v^{t+\phi}\|_2 &\leq \bar{\lambda}_v, \\ \|W_o^{t+\phi}\|_2 &\leq \bar{\lambda}_o.\end{aligned}$$

Di conseguenza, riprendendo l'equazione (2.60),

$$\begin{aligned}\|\mathbf{f}^{t+\phi} - \mathbf{f}^t\|_2 &\leq \rho\bar{\lambda}_V \|\theta^{t+\phi} - \theta^t\|_2 + \rho\bar{\lambda}_O(1 + \bar{\lambda}_V 2\tau_0 C_x^2 T(\bar{\lambda}_Q + \bar{\lambda}_K)) \|\theta^{t+\phi} - \theta^t\|_2 \\ &= \rho(\bar{\lambda}_V + \bar{\lambda}_O + \bar{\lambda}_V 2\tau_0 C_x^2 T(\bar{\lambda}_Q + \bar{\lambda}_K)) \|\theta^{t+\phi} - \theta^t\|_2 \\ &:= c_1 \|\theta^{t+\phi} - \theta^t\|_2,\end{aligned}\tag{2.61}$$

Per il secondo termine, dal lemma 2.2.8 segue

$$\|\nabla_{\theta} \mathbf{f}^{t+\phi}\|_2 \leq c_2,$$

mentre per il lemma 2.2.9 il terzo termine è limitato da

$$\|\nabla_{\theta} \mathbf{f}^{t+\phi} - \nabla_{\theta} \mathbf{f}^t\|_2 \leq c_3 \|\theta^{t+\phi} - \theta^t\|_2.$$

Di conseguenza, per l'equazione (2.59) vale

$$\begin{aligned}\|\nabla_{\theta} \ell(\theta^{t+\phi}) - \nabla_{\theta} \ell(\theta^t)\|_2 &\leq c_1 c_2 \|\theta^{t+\phi} - \theta^t\|_2 + 2\ell(\theta^0) c_3 \|\theta^{t+\phi} - \theta^t\|_2 \\ &= C \|\theta^{t+\phi} - \theta^t\|_2\end{aligned}\tag{2.62}$$

con

$$C := c_1 c_2 + 2c_3 \ell(\theta^0).\tag{2.63}$$

Di conseguenza le ipotesi del lemma precedente 2.2.10 sono soddisfatte e dal momento che possiamo scrivere

$$\theta^{t+1} = \theta^t - \gamma \nabla_{\theta} \ell(\theta^t)$$

vale

$$\begin{aligned}\ell(\theta^{t+1}) &\leq \ell(\theta^t) + \langle \nabla_{\theta} \ell(\theta^t), \theta^{t+1} - \theta^t \rangle + \frac{C}{2} \|\theta^{t+1} - \theta^t\|_F^2 \\ &= \ell(\theta^t) - \gamma \|\nabla_{\theta} \ell(\theta^t)\|_F^2 + \frac{C}{2} \gamma^2 \|\nabla_{\theta} \ell(\theta^t)\|_F^2 \\ &\stackrel{(a)}{\leq} \ell(\theta^t) - \frac{1}{2} \gamma \|\nabla_{\theta} \ell(\theta^t)\|_F^2 \\ &\stackrel{(b)}{\leq} \ell(\theta^t) - \frac{1}{2} \gamma \|\nabla_{W_o} \ell(\theta^t)\|_2^2 \\ &= \ell(\theta^t) - \frac{1}{2} \gamma \|(\mathbf{F}_{\text{pre}}^t)^T (\mathbf{f}^t - \mathbf{y})\|_2^2 \\ &\leq \ell(\theta^t) - \frac{1}{2} \gamma (\sigma_{\min}(\mathbf{F}_{\text{pre}}^t))^2 \|\mathbf{f}^t - \mathbf{y}\|_2^2 \\ &\leq (1 - \gamma \frac{\alpha}{2}) \ell(\theta^t)\end{aligned}$$

dove (a) vale se lo step-size  $\gamma$  soddisfa la condizione

$$\gamma \leq \frac{1}{C}$$

mentre (b) segue dal fatto che la matrice dei pesi  $W_o^t$  è compresa nei pesi  $\theta^t$  e l'ultima disuguaglianza segue per ipotesi induttiva. □

**Osservazione 2.2.11** (Estensione del Teorema di Convergenza). E' possibile estendere il teorema di convergenza 2.1.2 al caso del *residual Transformer*, cioè un modello costituito come sopra cui viene aggiunta una residual connection. In particolare, possiamo modificare l'equazione (2.1) aggiungendo una connessione residua con il dato iniziale

## 2.3 Convergenza del Transformer

**Definizione 2.3.1.** Consideriamo un'inizializzazione Gaussiana per i pesi delle matrici apprendibili, cioè

$$W_o^{(i,j)} \sim \mathcal{N}(0, \eta_o)$$

$$W_q^{(i,j)} \sim \mathcal{N}(0, \eta_q)$$

$$W_v^{(i,j)} \sim \mathcal{N}(0, \eta_v)$$

$$W_k^{(i,j)} \sim \mathcal{N}(0, \eta_k)$$

per  $i = 1, \dots, d$ ,  $j = 1, \dots, k$ , dove indichiamo con  $\eta_o, \eta_q, \eta_v$  e  $\eta_k$  le varianze dell'inizializzazione gaussiana delle matrici  $W_o, W_q, W_v$  e  $W_k$  rispettivamente. La seguente tabella contiene i valori delle *condizioni iniziali di He e di LeCun* [22]

Condizioni	$\eta_o$	$\eta_q$	$\eta_v$	$\eta_k$	$\tau_1$
LeCun	$k^{-1}$	$k^{-1}$	$k^{-1}$	$k^{-1}$	1
He	$2k^{-1}$	$2k^{-1}$	$2k^{-1}$	$2k^{-1}$	1

Tabella 2.1: Condizioni iniziali per i parametri  $\eta_o, \eta_q, \eta_v, \eta_k, \tau_1$

**Teorema 2.3.2** (Convergenza sotto le condizioni He/LeCun). Sotto le condizioni iniziali espresse nella tabella 2.1, supponiamo che per il dato iniziale  $\mathbf{X}$  valgono i due requisiti:

1.  $\mathbf{X}$  ha rango massimo per colonne;

2. per ogni coppia di dati di input  $(x_n, x_m)$ ,  $n \neq m$ ,  $n, m \in \{1, \dots, N\}$ ,

$$\mathbb{P}\left(\left|\langle x_n^T x_m, x_m^T x_n \rangle\right| \geq t\right) \leq \exp(t^{-c})$$

per una certa costante  $c > 0$ .

Supponiamo che  $k \geq d$  e che valga una delle due condizioni:

(i)  $\tau_0 = k^{-1/2}$  e  $k = \tilde{\Omega}(N^3)$ ;

(ii)  $\tau_0 = k^{-1}$  e  $k = \tilde{\Omega}(N^2)$ .

dove  $\tilde{\Omega}$  indica il comportamento asintotico a meno di fattori logaritmici. Allora con probabilità almeno

$$1 - 8e^{-k/2} - \delta - \exp\left(-\Omega((N-1)^{-c}T^{-1})\right),$$

per un  $\delta$  opportuno, l'algoritmo di discesa del gradiente per il modello Transformer definito nella sezione precedente converge ad un minimo globale, per uno step-size  $\gamma$  sufficientemente piccolo, come nell'equazione (2.16) del teorema 2.1.2.

**Osservazione 2.3.3.** La prima ipotesi sul dato  $\mathbf{X}$  richiede che ogni colonna  $\mathbf{X}^{(:,i)}$  sia linearmente indipendente per  $i = 1, \dots, T$ . Questa richiesta è ragionevole nell'ambito in cui è nato il Transformer: per compiti legati al linguaggio, anche se certi token potrebbero essere ripetuti, l'aggiunta di un embedding posizionale (il quale incorpora informazioni sulla posizione dei token nella sequenza) può rendere le colonne della matrice non correlate. Lo stesso vale per compiti legati all'ambito della visione artificiale, in cui il dato è costruito a partire da un'immagine suddivisa in patches. Ogni patch viene mappata tramite uno strato lineare in una rappresentazione numerica, formando le colonne della matrice  $\mathbf{X}$ . In questo caso si ipotizza che queste colonne possano essere non correlate, grazie al processo di suddivisione e trasformazione lineare, il che aiuta a mantenere l'indipendenza lineare tra le colonne.

Dunque, anche se i dati originali possono avere elementi ripetuti o correlati, l'idea di base è che le trasformazioni applicate possono rendere ogni colonna del dato finale sufficientemente diversa dalle altre, soddisfacendo l'ipotesi di indipendenza lineare.

**Osservazione 2.3.4.** La seconda ipotesi sul dato implica che dati di input diversi,  $x_n, x_m$ ,  $n \neq m$  abbiano una bassa somiglianza, che viene espressa tramite il prodotto scalare. La costante  $c > 0$  rappresenta il grado di dissimilarità

tra i dati: un valore più grande di  $c$  implica che i campioni sono meno distinti, mentre un valore più piccolo implica una maggiore separabilità.

Supponendo che questa somiglianza sia bassa con alta probabilità, si richiede che i dati nel dataset non siano troppo simili tra loro. Questa condizione è importante dal momento che dati troppo simili potrebbero portare a problemi di overfitting o ridurre l'efficacia del modello.

**Osservazione 2.3.5.** Il teorema implica che un'over parametrizzazione di tipo cubico (quadratico) è sufficiente per ottenere la convergenza del Transformer quando il fattore scalare  $\tau_0 = k^{-1/2}$  ( $\tau_0 = k^{-1}$ ).

**Schema della Dimostrazione del Teorema 2.3.2.** L'idea è quella di fornire un limite inferiore per il parametro  $\alpha$  in modo da verificare che questo soddisfi le condizioni (2.15) del teorema di convergenza 2.1.2. Si trovano dei bound superiori per le quantità  $\bar{\lambda}_q, \bar{\lambda}_k, \bar{\lambda}_v, \bar{\lambda}_o$  con le disuguaglianze di concentrazione. L'idea chiave per trovare il bound per  $\alpha$  è di utilizzare il Teorema del cerchio di Gershgorin, che fornisce informazioni sugli autovalori di una matrice. I vincoli trovati si inseriscono nelle equazioni (2.15) per trovare un vincolo sulla dimensione  $k$ , da cui si ricava  $k \geq \tilde{\Omega}(N^3)$  nel caso  $\tau_0 = k^{-1/2}$  e  $k \geq \tilde{\Omega}(N^2)$  nel caso  $\tau_0 = k^{-1}$ .

□

## 2.4 Bias-Variance Tradeoff e Double-Descent Curve

In questa sezione vogliamo studiare i problemi legati alla scelta di un giusto numero di parametri per i modelli di Machine e Deep Learning. La quantità di parametri, cioè pesi apprendibili, che un modello comprende è un fattore rilevante per determinare la capacità di un modello di apprendere dai dati di training e generalizzare su nuovi dati.

In questo contesto, a seconda del livello di complessità di un modello, il suo regime di apprendimento si può suddividere in due fasi [8]:

1. **under-fitting** o *sotto-apprendimento*: un modello troppo semplice non riesce a catturare la complessità dei dati di training. In questo regime la rete non riesce a rappresentare in modo adeguato le relazioni nei dati di addestramento apprendibili, da cui segue una scarsa accuratezza sia sui dati di training che sui dati di test.

2. **over-fitting** o *sovra-apprendimento*: un modello troppo complesso potrebbe rilevare in modo fin troppo specifico le relazioni sui dati di addestramento. La rete risulta troppo specifica sui dati di training e perde la capacità di generalizzazione, dal momento che cattura anche le fluttuazioni casuali del training, per cui l'accuratezza dati di test risulta scarsa.

Si introducono i concetti di *bias* e *variance* per cercare di trovare un compromesso tra i due possibili comportamenti:

- il *bias* si riferisce alla capacità di un modello di rappresentare la relazione esistente tra dati di input e output.
- la *variance* si riferisce alla sensibilità del modello ai cambiamenti nei dati di addestramento.

I modelli con un alto bias e bassa varianza sono tipicamente troppo semplici e non catturano tutte le variazioni nei dati, portando al comportamento di under-fitting. Al contrario, modelli con un basso bias e alta varianza potrebbero adattarsi troppo ai dati di addestramento, catturando anche l'errore: il modello produce risultati molto diversi con dati diversi, il quale comporta over-fitting.

Il bilanciamento tra l'accuratezza del modello e la capacità di generalizzazione si esprime con il concetto di ***bias-variance trade-off*** [8] ed è riassunto nella classica curva U-shape mostrata in figura 2.1

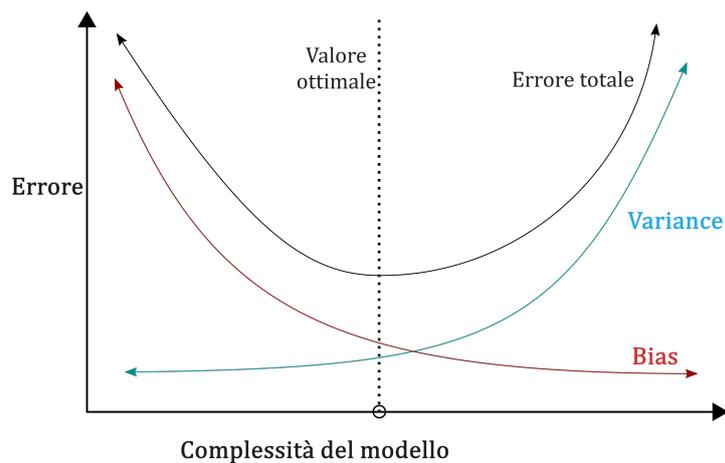


Figura 2.1: Bias-Variance trade-off

Il numero di parametri apprendibili di una rete è uno dei fattori che influisce sul suo livello di complessità. Un modello con un alto numero di parametri

rispetto ai dati forniti ha una maggiore capacità di adattarsi ad essi, infatti una maggior quantità di parametri permette la rappresentazione di pattern complessi, ma aumenta anche il rischio di overfitting.

Per quanto riguarda il modello da noi studiato, i Transformer tendono ad avere un numero estremamente elevato di parametri per costruzione. Inoltre, per il teorema di Convergenza, è necessaria un'over-parametrizzazione almeno di tipo cubica per la convergenza con condizione iniziali di He/LeCun. Nonostante l'elevato numero di parametri, questi modelli hanno una sorprendente capacità di generalizzazione. Nel Transformer, infatti, l'over-parametrizzazione può portare a una riduzione del bias senza necessariamente aumentare la varianza: questo fenomeno potrebbe essere spiegato dal comportamento della **double descent curve**[8]. La double descent curve è un fenomeno osservato in modelli recenti di Deep Learning, in cui la classica curva U-shape viene sostituita da una nuova curva che comprende tre fasi come mostrato in figura 2.2:

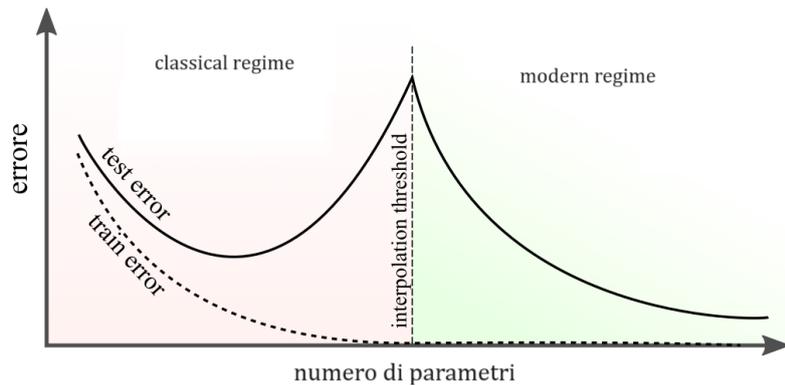


Figura 2.2: Double-descent curve

- (i) Inizialmente si ha una discesa dell'errore di training e test con l'aumentare della complessità del modello, come previsto dal classico bias-variance trade-off.
- (ii) In una seconda fase il modello inizia a sovra-adattarsi ai dati di training, catturandone dettagli specifici e rumore, con un conseguente aumento dell'errore sul test.
- (iii) In una terza fase osservata nei modelli deep, se la complessità aumenta oltre ad un certo punto critico, l'errore sul test set inizia nuovamente a diminuire.

I Transformer, con la loro complessità e profondità, possono effettivamente entrare nella seconda fase della curva, continuando a migliorare le loro prestazioni, nonostante l'enorme quantità di parametri.

Il caso estremo di overfitting accade in prossimità dell' *interpolation threshold* [8], soglia che rappresenta il punto in cui il modello ha esattamente abbastanza parametri per interpolare perfettamente i dati di training, cioè ottenere un errore sul training pari a zero. Al di sotto di questa soglia, il modello potrebbe non essere sufficientemente complesso per apprendere i dati. Lavori recenti suggeriscono che i moderni modelli di deep learning, come il Transformer, operano in un regime altamente sovra-parametrizzato, ben oltre la soglia di interpolazione, e tuttavia mantengono la capacità di generalizzare, anche senza l'utilizzo di tecniche di regolarizzazione [8].

Per generiche deep neural networks, è possibile trovare un bound superiore per l'*interpolation threshold* in funzione del numero di campioni del dataset  $N$  [8]. Detto  $P$  il numero di parametri del modello, possiamo studiare la soglia di interpolazione dividendo lo spazio delle fasi  $(N, P)$  in due regioni: un regime di over-parametrizzazione e un regime di sotto-parametrizzazione. Lo scopo è quello di cercare di dare una caratterizzazione della linea critica  $P^*(N)$  che delimita queste due regioni e che dunque rappresenta la soglia di *interpolation threshold*. Si dimostra che per una generica deep neural network vale un upper bound della forma  $P^* \propto N$ , sopra la quale si mostra che non esistono minimi locali stabili, e che non dipende dalla profondità della rete stessa ma dalla sola struttura dei dati.

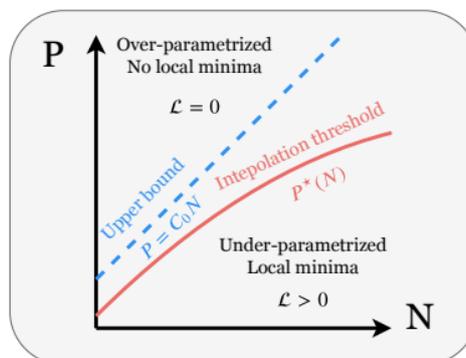


Figura 2.3: Soglia di Interpolazione  $P^*(N)$  [8]:  $N$  numero di esempi,  $P$  numero di parametri,  $\mathcal{L}$  loss function

**Teorema 2.4.1.** Sia  $\{(x_i, y_i)\}_{i=1}^N$  il training set di un problema di classificazione binaria, in cui  $y_i \in \{-1, 1\}$  per  $i = 1, \dots, N$ . Consideriamo una rete neurale fully-connected il cui output è rappresentata da una funzione parametrizzata  $f_W$  e la cui funzione di perdita è la *Hinge-Loss quadratica*. Allora

$$P^* \leq \frac{N}{C_0}$$

dove  $P^*$  è la soglia di interpolation threshold e  $C_0$  è una costante che dipende dal dataset.

*Dimostrazione.* La *Hinge-Loss quadratica* si calcola come

$$\ell(W) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (\max(0, \Delta_i))^2 \quad (2.64)$$

dove

$$\Delta_i := 1 - y_i f_W(x_i).$$

Si osserva facilmente che, essendo  $y_i = \pm 1$ ,

$$\Delta_i = \begin{cases} 0 & \text{se } y_i = f_W(x_i) \\ 1 & \text{altrimenti} \end{cases} \quad (2.65)$$

da cui

$$\ell(W) = \frac{1}{N} \sum_{i \in M} \frac{1}{2} \Delta_i^2$$

dove  $M$  è l'insieme

$$M := \{i \in \{1, \dots, N\} : y_i \neq f_W(x_i)\}$$

e supponiamo abbia cardinalità  $n_\Delta$ .

La soglia  $P^*$  che vogliamo caratterizzare rappresenta il punto di transizione da un regime di sotto-parametrizzazione in cui la rete non riesce a soddisfare tutti i vincoli, e dunque  $\ell > 0$ , ad un regime di sovra-parametrizzazione in cui si raggiunge  $\ell = 0$ . Supponiamo dunque che, con un'inizializzazione dei parametri opportuna, sia possibile che, aumentando il numero di parametri  $P$ , il metodo di discesa del gradiente possa portare a ottenere  $\ell = 0$  per un certo  $P^*$ , cioè

$$\lim_{P \rightarrow P^*} \ell = 0. \quad (2.66)$$

Chiedere che  $\ell \rightarrow 0$  equivale a chiedere che  $\Delta_i \rightarrow 0$  per ogni esempio del training set. Per ogni  $i = 1, \dots, N$ , il vincolo  $\Delta_i \approx 0$  definisce una varietà di

dimensione  $P-1$ ; quindi chiedere che vengano soddisfatte  $n_\Delta$  equazioni implica la definizione di una varietà di dimensione  $P - n_\Delta$ . Imponendo l'esistenza di una soluzione per (2.66), segue il bound

$$P^* \geq n_\Delta. \quad (2.67)$$

Cerchiamo di ottenere un un limite opposto, imponendo che la matrice Hessiana, cioè la matrice delle derivate seconde della loss, sia definita positiva in un minimo stabile. Per fare ciò dobbiamo considerare una funzione di attivazione che sia liscia. Definiamo

$$\begin{aligned} \mathcal{H} &:= Hess(\ell) = \frac{1}{N} \sum_{i \in M} \nabla \Delta_i \otimes \nabla \Delta_i + \frac{1}{N} \sum_{i \in M} \Delta_i \nabla \otimes \nabla \Delta_i \\ &=: \mathcal{H}_0 + \mathcal{H}_p \end{aligned} \quad (2.68)$$

dove  $\otimes$  rappresenta il prodotto tensore. Il primo termine  $\mathcal{H}_0$  è semi-definito positivo ed è la somma di  $n_\Delta$  matrici di rango 1, dunque  $rnk(\mathcal{H}_0) \leq n_\Delta$ , per cui il suo kernel ha dimensione almeno  $P - n_\Delta$ .

Sia  $E^-$  l'autospazio negativo di  $\mathcal{H}_p$ , cioè lo spazio costituito dagli autovettori associati agli autovalori negativi di  $\mathcal{H}_p$ , e indichiamo con  $P^-$  la sua dimensione. La stabilità del minimo impone che

$$ker(\mathcal{H}_0) \cup E^- = \{0\}$$

che è possibile solo se  $n_\Delta \geq P^-$ . Di conseguenza, è possibile ottenere un minimo con training loss positiva se

$$N \geq n_\Delta \geq P^-. \quad (2.69)$$

Nel caso in cui la funzione di attivazione è la *ReLU*, si mostra che lo spettro di  $\mathcal{H}_p$  è simmetrico con

$$\lim_{P \rightarrow \infty} \frac{P^-}{P^+} = 1$$

indipendentemente dalla profondità della rete, dove  $P^+$  è il numero di autovettori positivi. In generale è possibile dimostrare che lo spettro limite di  $\mathcal{H}_p$  contiene una frazione finita  $C_0 = P^-/P$  di autovalori negativi per  $P, N \rightarrow \infty$  (ad esempio,  $C_0 = 1/2$  per la *ReLU*, mentre  $C_0 \approx 0.43$  per la tangente iperbolica). Infine, assumendo che lo spettro di  $\mathcal{H}_p$  non abbia una densità finita di autovalori nulli, otteniamo dall'equazione (2.69) che non è possibile incontrare minimi locali con training loss positiva se  $P > N/C_0$ ; in particolare, segue che

$$P^* \leq \frac{N}{C_0}.$$

□

**Osservazione 2.4.2.** Lo stesso risultato si ottiene nel caso di funzioni di attivazione non lisce, come ad esempio la *ReLU*. In questo caso, è possibile che un minimo si trovi in un punto in cui la derivata seconda non è definita lungo qualche direzione, per cui bisogna modificare l'equazione (2.69). Detto  $P_c := \beta P$  il numero di direzioni che presentano una cuspidi in prossimità di un punto stabile ( $\beta \in (0, 1)$ ), allora

$$n_{\Delta} > P^- - P_c = P(C_0 - \beta)$$

da cui

$$P^*(N) \leq \frac{N}{C_0 - \beta}.$$

# Capitolo 3

## Applicazione ai brillamenti solari

In questo capitolo vogliamo applicare i metodi teorici studiati nei capitoli precedenti al problema della previsione dei brillamenti solari. Nella prima parte, si vogliono esaminare in dettaglio questi fenomeni fisici, fornendo una classificazione dei brillamenti e discutendo l'importanza di una previsione accurata. Successivamente, si introduce il dataset di riferimento che verrà utilizzato nel capitolo seguente per la classificazione di questi eventi, evidenziando il significativo sbilanciamento presente nei dati analizzati. Infine, nell'ultima sezione, si analizza la metodologia adottata per affrontare il problema sia in un contesto multiclasse che binario, fornendo una spiegazione dettagliata del modello utilizzato e delle sue specifiche tecniche.

### 3.1 Introduzione al problema

I brillamenti solari, o *solar flares*, sono potenti esplosioni che avvengono sulla superficie del Sole. Durante un flare, viene emessa un'enorme quantità di energia sotto forma di radiazione elettromagnetica, che include raggi X, raggi gamma e radiazioni ultraviolette, nonché luce visibile.

Nonostante il fenomeno fisico alla base di tali eventi non sia ancora completamente compreso, l'ipotesi più accreditata è legata alla riconnessione delle linee del campo magnetico. Si pensa che questi fenomeni si verificano quando l'energia immagazzinata nei campi magnetici del Sole viene improvvisamente rilasciata: il Sole, costituito da un plasma di gas carichi, genera intensi campi magnetici attraverso il movimento turbolento del plasma stesso; talvolta,

questi campi si intrecciano, accumulando energia potenziale. Quando i campi magnetici si rompono o si riconfigurano improvvisamente, l'energia accumulata viene rilasciata sotto forma di un flare solare.

Questi eventi hanno origine a partire dalle regioni attive del Sole, anche dette *Active Regions (ARs)*, aree del Sole con un'alta concentrazione di flusso del campo magnetico. Tuttavia, non tutte le regioni attive del Sole danno luogo ad un flare solare.

La previsione di tali eventi è di grande importanza per la nostra società a causa delle possibili ripercussioni che questi possono avere sulla Terra: nonostante la parte più energetica delle radiazioni emesse non oltrepassa l'atmosfera terrestre, queste possono avere un grave impatto sulle tecnologie di uso quotidiano. I brillamenti solari più intensi, infatti, possono provocare disturbi a livello di comunicazioni satellitari e, conseguentemente, per i sistemi di navigazione satellitare (GPS), creare interferenze nelle comunicazioni radio sulla Terra, causare guasti alle reti elettriche e conseguenti blackout oppure costringere le compagnie aeree a modificare le rotte. Oltre ciò, le radiazioni risultano pericolose per i veicoli spaziali e gli astronauti al di fuori della magnetosfera terrestre. Un'altra conseguenza di tali fenomeni sono le aurore boreali e australi. A titolo di esempio, un rapporto del 2008 del National Research Council ha concluso che una supertempesta solare simile all'evento di Carrington del 1859, la più potente tempesta geomagnetica mai registrata nella storia moderna, potrebbe paralizzare l'intera rete elettrica degli Stati Uniti per mesi e causare danni economici tra 1 e 2 trilioni di dollari [7].

Di conseguenza, per cercare di mitigare gli effetti delle attività eruttive solari e l'impatto economico che le sue conseguenze hanno sulla Terra, negli ultimi anni la comunità eliofisica si è concentrata sempre più sulla previsione dei brillamenti solari, analizzando dati attuali e storici sul campo magnetico provenienti da regioni attive dal Sole.

Dal momento che i flares solari sono principalmente causati dal rilascio improvviso di energia magnetica nella corona solare, è ragionevole che una loro previsione si basi sulla scelta di adeguate proprietà e informazioni disponibili sul campo magnetico delle regioni attive. Tali eventi sono classificati in modo logaritmico in base al loro picco di flusso di raggi X come *A, B, C, M* e *X*.

## 3.2 Dataset

Il dataset di riferimento che utilizziamo per lo studio del problema è quello fornito da *Angry et al.*[3] nell'articolo *Multivariate time series dataset for space weather data analytics*. L'obiettivo degli autori è quello di fornire un dataset completo e pulito degli eventi solari su cui fare riferimento per la previsione dei flares, costituito da *serie temporali multivariate (MVTS)*.

Il dataset creato si basa principalmente sui Spaceweather HMI Active Region Patches (SHARPs) disponibili presso il Joint Science Operations Center (JSOC), magnetogrammi vettoriali solari ottenuti dall'Helioseismic and Magnetic Imager (HMI) a bordo del Solar Dynamics Observatory (SDO). Una volta che una regione attiva viene individuata dal sistema automatizzato di monitoraggio di HMI, detto HARP (HMI Active Region Patches), le viene assegnato un identificatore univoco, detto *HARPNUM*, e viene monitorata per tutto il suo tempo di vita: ogni 12 minuti vengono calcolati automaticamente alcuni parametri della regione attiva a partire dai dati HMI del vettore campo magnetico. Si ottengono, dunque, serie temporali di parametri, dette *features*, che caratterizzano le *ARs* e che rappresentano i dati SHARPs. Il numero di osservazioni in una serie HARP dipende da quanto tempo le regioni attive sono state nel campo di vista dello strumento HMI.

Gli autori dell'articolo hanno ricavato un insieme di parametri del campo magnetico dalle singole patch di regione e li hanno trasformati in serie temporali multivariate per l'intera durata di una determinata serie HARP, in modo da poter analizzare l'evoluzione delle *ARs*. Le 24 features del campo magnetico selezionate per la previsione dei flare sono elencate di seguito nell'originale dicitura in inglese:

1. ABSNJZH : Absolute value of the net current helicity in G<sup>2</sup>/m;
2. EPSX\* : Sum of X-component of normalized Lorentz force;
3. EPSY\* : Sum of Y-component of normalized Lorentz force;
4. EPSZ\* : Sum of Z-component of normalized Lorentz force;
5. MEANALP : Mean twist parameter;
6. MEANGAM : Mean inclination angle;
7. MEANGBH : Mean value of the horizontal field gradient;

8. MEANGBT : Mean value of the total field gradient;
9. MEANGBZ : Mean value of the vertical field gradient;
10. MEANJZD : Mean vertical current density;
11. MEANJZH : Mean current helicity;
12. MEANPOT : Mean photospheric excess magnetic energy density;
13. MEANSHR : Mean shear angle;
14. R\_VALUE\* : Total unsigned flux around high gradient polarity inversion lines;
15. SAVNCP : Sum of the absolute value of the net current per polarity;
16. SHRG45 : Area with shear angle greater than 45 degrees;
17. TOTBSQ\* : Total magnitude of Lorentz force;
18. TOTFX\* : Sum of X-component of Lorentz force;
19. TOTFY\* : Sum of Y-component of Lorentz force;
20. TOTFZ\* : Sum of Z-component of Lorentz force;
21. TOTPOT : Total photospheric magnetic energy density;
22. TOTUSJH : Total unsigned current helicity;
23. TOTUSJZ : Total unsigned vertical current;
24. USFLUX : Total unsigned flux in Maxwells.

Gli 8 parametri contrassegnati con \* non sono disponibili direttamente nei dati SHARPs.

Le informazioni sui flare che potrebbero verificarsi nella regione di interesse, tuttavia, non sono presenti negli SHARPs, ma vengono forniti dai Geostationary Operational Environmental Satellites (GOES), una serie di satelliti meteorologici gestiti dalla National Oceanic and Atmospheric Administration (NOAA) degli Stati Uniti, progettati per monitorare e raccogliere dati dal Sole e dotati di rilevatori di raggi X e particelle. Ad ogni serie temporale, viene dunque associata un'etichetta che indica la classe di appartenenza dell'evento.

Come già accennato in precedenza, la classificazione di un brillamento avviene in base alla loro intensità, misurata in raggi X, utilizzando un sistema che li divide in cinque classi principali. Ognuna di queste classi rappresenta un diverso livello di energia rilasciata, e ciascuna classe superiore è 10 volte più potente della precedente:

- i flare di classe *A* risultano i più deboli: la loro intensità in raggi X è inferiore a 10 nanowatt per metro quadrato ( $\text{nW}/\text{m}^2$ );
- i flare di classe *B* hanno un'intensità compresa tra 10 e 100  $\text{nW}/\text{m}^2$ ;
- i flare di classe *C* hanno un'intensità compresa tra 100 e 1.000  $\text{nW}/\text{m}^2$ ;
- i flare di classe *M* hanno un'intensità compresa tra 1.000 e 10.000  $\text{nW}/\text{m}^2$ ; questi possono causare brevi blackout radio nelle regioni polari della Terra e disturbare le comunicazioni radio;
- i flare di classe *X* sono i più potenti, con un'intensità superiore a 10.000  $\text{nW}/\text{m}^2$ . I flare più intensi di questa classe possono avere gravi effetti sulla Terra, possono provocare blackout radio in tutto il mondo e tempeste di radiazioni di lunga durata nell'alta atmosfera, disturbare le reti elettriche, le comunicazioni e i sistemi GPS.

In generale, si è particolarmente interessati a prevedere soprattutto i flare di classe *M* e *X* in quanto tali brillamenti possono innescare espulsioni di massa coronale (*CME*), che, se dirette verso la Terra, possono generare tempeste geomagnetiche intense che possono impattare il nostro pianeta.

I dati messi a disposizione da [3] comprendono 4098 MVTs provenienti da regioni attive verificatesi tra Maggio 2010 e Dicembre 2018. In figura 3.1 si presenta l'intero processo di costruzione del dataset.

**Partizionamento.** Su ciascuna MVTs vengono effettuate più previsioni, determinando una finestra temporale di osservazione delle features e una conseguente finestra di osservazione per la previsioni. Dunque, da ognuna di esse si possono creare più serie temporali, ciascuna ricoprente una finestra temporale di 12 ore. Poichè i parametri di ogni *ARs* vengono registrate ogni 12 minuti, una serie temporale è costituita da 60 tempi. É importante, quindi, che i set utilizzati per l'addestramento del modello, in particolare training e test, facciano affidamento su serie temporali appartenenti a diverse MVTs, in modo che la valutazione del modello sia effettivamente su nuovi campioni e risulti più

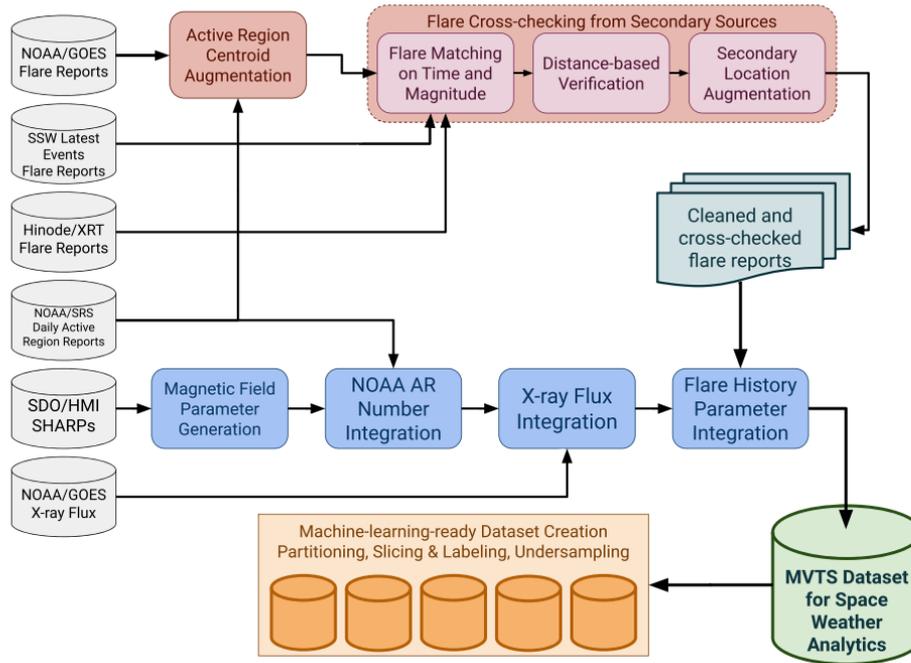


Figura 3.1: Diagramma del processo di generazione del dataset [3]. In rosso la pulizia dei flares, in blu la generazione delle MVTTS, in verde la creazione di set pronti per l'apprendimento automatico.

affidabile. In merito a ciò, gli autori hanno proposto una divisione opportuna del dataset in 5 partizioni di tempo disuguali per la validazione degli algoritmi di Machine Learning proposti, in modo da evitare sovrapposizioni di segmenti temporali in diverse partizioni, per cui training set e test set si basano effettivamente su diverse MVTTS.

Un'altra problematica legata alla previsione dei flares solari è il forte sbilanciamento del dataset: i brillamenti di forte intensità (classe  $M$  o  $X$ ), i quali hanno i maggiori impatti sull'ambiente spaziale e sono quindi i più interessanti nelle analisi predittive, sono anche i più rari. Nel dataset finale, si trovano solo 730 flare di classe  $M$  e 50 di classe  $X$ , che corrispondono a circa il 6,8% del totale. Data la scarsità dei campioni di classe  $M$  e  $X$ , è stata proposta una strategia di validazione più robusta per le applicazioni: la stratificazione segmentata nel tempo. Il dataset è diviso in modo che ciascuna partizione contenga una percentuale ammissibile di flare delle classe minoritarie, in modo da preservare lo sbilanciamento del dataset.

Si riportano le 5 partizioni ottenute, le quali hanno le cinque classi in percentuali abbastanza simili ma un diverso numero di campioni totali.

Partizione:	1	2	3	4	5
Totale	69189	79541	37812	43585	66503
Classe A	56319 81.40%	65364 82.18%	30766 81.37%	36667 84.13%	54908 82.56%
Classe B	5424 7.84%	4681 5.89%	638 1.69%	693 1.59%	5384 8.10%
Classe C	6266 9.06%	8211 10.32%	5131 13.57%	5335 12.24%	5317 8.00%
Classe M	1029 1.49%	1225 1.54%	1164 3.08%	817 1.87%	884 1.33%
Classe X	151 0.22%	60 0.08%	113 0.3%	73 0.17%	10 0.02%

Tabella 3.1: Numero di campioni per classe nelle 5 partizioni

### 3.2.1 Data Preprocessing e Normalizzazione

Poichè i valori dei parametri del campo magnetico sono registrati in scale diverse, viene eseguita una *normalizzazione z-score*, una tecnica utilizzata per trasformare i dati in modo che abbiano una media di zero e una deviazione standard di uno. Utilizzando questo metodo, possiamo valutare e confrontare efficacemente la significatività relativa delle varie caratteristiche all'interno del nostro dataset. Possiamo rappresentare le  $N$  MVTS composte da  $d = 24$  features e  $T$  punti temporali come un tensore 3-dimensionale  $\chi \in \mathbb{R}^{N \times d \times T}$ . La normalizzazione z-score prevede l'elaborazione dei dati attraverso tre step:

1. si effettua un reshape del tensore in modo che le dimensioni dei parametri diventino le colonne della matrice; la nuova matrice  $\chi_1 \in \mathbb{R}^{dT \times N}$  ha colonne  $C_1, \dots, C_N$ ;
2. per ciascuna colonna, si esegue la *z-normalization*

$$\hat{x}_k^j = \frac{x_k^j - \mu^j}{\sigma^j} \quad (3.1)$$

dove  $x_k^j$  è il k-esimo elemento della colonna  $C_j$ ,  $1 \leq k \leq dT$ ,  $\mu^j$  e  $\sigma^j$  sono rispettivamente la media e la varianza della colonna  $C_j$ ;

3. infine si esegue nuovamente un reshape per tornare al tensore 3-dimensionale  $\hat{\chi} \in \mathbb{R}^{N \times d \times T}$ .

### 3.3 Metodologia

Il modello utilizzato è ripreso dall'articolo *End-to-End Attention/Transformer Model for Solar Flare Prediction from Multivariate Time Series Data* [2], in cui si sfrutta il meccanismo dell'attenzione del Transformer. Il modello, mostrato in figura 3.2, è quello definito nel Capitolo 1, composto da un encoder che individua le dipendenze temporali tramite il meccanismo dell'attenzione e produce rappresentazioni sensibili al contesto per ciascun timestamp, seguito da una fully-connected network che viene applicata a ciascuno timestamp in modo indipendente e che presenta non linearità, consentendo al modello di incorporare informazioni aggiuntive. Gli autori dell'articolo hanno validato il

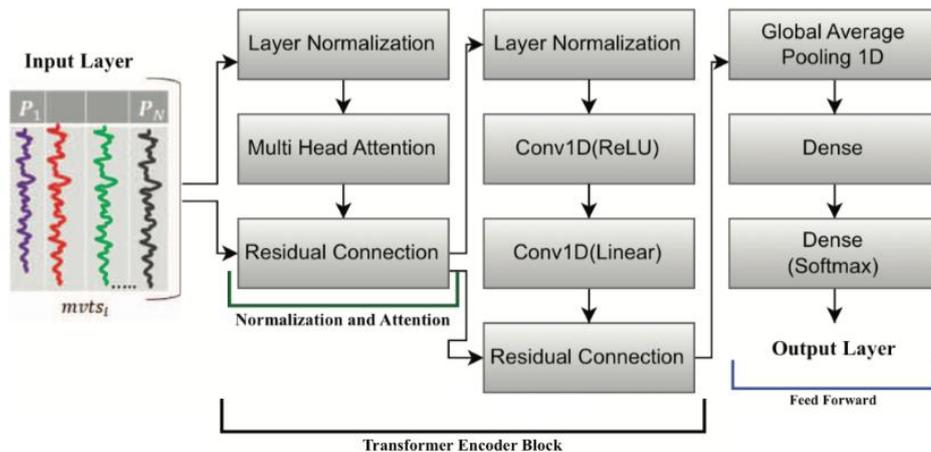


Figura 3.2: Modello per la classificazione [2]

modello su un dataset di serie temporali costituito da 4 classi rappresentate in maniera bilanciata, cioè ciascuna delle classi  $B, C, M$  e  $X$  costituisce il 25% del totale. Il nostro studio, invece, si propone di utilizzare il modello su un dataset sbilanciato, in modo da capire se il Transformer è un modello in grado di cogliere anche le relazioni sulle classi meno rappresentate.

Analizziamo nello specifico i due casi multiclasse e binario, che affrontano il problema con lo stesso metodo ma presentano alcune diversità.

#### 3.3.1 Il caso multiclasse

Di seguito, si propone in pseudo-codice la definizione delle funzioni per la generazione del modello.

---

## Transformer Encoder

---

```
def transformer_encoder(inputs, head_size, num_heads, ff_dim):
    x ← LayerNormalization(inputs, epsilon=1e-6)
    x ← MultiHeadAttention(x, x, key_dim=head_size, num_heads=num_heads)
    res ← x + inputs
    x ← LayerNormalization(inputs, epsilon=1e-6)
    x ← Conv1D(x, filters=ff_dim, kernel_size=1, activation="relu")
    x ← Conv1D(x, filters=inputs.shape[-1], kernel_size=1)
    return res + x
```

---

## Transformer Model

---

```
def build_model(
    input_shape,
    head_size,
    num_heads,
    ff_dim,
    num_transformer_blocks,
    mlp_units,
    n_classes,
):
    inputs ← Input(shape=input_shape)
    x ← inputs
    for i in range(num_transformer_blocks):
        x ← transformer_encoder(inputs, head_size, num_heads, ff_dim)
    x ← GlobalAveragePooling(x, data_format="channel_first")
    for dim in mlp_units:
        x ← Dense(x, dim, activation="relu")
    outputs ← Dense(x, n_classes, dim, activation="softmax")
    return Model(inputs, outputs)
```

---

La prima funzione definisce il solo blocco dell'encoder del Transformer, costituito dalle seguenti componenti:

- Layer di normalizzazione : il tensore *inputs* che rappresenta il dato viene passato normalizzato lungo la dimensione delle features;
- Multi-Head Attention: si implementa il meccanismo dell'attenzione sul tensore normalizzato. Ogni testa in *num\_heads* impara a catturare diverse relazioni all'interno della sequenza. L'output di questo strato mantiene la stessa dimensione dell'input;
- Residual Connection: l'output precedente viene sommato elemento per elemento all'input originale *inputs*;
- Layer di normalizzazione: il risultato della connessione residua viene nuovamente normalizzato;
- 1-D Convolutional Layer: si utilizza una convoluzione con *ff\_dim* kernels di dimensione 1 e con funzione di attivazione *ReLU*. Questo layer agisce come una parte di feed-forward network, applicando una trasformazione non lineare tramite la funzione di attivazione indipendente dalla posizione nella sequenza;
- 1-D Convolutional Layer: un altro layer convoluzionale viene applicato con  $inputs.shape[-1]=\#features$  filtri di dimensione 1 per ottenere la dimensione del primo input;
- Residual Connection: l'output della funzione è dato dalla somma dell'ultimo risultato con la penultima connessione residua.

La seconda funzione definisce il modello completo usato per l'addestramento e opera come segue:

- definisce il primo Input layer e lo imposta come livello corrente;
- applica multipli blocchi del *transformer\_encoder*, tante volte quanto specificato in *num\_transformer\_blocks*;
- applica un Global Average Pooling;
- infine si implementa un perceptrone multistrato composto da *mlp\_units.shape* livelli nascosti e, per ogni strato *i*, il numero di strati è specificato in *mlp\_units[i]*. Poichè stiamo considerando il caso multiclasse, l'ultimo strato calcola l'output tramite l'utilizzo della *softmax*, generando un output di dimensione uguale al numero delle classi presenti.

I valori utilizzati per la definizione delle funzioni sono i seguenti:

- *input\_shape* = (60, 24), taglia che caratterizza l'input, ovvero ( $T, d$ ) dove  $T$  rappresenta il numero dei tempi e  $d$  il numero di features;
- *head\_size* = 256, che rappresenta la dimensione dello spazio in cui sono codificate le sequenze;
- *num\_heads* = 4, il numero di teste dell'attenzione per il Multi-Head Attention layer;
- *ff\_dim* = 4, la dimensione del kernel del primo layer convoluzionale;
- *num\_transformer\_blocks* = 10, il numero di volte per cui si applica l'encoder;
- *mlp\_units* = [128], cioè si implementa un perceptrone con un solo layer nascosto composto da 128 neuroni;
- *n\_classes* = 3 per quanto vedremo nel capitolo successivo.

### 3.3.2 Il caso binario

Nel caso binario, le due classi che vogliamo predire riguardano l'avvenimento o meno di un flare solare sopra la classe M, (quindi di classe M o X) nelle successive 24 ore a partire da una serie temporale. In questo caso il blocco dell'Encoder resta invariato, mentre si apportano delle modifiche alla seconda funzione che costruisce il modello. In particolare si modifica l'ultima riga del codice precedente, infatti, avendo un problema binario, la funzione di attivazione utilizzata è la sigmoide (1.29), per cui bisogna cambiare l'output dell'ultimo strato della rete.

---

#### Transformer Model Binary

---

```
def build_model_bin(  
    input_shape,  
    head_size,  
    num_heads,  
    ff_dim,  
    num_transformer_blocks,  
    mlp_units,
```

```

    n_classes,
):
    inputs ← Input(shape=input_shape)
    x ← inputs
    for i in range(num_transformer_blocks):
        x ← transformer_encoder(inputs, head_size, num_heads, ff_dim)
    x ← GlobalAveragePooling(x, data_format="channel_first")
    for dim in mlp_units:
        x ← Dense(x, dim, activation="relu")
    outputs ← Dense(x, n_classes-1, dim, activation="sigmoid")
    return Model(inputs, outputs)

```

---

In questo modo, l'output della rete risulta coerente con quanto si aspetta la funzione di perdita, che sarà la *SOL function*, definita in equazione (1.42), dal momento che il codice verrà applicato ad un set con  $n\_classes=2$ .

### 3.4 Specifiche (o Condizioni) sul modello

In quest'ultima sezione vogliamo discutere sulle scelte fatte per l'esecuzione del modello.

**Early Stopping.** Come accennato nel primo capitolo, una tecnica molto usata nel contesto del Deep Learning per evitare l'overfitting è l'*early stopping*, per cui si sceglie un numero di epoche  $N_p$  dopo il quale si arresta il modello se l'errore sul test di validazione non migliora dopo le  $N_p$  epoche scelte. Nel modello da noi implementato, si è scelto di porre il suddetto numero di epoche a  $N_p = 15$ . Tale numero è stato appropriatamente scelto dopo un elevato numero di esperimenti che monitoravano l'apprendimento del transformer.

**Numero di epoche.** Il modello del Transformer ha, per sua natura, un numero molto elevato di parametri. Di conseguenza non è consigliabile scegliere un numero troppo elevato di epoche, altrimenti l'addestramento del modello richiederebbe troppo tempo. Per questo motivo si è scelto di fissare il numero delle epoche a 30. In generale, comunque, grazie alla tecnica dell'early stopping, il modello tende a interrompere prima l'addestramento sul dataset.

**Batch Size.** Quando un modello viene addestrato su un dataset di grandi dimensioni, anzichè utilizzare il classico algoritmo di discesa del gradiente (GD), si consiglia una sua versione più particolare, il *mini-batch gradient descent*. L'idea è che l'aggiornamento dei pesi non avviene dopo che il modello ha calcolato l'output per tutto il dataset, ma dopo che lo ha fatto solo su una sua porzione, detto *batch*. Questa tecnica rappresenta un compromesso tra il GD e lo Stochastic GD, in cui si segue la direzione di un solo esempio scelto casualmente. In tal modo l'addestramento del modello risulta più veloce perchè, nel caso di grandi dataset, è necessario molto tempo per calcolare il gradiente su tutti i dati prima di eseguire un singolo aggiornamento. La scelta del batch è dunque legata alla grandezza del dataset: un batch di piccola dimensione comporta aggiornamenti più frequenti, ma più rumorosi, il che può portare a una convergenza più irregolare, mentre uno di dimensione più grande porta ad aggiornamenti meno frequenti e più stabili, ma richiedono però più memoria. Nella nostra trattazione, si sceglie un batch pari a 32.

**Loss function.** Abbiamo trattato le funzioni di perdita più utilizzate nel capitolo 1. Per quanto riguarda il caso multiclasse, è possibile effettuare due scelte equivalenti:

- utilizzare la *categorical cross-entropy* (CCE (1.43)) e di conseguenza eseguire il one-hot encoding delle etichette 0, 1, 2 nei vettori  $[1, 0, 0]$ ,  $[0, 1, 0]$  e  $[0, 0, 1]$ ;
- utilizzare la *sparse categorical cross-entropy*, funzione che opera esattamente come la CCE ma che non necessita della codifica delle etichette.

Per quanto riguarda il caso binario, per lo sbilanciamento del dataset si utilizza la *Score-Oriented loss function* (1.42), che ricordiamo esegue l'addestramento in modo da ottimizzare lo skill score  $TSS$  (1.54), metrica particolarmente adatta a valutare le performance di un modello su dataset fortemente sbilanciati.



# Capitolo 4

## Analisi dei risultati

In quest'ultimo capitolo, vogliamo analizzare i risultati ottenuti applicando il modello definito nel capitolo precedente al problema dei brillamenti solari. In particolare, sia per il caso multiclasse che per il caso binario, si confronta lo studio di quattro diversi Test, in cui si considera lo stesso modello addestrato su quattro diversi dataset, costruiti in modo da aumentare gradualmente la percentuale di campioni della classe minoritaria. Si analizza, per primo, il caso multiclasse, per poi affrontare lo studio del caso binario. Infine, nell'ultima sezione, si fanno alcune considerazioni su possibili sviluppi futuri che potrebbero derivare da questa Tesi.

### 4.1 Il caso multiclasse

Come già detto nel capitolo precedente, il dataset presenta un forte sbilanciamento. Oltre a ciò, dalla tabella 3.1 si nota che i campioni della classe meno rappresentata, cioè la classe  $X$ , risultano in media circa lo 0.15% del totale. Di conseguenza, una previsione di tale classe risulta molto difficile anche per modelli avanzati come il Transformer senza adeguate tecniche di bilanciamento dei dati. A tal fine si è deciso di inglobare le classi  $M$  e  $X$  in un'unica classe, accontentandoci di prevedere la possibile occorrenza di almeno un flare di classe  $\geq M$ . D'altra parte, poichè i flare di classe  $B$  sono spesso considerati flare minori, creiamo un'unica classe  $Q$  (quiet) che incorpori gli eventi di classe  $A$  e  $B$ . Di conseguenza, il caso multiclasse equivale ad un problema di previsione

su tre classi cui assegniamo le seguenti etichette

$$\begin{cases} B, Q \rightarrow 0 \\ C \rightarrow 1 \\ M, X \rightarrow 2 \end{cases}$$

#### 4.1.1 Test 1

Il primo dataset che utilizziamo per la validazione del modello è costruito nel modo seguente:

- il training set è costituito da tutti gli esempi presenti nella partizione 1 e chiediamo che questo costituisca il 70% del totale del dataset; dalla tabella 3.1 vediamo che il numero di campioni nel training set è pari a  $n_{train} = 69189$ , dunque il numero totale di esempi nel dataset che vogliamo costruire è

$$N = 98837 \approx \frac{69189}{0.7}.$$

- il validation set è costituito da esempi della partizione 2, selezionati in modo casuale in modo che il set contenga circa il 15% degli esempi totali; di conseguenza  $n_{val} = 14824$ .
- il test set viene costruito a partire dalla partizione 3, selezionando gli esempi in modo casuale da costituire circa il 15% del totale; di conseguenza anche  $n_{test} = 14824$ .

I campioni per il validation vengono selezionati in modo che le percentuali delle classi  $Q, B = 0$ ,  $C = 1$  e  $M, X = 2$  siano mantenute come nel training. Per il test set, invece, si è scelto di riportare una maggiore percentuale di campioni della classe  $M, X$ , in modo da poter analizzare meglio il comportamento del modello sulle suddette classi. Si riportano i dati relativi ai tre set nelle seguenti tabelle.

TRAINING SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	61743	6266	1180	69189
Percentuale	89.24%	9.06%	1.70%	

Tabella 4.1: Campioni nel training set per il Test 1

VALIDATION SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	13055	1530	239	14824
Percentuale	88.07%	10.32%	1.61%	

Tabella 4.2: Campioni nel validation set per il Test 1

TEST SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	12313	2011	500	14824
Percentuale	83.06%	13.57%	3.37%	

Tabella 4.3: Campioni nel test set per il Test 1

Si osserva subito il forte sbilanciamento delle classi, con la classe 2 che viene rappresentata solo al 1.70% e al 1.61% rispettivamente nel training e validation set. Di conseguenza, possiamo aspettarci una previsione poco accurata del modello sulla suddetta classe. Si riportano le matrici di confusione sul training, validation e test set, in cui sulle righe si considerano i true labels, mentre sulle colonne i predicted labels.

$$CM_{train} = \begin{pmatrix} 61463 & 278 & 2 \\ 4998 & 1265 & 3 \\ 430 & 741 & 9 \end{pmatrix} \quad (4.1)$$

$$CM_{val} = \begin{pmatrix} 12981 & 74 & 0 \\ 1171 & 356 & 3 \\ 93 & 144 & 2 \end{pmatrix} \quad (4.2)$$

$$CM_{test} = \begin{pmatrix} 12138 & 172 & 3 \\ 1572 & 438 & 1 \\ 240 & 257 & 3 \end{pmatrix} \quad (4.3)$$

Come ci si aspettava, si nota che pochissimi campioni di classe 2 sono classificati correttamente, essendo la maggior parte predetti nella classe 1; lo stesso comportamento si nota per la classe 1, predetta maggiormente come classe 0. Ciò è dovuto ai pochi campioni delle suddette classi che vengono presentati al modello, il quale si specializza sui campioni di classe 0. Per un'ulteriore analisi, si riportano le metriche definite nella sezione 1.4 del modello sui tre set di dati. Per quanto detto in tale sezione, le metriche che ci interessano maggiormente studiare sono quelle calcolate con la macro-averaging; tuttavia,

per completezza, si riportano anche i valori ottenuti con il micro-averaging. Di conseguenza, per questo Test e per i successivi, andremo ad analizzare i risultati sulle macro metriche.

<b>Metriche sul training set</b>		
Metrica	Macro	Micro
Accuracy	0.9378	0.9067
<i>TSS</i>	0.1532	0.8601
Recall	0.4017	0.9067
Specificity	0.7516	0.9534
Precision	0.7052	0.9067
Balanced Accuracy	0.1509	0.4322
F1 score	0.5118	0.9067
CSI	0.3654	0.8294
HSS	0.2192	0.8601

Tabella 4.4: Metriche sul training set per il Test 1

<b>Metriche sul validation set</b>		
Metrica	Macro	Micro
Accuracy	0.9332	0.8998
<i>TSS</i>	0.1681	0.8497
Recall	0.4118	0.8998
Specificity	0.7563	0.9499
Precision	0.6438	0.8998
Balanced Accuracy	0.1557	0.4274
F1 score	0.5023	0.8998
CSI	0.3728	0.8179
HSS	0.2364	0.8497

Tabella 4.5: Metriche sul validation set per il Test 1

Metriche sul test set		
Metrica	Macro	Micro
Accuracy	0.8990	0.8486
<i>TSS</i>	0.1514	0.7728
Recall	0.4032	0.8486
Specificity	0.7482	0.9243
Precision	0.6013	0.8486
Balanced Accuracy	0.1508	0.3922
F1 score	0.4827	0.8486
CSI	0.3483	0.7370
HSS	0.2034	7728

Tabella 4.6: Metriche sul test set per il Test 1

Il modello mostra un'alta accuratezza su tutti e tre i set di dati, ma dal momento che le classi sono sbilanciate, questa metrica non può determinare, da sola, la performance del modello. La metrica principale che consideriamo è la macro *TSS*, la quale risulta molto bassa sui tre set. Ne deduciamo che quello che consideriamo non è un buon modello: da una parte il valore alto dell'accuracy è dovuto alla buona classificazione della classe maggioritaria, mentre il basso valore della *TSS* indica che il modello non riesce a classificare quelle minoritarie. Si nota questo comportamento anche dai valori della Recall, che resta intorno allo 0.4 e della Specificity, che risulta un po' più alta, sullo 0.75. I valori delle altre metriche, come Balanced Accuracy, HSS, CSI e F1-score, i quali indicano l'andamento generale del modello, risultano molto bassi. Ne deduciamo che il modello addestrato su un dataset così fortemente sbilanciato non riesce a distinguere le classi meno rappresentate, specialmente la classe 2 che rappresenta solo l'1.7% del totale.

L'idea per i prossimi Test è quella di modificare le percentuali che rappresentano ciascuna classe, per capire se il modello riesce ad apprendere meglio con più esempi. In particolare proveremo a diminuire i campioni della classe maggioritaria (Test 2), aumentare quelli della classe minoritaria (Test 3) e combinare queste due tecniche (Test 4).

### 4.1.2 Test 2

Dal momento che gli esempi di classe 0 risultano molto superiori rispetto agli altri, potrebbe accadere che il modello si specializzi troppo su di essi. Di conseguenza, passiamo ad un secondo approccio in cui andiamo a diminuire i campioni di classe 0 dal training e validation set in modo da aumentare la percentuale degli esempi delle altre classi. Per poter comparare al meglio la performance del modello sui diversi dataset, il test set viene lasciato invariato. Modifichiamo dunque il set di campioni del Test 1 in modo che gli esempi di classe 2 siano pari al 5% del totale nel training e nel validation. In questo modo, il numero degli esempi totali per il training set risulta

$$n_{train} = \frac{1180}{0.05} = 23600$$

mentre per il validation

$$n_{val} = \frac{239}{0.05} = 4780$$

Per quanto riguarda le classi, otteniamo i seguenti risultati:

TRAINING SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	16154	6266	1180	23600
Percentuale	68.45%	26.55%	5.00%	

Tabella 4.7: Campioni nel training set per il Test 2

VALIDATION SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	3011	1530	239	4780
Percentuale	62.99%	32.01%	5.00%	

Tabella 4.8: Campioni nel validation set per il Test 2

Il test set risulta invariato, quindi si fa riferimento alla tabella 4.3. Risulta ovvio che le percentuali scelte per il Test 1, in questo caso, risultano differenti, avendo diminuito il numero di esempi per il training e il validation, ma lasciando invariato il test, infatti

$$\text{Dataset 2: 43204 esempi} \left\{ \begin{array}{l} \text{train: 23600} \rightarrow 54.26\% \\ \text{val: 4780} \rightarrow 11.06\% \\ \text{test: 14824} \rightarrow 34.31\% \end{array} \right.$$

Le matrici di confusione ottenute su training, validation e test set sono sotto riportate.

$$CM_{train} = \begin{pmatrix} 15326 & 828 & 0 \\ 2259 & 3826 & 181 \\ 44 & 849 & 287 \end{pmatrix} \quad (4.4)$$

$$CM_{val} = \begin{pmatrix} 2823 & 187 & 1 \\ 607 & 825 & 98 \\ 17 & 161 & 61 \end{pmatrix} \quad (4.5)$$

$$CM_{test} = \begin{pmatrix} 11469 & 831 & 13 \\ 698 & 1283 & 30 \\ 38 & 411 & 51 \end{pmatrix} \quad (4.6)$$

Rispetto al caso precedente, si nota che ci sono più campioni classificati nella classe 2, anche se non correttamente, segno che il modello inizia a rilevare la suddetta classe. Inoltre in tutte e tre le matrici, guardando alla seconda riga e in particolare all'elemento centrale, si osserva che, per la classe 1, la maggior parte dei campioni è stata classificata correttamente. Dunque, il modello con un numero minore di campioni di classe 0 riesce a distinguere meglio le altre due. Infine, si riportano le tabelle con le metriche macro micro.

<b>Metriche sul training set</b>		
Metrica	Macro	Micro
Accuracy	0.8825	0.8237
<i>TSS</i>	0.4628	0.7355
Recall	0.6009	0.8237
Specificity	0.8620	0.9118
Precision	0.7260	0.8237
Balanced Accuracy	0.2590	0.3755
F1 score	0.6575	0.8237
CSI	0.5076	0.7002
HSS	0.5131	0.7355

Tabella 4.9: Metriche sul training set per il Test 2

<b>Metriche sul validation set</b>		
Metrica	Macro	Micro
Accuracy	0.8506	0.7759
<i>TSS</i>	0.4168	0.6639
Recall	0.5773	0.7759
Specificity	0.8395	0.8880
Precision	0.6345	0.7759
Balanced Accuracy	0.2423	0.3445
F1 score	0.6046	0.7759
CSI	0.4655	0.6339
HSS	0.4512	0.6639

Tabella 4.10: Metriche sul validation set per il Test 2

<b>Metriche sul test set</b>		
Metrica	Macro	Micro
Accuracy	0.9091	0.8637
<i>TSS</i>	0.4261	0.7955
Recall	0.5571	0.8637
Specificity	0.8690	0.9318
Precision	0.6635	0.8637
Balanced Accuracy	0.2421	0.4024
F1 score	0.6057	0.8637
CSI	0.4557	0.7600
HSS	0.4363	0.7955

Tabella 4.11: Metriche sul test set per il Test 2

Per il training e validation, si nota subito una riduzione del valore dell'accuracy rispetto al caso precedente, in cui un alto numero di campioni di classe 0 erano stati classificati correttamente, numero che si è abbassato in questo Test. D'altra parte, nel test set questo valore è leggermente più alto, quindi il modello compensa la riduzione dei classificati correttamente di classe 0 con quelli di classe 1 e 2. Inoltre, si osserva che la qualità delle previsioni è più alta in tutti i set: guardando al valore del *TSS* macro, osserviamo un netto miglioramento, poichè il modello riesce a distinguere correttamente più campioni di classe 1 e 2, risultando più bilanciato. Questo comportamento si nota

anche dal valore più alto della macro Recall e Specificity. Anche per quanto riguarda le metriche macro che evidenziano il comportamento generale del modello, cioè F1-score, CSI, HSS e Balanced Accuracy, si nota un miglioramento dei valori. Ne deduciamo che, diminuendo il numero di campioni della classe maggioritaria, il modello riesce effettivamente a rilevare le altre due classi meno rappresentate.

### 4.1.3 Test 3

In questo nuovo dataset vogliamo sempre aumentare la percentuali dei campioni della classe minoritaria, ma anzichè diminuire la classe maggiore come fatto nel caso precedente, proviamo ad aumentare il numero effettivo di esempi di classe 2 dal quale il modello può imparare. L'idea è quella di considerare i campioni della partizione 4 e 5, ancora inutilizzate, e aggiungerli al training set. In questo modo la percentuale minoritaria risulterà più bassa rispetto al caso precedente, ma in questo caso il modello ha un numero maggiore di esempi dal quale imparare. Mantenendo il validation set e il test set come nel primo studio, il training set viene modificato in modo da ottenere i seguenti effetti

TRAINING SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	61743	6266	2964	70973
Percentuale	87.00%	8.83%	4.18%	

Tabella 4.12: Campioni nel training set per il Test 3

Si riportano le matrici di confusione sui tre set:

$$CM_{train} = \begin{pmatrix} 59946 & 1765 & 32 \\ 2687 & 3018 & 561 \\ 207 & 964 & 1793 \end{pmatrix} \quad (4.7)$$

$$CM_{val} = \begin{pmatrix} 12542 & 464 & 49 \\ 727 & 504 & 299 \\ 37 & 60 & 142 \end{pmatrix} \quad (4.8)$$

$$CM_{test} = \begin{pmatrix} 11646 & 514 & 153 \\ 793 & 791 & 427 \\ 46 & 154 & 300 \end{pmatrix} \quad (4.9)$$

Si osserva che, per la prima volta, la maggior parte dei campioni di classe 2 è classificata correttamente in tutte e tre le matrici. Tuttavia si osserva che il modello fa più fatica a riconoscere i campioni di classe 1, confondendoli spesso con la classe 0, segno che questa è ancora dominante. Tuttavia, in generale, il modello riesce a distinguere le tre classi. Per quanto riguarda le metriche, si osservano i risultati riportati nelle seguenti tabelle.

<b>Metriche sul training set</b>		
Metrica	Macro	Micro
Accuracy	0.9416	0.9124
<i>TSS</i>	0.5643	0.8686
Recall	0.6858	0.9124
Specificity	0.8785	0.9562
Precision	0.7435	0.9124
Balanced Accuracy	0.3013	0.4362
F1 score	0.7135	0.9124
CSI	0.5890	0.8389
HSS	0.6021	0.8686

Tabella 4.13: Metriche sul training set per il Test 3

<b>Metriche sul validation set</b>		
Metrica	Macro	Micro
Accuracy	0.9264	0.8896
<i>TSS</i>	0.4630	0.8345
Recall	0.6281	0.8896
Specificity	0.8349	0.9448
Precision	0.5742	0.8896
Balanced Accuracy	0.2622	0.4203
F1 score	0.5999	0.8896
CSI	0.4650	0.8012
HSS	0.4262	0.8345

Tabella 4.14: Metriche sul validation set per il Test 3

Metriche sul test set		
Metrica	Macro	Micro
Accuracy	0.9061	0.8592
<i>TSS</i>	0.5041	0.7888
Recall	0.6464	0.8592
Specificity	0.8577	0.9296
Precision	0.6053	0.8592
Balanced Accuracy	0.2772	0.3994
F1 score	0.6252	0.8592
CSI	0.4862	0.7532
HSS	0.4747	0.7888

Tabella 4.15: Metriche sul test set per il Test 3

Al contrario del test precedente, per il training e validation, si nota un aumento del valore dell'accuracy rispetto al caso precedente, mentre nel test set questo valore è leggermente più basso. D'altra parte, si osserva un aumento nel valore del *TSS* macro, specialmente per il training e il test, per cui il modello risulta ancora più equilibrato nelle previsioni corrette rispetto al caso precedente. Anche i valori di Recall sono sempre più alti, mentre quelli della Specificity rimangono abbastanza uguali. Infine, per quanto riguarda le altre metriche, per ognuna si nota un miglioramento nel training e test rispetto al Test 2, mentre per il validation restano più o meno le stesse. In conclusione, il modello addestrato sul Test 3 risulta avere una performance generale migliore e un buon bilanciamento.

#### 4.1.4 Test 4

L'ultimo studio riguarda la combinazione dei due casi precedenti in un unico, diminuendo la percentuale di campioni della classe maggioritaria e al tempo stesso aumentando il numero di esempi di quella minoritaria.

TRAINING SET	0 (Q,B)	1 (C)	2 (M,X)	Totale
Numero	16154	6266	2964	25384
Percentuale	63.64%	24.68%	11.68%	

Tabella 4.16: Campioni nel training set per il Test 4

Si riportano le matrici di confusione sui tre set:

$$CM_{train} = \begin{pmatrix} 14694 & 1457 & 3 \\ 1102 & 4910 & 254 \\ 17 & 1179 & 1768 \end{pmatrix} \quad (4.10)$$

$$CM_{val} = \begin{pmatrix} 2594 & 408 & 9 \\ 393 & 886 & 251 \\ 28 & 85 & 126 \end{pmatrix} \quad (4.11)$$

$$CM_{test} = \begin{pmatrix} 10528 & 1595 & 190 \\ 399 & 1147 & 465 \\ 8 & 191 & 301 \end{pmatrix} \quad (4.12)$$

Per la prima volta, in ciascun set, la maggior parte dei campioni di ciascuna classe è stata classificata correttamente. In questo caso il modello riesce a classificare meglio i campioni di classe 1 rispetto al caso precedente, mentre non si notano miglioramenti per la classe 2. D'altra parte si osserva che vi sono molti più campioni di classe 0 la cui classificazione è errata e questo segue dal fatto che ne abbiamo diminuito in modo significativo la percentuale (dall'89.24% del Test 1 63.64% al del Test 4). Infine osserviamo i risultati ottenuti per le metriche di valutazione.

<b>Metriche sul training set</b>		
Metrica	Macro	Micro
Accuracy	0.8946	0.8419
<i>TSS</i>	0.6730	0.7629
Recall	0.7632	0.8419
Specificity	0.9098	0.9210
Precision	0.8177	0.8419
Balanced Accuracy	0.3472	0.3877
F1 score	0.7895	0.8419
CSI	0.6504	0.7270
HSS	0.6882	0.7629

Tabella 4.17: Metriche sul training set per il Test 4

<b>Metriche sul validation set</b>		
Metrica	Macro	Micro
Accuracy	0.8363	0.7544
<i>TSS</i>	0.5070	0.6316
Recall	0.6559	0.7544
Specificity	0.8510	0.8772
Precision	0.6098	0.7544
Balanced Accuracy	0.2791	0.3309
F1 score	0.6320	0.7544
CSI	0.4821	0.6056
HSS	0.4755	0.6316

Tabella 4.18: Metriche sul validation set per il Test 4

<b>Metriche sul test set</b>		
Metrica	Macro	Micro
Accuracy	0.8719	0.8079
<i>TSS</i>	0.5601	0.7118
Recall	0.6758	0.8079
Specificity	0.8843	0.9039
Precision	0.5562	0.8079
Balanced Accuracy	0.2988	0.3651
F1 score	0.6102	0.8079
CSI	0.4635	0.6777
HSS	0.4387	0.7118

Tabella 4.19: Metriche sul test set per il Test 4

Sia per il training che per il validation, si osserva un aumento in tutte le macro metriche eccetto che per l'accuracy, dovuto al fatto che molti campioni di classe 0 non sono stati classificati correttamente come nel test precedente. D'altra parte, avendo un aumento per le altre due classi, si ha un miglioramento per le restanti metriche. Per quanto riguarda il test set, invece, si nota un aumento rispetto al Test 3 solo per il TSS, la Recall, la Specificity e la Balanced Accuracy: essendo tali metriche le più utili nel caso si volesse studiare la performance di un modello su dataset sbilanciati, ne deduciamo che il modello addestrato sul Test 4 risulta il migliore per i nostri scopi.

### 4.1.5 Analisi dei risultati

Per concludere il caso multiclasse, vogliamo analizzare i risultati ottenuti nei quattro Test, in particolare confrontando tra loro i valori delle macro metriche per training, validation e test separatamente.

Per quanto riguarda il training set, si riassumono i risultati dei quattro Test nella seguente tabella, in cui vengono evidenziati i valori più alti per ciascuna metrica.

**TRAINING SET**

Macro-Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.9378	0.8825	0.9416	0.8946
<i>TSS</i>	0.1532	0.4628	0.5643	0.6730
Recall	0.4017	0.6009	0.6858	0.7632
Specificity	0.7516	0.8620	0.8785	0.9098
Precision	0.7052	0.7260	0.7435	0.8117
Balanced Accuracy	0.1509	0.2590	0.3013	0.3472
F1 score	0.5118	0.6575	0.7135	0.7895
CSI	0.3654	0.5076	0.6021	0.6504
HSS	0.2192	0.5131	0.5890	0.6882

Tabella 4.20: Confronto delle metriche sul training set

Analizziamo separatamente ciascuna metrica:

- In tutti i Test si mantengono valori alti di accuracy, con il valore più alto ottenuto nel Test 3 (0.9416). Questo suggerisce che, nonostante i cambiamenti nella distribuzione delle classi, il modello riesce a mantenere una buona capacità di classificazione, anche se la riduzione dei campioni di classe 0 inizia a influenzare negativamente le prestazioni globali nei Test 2 e 4.
- Per quanto riguarda il *TSS*, il valore più basso è nel Test 1 (0.1532), il quale indica la bassa capacità del modello di distinguere le classi poco rappresentate. Tuttavia, il TSS migliora significativamente nei Test 2, 3 e 4, con il valore massimo di 0.6730 nel Test 4. Questo dimostra che l'aumento delle classi minoritarie nei dataset successivi porta a una maggiore capacità discriminativa del modello.

- Come il *TSS*, anche la Recall mostra un miglioramento nei vari Test. Dal valore più basso del Test 1 (0.4017) si raggiunge il massimo nel Test 4 con un recall di 0.7632. Anche in questo caso, l'aumento della percentuale dei campioni di classe 1 e 2 migliora sensibilmente la capacità del modello di individuare correttamente le suddette classi.
- Anche per la *a specificity* si osserva un graduale aumento, anche se risulta piuttosto alta in tutti i Test, da un minimo di 0.7516 ad un massimo di 0.9098. Questo riflette la capacità del modello di mantenere buone prestazioni nel rilevare correttamente i campioni negativi rispetto a ciascuna classe, nonostante le variazioni nella distribuzione delle classi.
- La precision risulta, come le altre metriche, migliorare nei Test: a partire da un valore di 0.7052 nel Test 1 cresce fino a 0.8117 nel Test 4. Questo indica che con l'aumento della rappresentazione delle classi minoritarie, il modello non solo identifica meglio queste classi (che vediamo dalla Recall) ma riduce anche il numero di falsi positivi.
- Per la *balanced accuracy*, che ricordiamo essere la media aritmetica di recall e specificity, è molto bassa in tutti i Test, nonostante si possa osservare un graduale aumento. Ciò suggerisce che, nonostante il miglioramento nelle metriche di Precision e Recall, il bilanciamento tra la capacità di riconoscere le classi potrebbe essere ulteriormente migliorato.
- Anche l'*F1 score*, che combina precision e recall, mostra un chiaro miglioramento nei diversi Test. Dal valore iniziale di 0.5118 nel Test 1, l'*F1 score* raggiunge 0.7895 nel Test 4. Questo dimostra che l'equilibrio tra precision e recall è migliorato costantemente, in particolare quando si correggono le distribuzioni di classe.
- Il *CSI* segue lo stesso andamento delle altre metriche: da un valore di 0.3654 nel Test 1 aumenta fino a 0.6504 nel Test 4. Questo incremento dimostra che l'incremento delle percentuali delle classi minoritarie migliora la capacità del modello di classificare correttamente tali istanze, riducendo il numero di falsi positivi e falsi negativi.
- Infine, anche per l'*HSS*, il quale misura l'efficacia del modello rispetto a un'assegnazione casuale, si osserva un miglioramento, passando da 0.2192 nel Test 1 a 0.6882 nel Test 4.

Nel complesso, l'analisi sul training set mostra che la ridotta rappresentazione delle classi 1 e 2 nel Test 1 penalizza il modello, causando una scarsa capacità discriminativa, come evidenziato dal TSS, Recall, Specificity e Balanced Accuracy. D'altra parte, aumentando la percentuale di queste classi sul totale, tutte le metriche migliorano sensibilmente. In particolare, il Test 4, che rappresenta un bilanciamento tra la riduzione della classe 0 e l'aumento delle classi 1 e 2, mostra i migliori risultati complessivi. Questi risultati evidenziano l'importanza di una distribuzione più equilibrata delle classi per garantire una performance ottimale del modello.

Per quanto riguarda il validation set, si hanno i seguenti risultati.

### VALIDATION SET

Macro-Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.9332	0.8506	0.9264	0.8363
<i>TSS</i>	0.1682	0.4168	0.4630	0.5070
Recall	0.4118	0.5773	0.6281	0.6559
Specificity	0.7563	0.8395	0.8349	0.8510
Precision	0.6438	0.6345	0.5742	0.6098
Balanced Accuracy	0.1557	0.2423	0.2622	0.2791
F1 score	0.5023	0.6046	0.5999	0.6320
CSI	0.3728	0.4512	0.4650	0.4821
HSS	0.2364	0.4655	0.4262	0.4755

Tabella 4.21: Confronto delle metriche sul validation set

Si osservano i seguenti fatti:

- L'accuracy più alta si raggiunge nel Test 1 (0.9332), e risulta leggermente più bassa nei Test 2 (0.8506) e 4 (0.8363), che presentano le percentuali di classi 1 e 2 più elevate. Questo è dovuto al fatto che nel Test 1 ci sono molti campioni della classe maggioritaria classificati correttamente, ma il numero di tali campioni diminuisce con l'aumentare delle percentuali delle altre classi, da cui si ha un diminuito nell'accuracy.
- Al contrario, il *TSS* mostra un miglioramento nei vari Test. Nel Test 1, il valore è il più basso (0.1682), indicando che il modello non riesce a distinguere bene le classi 1 e 2. Tuttavia, con l'aumento delle percentuali

di queste classi nei Test 2, 3 e 4, il TSS aumenta fino a raggiungere 0.5070 nel Test 4, dimostrando una migliore capacità discriminativa.

- Lo stesso comportamento si osserva nella Recall, che dal valore 0.4118 del Test 1, aumenta fino a 0.6559 nel Test 4. Di conseguenza, con un aumento graduale dei campioni delle classi minoritarie, il modello diventa progressivamente più abile nel riconoscerle, riducendo il numero di falsi negativi.
- A differenza delle altre metriche, la Specificity risulta abbastanza costante nei vari test, raggiungendo il massimo nel Test 4 (0.8510). Quindi, nonostante le variazioni nella distribuzione delle classi, il modello riesce comunque a riconoscere bene le istanze negative di ciascuna classe rispetto alle altre.
- Per la Precision, il valore più alto si ottiene nel Test 1 (0.6438), mentre nei Test 2, 3 e 4 scende gradualmente, con il valore più basso registrato nel Test 3 (0.5742): mentre l'aumento dei campioni di classe 1 e 2 migliora il recall, potrebbe causare un aumento dei falsi positivi, influenzando negativamente la precision.
- Il valore più basso della balanced accuracy si ottiene nel Test 1 con un valore di 0.1557, mentre il più alto è 0.2791 nel Test 4. Anche se si ha un miglioramento nei Test, il valore finale resta comunque piuttosto basso, suggerendo che, nonostante i progressi fatti nel riconoscere le classi minoritarie, il modello continua a non essere completamente bilanciato nelle sue predizioni.
- Anche l'F1 score mostra un leggero miglioramento attraverso i Test, passando da 0.5023 nel Test 1 a 0.6320 nel Test 4. Il valore massimo nel Test 4 indica che, nonostante la precision sia calata, il miglioramento della Recall ha avuto un effetto positivo sulla capacità del modello di mantenere un buon equilibrio tra falsi positivi e falsi negativi.
- Il CSI parte da un valore di 0.3728 nel Test 1 e cresce progressivamente fino a 0.4821 nel Test 4: l'aumento dei campioni delle classi minoritarie ha migliorato la capacità complessiva del modello di fare previsioni accurate.
- Infine, anche l'HSS segue un andamento crescente, passando da 0.2364 nel Test 1 a 0.4755 nel Test 4, perciò il modello diventa progressiva-

mente più efficace man mano che le classi minoritarie vengono meglio rappresentate nel dataset.

In generale, l'andamento delle metriche sul validation set riflette quanto già osservato nel training set: la performance del modello migliora significativamente quando si incrementa la rappresentazione delle classi minoritarie, in particolare la classe 2. Tuttavia, il Test 1, che ha la distribuzione più sbilanciata (con una percentuale molto bassa di classi 1 e 2), mantiene un'accuracy alta ma scarse prestazioni in altre metriche come Recall e TSS. Nei Test 2, 3 e 4, le metriche migliorano in termini di capacità discriminativa e bilanciamento tra le classi, come evidenziato dai valori più alti di TSS, Specificity e Balanced Accuracy. Il Test 4, si dimostra ancora il migliore in termini di bilanciamento tra le classi, portando a migliori prestazioni complessive del modello anche sul validation set.

Analizziamo infine il comportamento del modello sul test set, il quale, ricordiamo, è lo stesso per tutti i Test.

#### TEST SET

Macro-Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.8990	0.9091	0.9061	0.8719
<i>TSS</i>	0.1514	0.4261	0.5041	0.5601
Recall	0.4032	0.5571	0.6464	0.6758
Specificity	0.7482	0.8690	0.8577	0.8843
Precision	0.6013	0.6635	0.6053	0.5562
Balanced Accuracy	0.1508	0.2421	0.2772	0.2988
F1 score	0.4827	0.6046	0.6252	0.6102
CSI	0.3483	0.4557	0.4747	0.4635
HSS	0.2034	0.4263	0.4862	0.4387

Tabella 4.22: Confronto delle metriche sul test set

Si osserva che, in questo caso, oltre al Test 4 emerge anche il Test 3:

- L'accuracy mostra valori abbastanza vicini, con un massimo di 0.9091 nel Test 2 e un minimo di 0.8719 nel Test 4. Nel complesso, il modello mantiene una buona capacità di classificazione generale in tutti i casi, anche se nel Test 4 è leggermente meno efficace.

- Il *TSS* mostra lo stesso comportamento osservato per il training e validation: il valore più basso è nel Test 1 (0.1514) mentre nei Test 2, 3 e 4 migliora significativamente, con il Test 4 che raggiunge il valore massimo di 0.5601.
- Allo stesso modo la Recall presenta un miglioramento graduale: nel Test 1 ha il valore più basso (0.4032) e raggiunge il massimo nel Test 4 (0.6758). L'aumento della rappresentazione delle classi minoritarie nei dati di addestramento porta a un miglioramento sostanziale nella capacità del modello di identificare correttamente queste classi anche su dati mai visti.
- La specificity è elevata in tutti i casi, con valori che vanno da 0.7482 nel Test 1 a un massimo di 0.8843 nel Test 4, dimostrando che l'aggiunta di campioni di classi minoritarie non compromette la capacità del modello di riconoscere correttamente i campioni negativi di ciascuna classe rispetto alle altre.
- La precision varia leggermente tra i test: il valore più alto si ottiene nel Test 2 (0.6635), seguito dal Test 3 (0.6053). Nel Test 4, la precision è la più bassa (0.5562), dunque, sebbene la Recall e il *TSS* migliorino con l'aumento delle classi minoritarie, ciò avviene a scapito della precision. Un possibile motivo è che l'aumento della Recall comporta un incremento dei falsi positivi, riducendo così la precision.
- La balanced accuracy migliora nei vari Test come per validation e test set. Parte da un valore molto basso nel Test 1 (0.1508) e raggiunge il valore più alto nel Test 4 (0.2988), in cui il modello presenta una capacità più bilanciata di identificare correttamente le classi.
- Anche per l'*F1* score si osserva un incremento nei Test 2, 3 e 4, mentre raggiunge il minimo nel Test 1 (0.4827). A differenza di quanto osservato nel training e validation, il massimo non si ottiene nel Test 4 ma nel Test 3, probabilmente a causa della diminuzione della precision. Ciò suggerisce che, nonostante il miglioramento del recall nel Test 4, il calo della precision ha limitato l'efficacia complessiva del modello nel bilanciare i falsi positivi e negativi.
- Il CSI segue una tendenza simile all'*F1* score: parte da 0.3483 nel Test 1 e aumenta nei Test successivi, ma raggiunge il valore più alto nel Test 3

(0.4747). Il Test 4 ha un CSI leggermente inferiore (0.4635), a conferma di quanto detto sopra.

- Infine, si osserva lo stesso comportamento per l'HSS, il quale dal valore più basso di 0.2034 nel Test 1 raggiunge il valore massimo di 0.4862 nel Test 3, mentre scende leggermente nel Test 4, probabilmente a causa della riduzione della precision. Tuttavia, anche con questa leggera diminuzione, i Test 3 e 4 dimostrano che il modello migliora sensibilmente rispetto all'assegnazione casuale.

In conclusione, possiamo osservare anche per il test set lo stesso comportamento evidenziato sul training e validation set, dove il modello migliora gradualmente con l'aumento della rappresentazione delle classi minoritarie. In particolare, l'aumento della Recall e del *TSS* nei vari Test successivi riflette una migliore capacità del modello di riconoscere correttamente le istanze delle classi 1 e 2 man a mano che vengono sempre più rappresentate. D'altro canto, il calo della precision suggerisce una maggiore tendenza del modello a generare falsi positivi ogni classe rispetto alle altre, soprattutto nei test con una distribuzione più bilanciata. Infine, il Test 4 risulta il migliore tra tutti, infatti, bilanciando efficacemente le classi nei dati di addestramento, il modello è in grado di riconoscere abbastanza le classi minoritarie anche su nuovi dati.

Si nota che, per questi Test, non sono state utilizzate tecniche di balancing che avrebbero potuto aumentare la rappresentazione delle classi minoritarie nel dataset e di conseguenza la capacità del modello di riconoscere tali classi. Nello studio del caso binario, grazie all'utilizzo della Score-Oriented loss function, otterremo risultati migliori in termini di metriche che valutano il bilanciamento del modello.

## 4.2 Il caso binario

Nel caso binario, il problema che vogliamo risolvere è capire se, data una determinata serie temporale relativa ad una regione attiva, questa darà origine o meno ad un flare. In questo senso, un evento si considera positivo (classe 1) se avverrà almeno un flare di classe maggiore o uguale a  $M$  nelle successive 24 ore altrimenti viene definito negativo (classe 0).

Anche in questo caso è presente un forte sbilanciamento del dataset, dal momento che i campioni positivi risultano nettamente inferiori rispetto ai negativi. Per questo motivo, a differenza del caso multiclasse, la funzione di

perdita utilizzata è la Score-Oriented loss function presentata nel capitolo 1. In questo modo il modello esegue l'addestramento cercando di ottimizzare lo score della *TSS* che, come abbiamo già detto, è una metrica adatta per valutare le performance su dataset sbilanciati.

### 4.2.1 Test 1

Il dataset del primo studio viene costruito in modo analogo al caso multiclasse: il set di training comprende tutti gli esempi della partizione 1 e viene considerato uguale al 70% del totale, i set di validazione e di test vengono selezionati, rispettivamente, a partire dalle partizioni 2 e 3 in modo che questi risultino entrambi il 15% del dataset totale. In questo modo otteniamo, come nel caso multiclasse, il seguente

$$\text{Dataset} \left\{ \begin{array}{ll} \text{Training set:} & 69189 \text{ campioni } (\sim 70\%) \\ \text{Validation set:} & 14825 \text{ campioni } (\sim 15\%) \\ \text{Test set:} & 14825 \text{ campioni } (\sim 15\%) \end{array} \right.$$

I tre insiemi vengono costruiti in modo da mantenere le percentuali di campioni di classe 1 in modo coerente con le partizioni considerate; si riportano i campioni dei tre set nelle seguenti tabelle.

TRAINING SET	0	1	Totale
Numero	68009	1180	69189
Percentuale	98.29%	1.71%	

Tabella 4.23: Campioni nel training set per il Test 1

VALIDATION SET	0	1	Totale
Numero	14586	239	14825
Percentuale	98.39%	1.61%	

Tabella 4.24: Campioni nel validation set per il Test 1

TEST SET	0	1	Totale
Numero	14325	500	14825
Percentuale	96.63%	3.37%	

Tabella 4.25: Campioni nel test set per il Test 1

Si osserva subito che la percentuale dei campioni che ci interessa maggiormente predire è molto bassa, di conseguenza non ci aspettiamo che il modello riesca a imparare a distinguere bene i suddetti campioni. Le matrici di confusione del modello addestrato, infatti, rispecchiano questo comportamento. Riportiamo le matrici di confusione sul training, validation e test set, ricordando che sulle righe consideriamo i true labels e sulle colonne le labels predette.

$$CM_{train} = \begin{pmatrix} 66675 & 1334 \\ 581 & 599 \end{pmatrix} \quad (4.13)$$

$$CM_{val} = \begin{pmatrix} 14211 & 375 \\ 108 & 131 \end{pmatrix} \quad (4.14)$$

$$CM_{test} = \begin{pmatrix} 13609 & 716 \\ 243 & 257 \end{pmatrix} \quad (4.15)$$

Analizziamo i risultati ottenuti riportando le tabelle delle metriche definite nella sezione 1.4

<b>Metriche sul training set</b>	
Accuracy	0.9723
<i>TSS</i>	0.4880
Recall	0.5076
Specificity	0.9804
Precision	0.3099
Balanced Accuracy	0.7440
F1 score	0.3848
CSI	0.2383
HSS	0.3715

Tabella 4.26: Metriche sul training set per il Test 1

<b>Metriche sul validation set</b>	
Accuracy	0.9674
<i>TSS</i>	0.5224
Recall	0.5481
Specificity	0.9743
Precision	0.2589
Balanced Accuracy	0.7612
F1 score	0.3517
CSI	0.2134
HSS	0.3372

Tabella 4.27: Metriche sul validation set per il Test 1

<b>Metriche sul test set</b>	
Accuracy	0.9353
<i>TSS</i>	0.4640
Recall	0.5140
Specificity	0.9500
Precision	0.2641
Balanced Accuracy	0.7320
F1 score	0.3489
CSI	0.2113
HSS	0.3186

Tabella 4.28: Metriche sul test set per il Test 1

Nel complesso, il nostro modello mostra un'alta accuratezza su tutti e tre i set di dati. Tuttavia, essendo in presenza di classi sbilanciate, tale indicatore risulta insufficiente per determinare la performance del modello. Il valore cui facciamo riferimento è il *TSS*, che nel caso del test set è 0.4640: ricordando che un valore ottimale per tale metrica è 1, ne deduciamo che il modello non riesce a distinguere bene le due classi. Inoltre, è possibile osservare dalle matrici di confusione che il numero di campioni di classe 1 classificati correttamente e non correttamente sono più o meno la stessa percentuale. Questo fatto è osservabile nel valore della Recall che per tutti i set si aggira sullo 0.5: si ricorda, infatti, che la Recall misura la capacità del modello di identificare correttamente tutte le previsioni positive. A differenza, invece, la Specificity, risulta molto alta, dal momento che indica l'abilità nel distinguere le classi negative. Questi valori

sono coerenti con il dataset sbilanciato che stiamo considerando: il training ed il validation rappresentano la classe positiva con una percentuale media dell' 1.65%, quindi il modello si specializza molto sui campioni negativi. É notevole, comunque, che il modello riesca a raggiungere un  $TSS$  che si avvicina allo 0.5 nonostante la scarsa presenza di esempi di classe 1.

Per i prossimi test vorremmo modificare il dataset in modo da ottenere diverse percentuali tra i negativi e i positivi, per capire come si comporta il modello quando si diminuiscono i campioni negativi (Test 2), quando si aumentano i campioni positivi (Test 3) e quando queste due tecniche vengono utilizzate simultaneamente (Test 4).

### 4.2.2 Test 2

Come abbiamo fatto per il caso multiclasse, per il secondo test vogliamo diminuire i campioni di classe 0 nel training e nel validation set, in modo che gli esempi positivi risultino il 5% del totale in entrambi. Il numero di campioni negativi del training set viene diminuito da 68009, come si vede dalla Tabella 4.29, a 22420, mentre per il validation si passa da 14586, (come in Tabella 4.30) a 4541. Per poter confrontare al meglio i risultati ottenuti con i diversi modelli, lasciamo invariato il test set. Si riportano set del training e validation, mentre per il test si fa riferimento alla Tabella 4.25.

TRAINING SET	0	1	Totale
Numero	22420	1180	23600
Percentuale	95.00%	5.00%	

Tabella 4.29: Campioni nel training set per il Test 2

VALIDATION SET	0	1	Totale
Numero	4541	239	4780
Percentuale	95.00%	5.00%	

Tabella 4.30: Campioni nel validation set per il Test 2

Riportiamo le matrici di confusioni sui tre set

$$CM_{train} = \begin{pmatrix} 20955 & 1465 \\ 176 & 1004 \end{pmatrix} \quad (4.16)$$

$$CM_{val} = \begin{pmatrix} 4173 & 368 \\ 21 & 218 \end{pmatrix} \quad (4.17)$$

$$CM_{test} = \begin{pmatrix} 12573 & 1752 \\ 59 & 441 \end{pmatrix} \quad (4.18)$$

e le relative metriche calcolate

<b>Metriche sul training set</b>	
Accuracy	0.9305
<i>TSS</i>	0.7855
Recall	0.8508
Specificity	0.9347
Precision	0.4066
Balanced Accuracy	0.8928
F1 score	0.5503
CSI	0.3769
HSS	0.5177

Tabella 4.31: Metriche sul training set per il Test 2

<b>Metriche sul validation set</b>	
Accuracy	0.9186
<i>TSS</i>	0.8311
Recall	0.9121
Specificity	0.9190
Precision	0.3720
Balanced Accuracy	0.9155
F1 score	0.5285
CSI	0.3591
HSS	0.4924

Tabella 4.32: Metriche sul validation set per il Test 2

Metriche sul test set	
Accuracy	0.8778
<i>TSS</i>	0.7597
Recall	0.8820
Specificity	0.8777
Precision	0.2011
Balanced Accuracy	0.8798
F1 score	0.3275
CSI	0.1958
HSS	0.2884

Tabella 4.33: Metriche sul test set per il Test 2

Si osserva subito che, rispetto al caso precedente, si ha un valore di accuratezza più basso e al tempo stesso un valore di *TSS* molto più alto. Anche i valori di Recall e Precision, legati ai campioni positivi, risultano più alti. Ne deduciamo che il modello, con un minor numero di campioni negativi, si specializza meno su questi e riesce ad adattarsi in modo da poter predire anche i campioni di classe positiva. Come ci si aspetta, i valori migliori si ottengono per il training set, ad eccezione del *TSS* e della Recall che sono poco più alti per il validation set. Questo potrebbe essere dovuto al fatto che, per il modo in cui è stato costruito, il validation set contiene molti meno esempi rispetto agli altri due set. Tuttavia, se da una parte otteniamo una miglior predizione sui campioni positivi, nel test set otteniamo valori peggiori rispetto al primo studio per quanto riguarda l’F1-score, il CSI e l’HSS: ricordiamo che il primo rappresenta la media armonica tra la Precision e la Recall di un modello di classificazione, il secondo misura l’accuratezza del modello tenendo conto anche degli errori, mentre l’ultimo misura l’abilità di un modello rispetto a una previsione casuale. Questo comportamento potrebbe essere giustificato dal fatto che ci sono molti più esempi negativi classificati non correttamente rispetto al caso precedente, proprio per la ridotta presenza di questi nel set di allenamento. Questo fatto è osservabile anche nei valori generici più bassi per la Specificity. Se, dunque, da una parte riusciamo a migliorare la predizione sui positivi, dall’altra peggioriamo quella sui negativi.

L’idea successiva è, quindi, quella di mantenere invariato il numero di campioni negativi, andando però ad aumentare il numero di positivi.

### 4.2.3 Test 3

In questo studio, consideriamo il dataset 1 e aumentiamo la percentuale di campioni positivi nel solo training set andando ad aggiungere nuovi campioni di classe 1, selezionati dalle partizioni 4 e 5. Mantenendo il validatione e il test set come nel primo caso (tabelle 4.24 e 4.25), il training set viene modificato in modo da raggiungere poco più del 4% di campioni positivi sul totale. La seguente tabella mostra la divisione del set:

TRAINING SET	0	1	Totale
Numero	68009	2964	70973
Percentuale	95.82%	4.18%	

Tabella 4.34: Campioni nel training set per il Test 3

Le confusion matrix per il training, validation e test set sono sotto riportate.

$$CM_{train} = \begin{pmatrix} 64363 & 3646 \\ 241 & 2723 \end{pmatrix} \quad (4.19)$$

$$CM_{val} = \begin{pmatrix} 13647 & 939 \\ 45 & 194 \end{pmatrix} \quad (4.20)$$

$$CM_{test} = \begin{pmatrix} 12742 & 1583 \\ 90 & 410 \end{pmatrix} \quad (4.21)$$

Le metriche calcolate sulle suddette matrici sono

Metriche sul training set	
Accuracy	0.9452
<i>TSS</i>	0.8651
Recall	0.9187
Specificity	0.9464
Precision	0.4275
Balanced Accuracy	0.9325
F1 score	0.5835
CSI	0.4120
HSS	0.5583

Tabella 4.35: Metriche sul training set per il Test 3

<b>Metriche sul validation set</b>	
Accuracy	0.9336
<i>TSS</i>	0.7473
Recall	0.8117
Specificity	0.9356
Precision	0.1712
Balanced Accuracy	0.8737
F1 score	0.2828
CSI	0.1647
HSS	0.2632

Tabella 4.36: Metriche sul validation set per il Test 3

<b>Metriche sul test set</b>	
Accuracy	0.8872
<i>TSS</i>	0.7095
Recall	0.8200
Specificity	0.8895
Precision	0.2057
Balanced Accuracy	0.8547
F1 score	0.3289
CSI	0.1968
HSS	0.2907

Tabella 4.37: Metriche sul test set per il Test 3

In questo caso, l'andamento generale per il training set risulta migliore rispetto ai due casi precedenti, raggiungendo in particolare il *TSS* più alto con 0.8651. I valori più bassi si ottengono per l'accuracy e la specificity, dal momento che il modello si specializza meno sui campioni negativi, data la presenza degli esempi positivi aggiunti. È comunque notevole che, in tutti i set, il modello riesca a raggiungere un valore abbastanza alto nel *TSS* nonostante la scarsa presenza di campioni positivi durante l'addestramento, i quali raggiungo circa il 4.18% del totale nel training set. Per quanto riguarda il validation set, solo i valori del *TSS* e del Recall sono superiori al primo studio, corentemente con il fatto che abbiamo aumentato il numero di campioni di classe 1, mentre, rispetto al secondo caso, è superiore solo nell'accuracy e nella specificity per lo stesso discorso fatto sul training. Infine, anche il test set ha raggiunto un

valore più alto del  $TSS$  e Recall rispetto al primo caso, ma non al secondo: dal momento che i campioni di classe negativi sono ancora molto superiori rispetto ai positivi, il modello predice molto bene i primi, infatti la specificity risulta superiore. L'andamento generale, espresso dai valori dell'f1-score, CSI e HSS, risulta comunque migliore.

#### 4.2.4 Test 4

Per l'ultimo studio, vogliamo combinare i due casi precedenti, considerando un dataset con una diminuzione dei campioni di classe 0 e un contemporaneo aumento di quelli di classe 1 sul training set. Il validation set fa riferimento al secondo caso (Tabella 4.30), mentre il test set rimane come in Tabella 4.25. Il training set ottenuto è il seguente, mostrato in Tabella 4.38.

TRAINING SET	0	1	Totale
Numero	22420	2964	25384
Percentuale	88.32%	11.68%	

Tabella 4.38: Campioni nel training set per il Test 4

Le confusion matrix per i tre set sono le seguenti:

$$CM_{train} = \begin{pmatrix} 21201 & 1219 \\ 206 & 2758 \end{pmatrix} \quad (4.22)$$

$$CM_{val} = \begin{pmatrix} 4228 & 313 \\ 36 & 203 \end{pmatrix} \quad (4.23)$$

$$CM_{test} = \begin{pmatrix} 12723 & 1602 \\ 71 & 429 \end{pmatrix} \quad (4.24)$$

Le metriche calcolate sulle suddette matrici sono riportate nelle seguenti tabelle.

<b>Metriche sul training set</b>	
Accuracy	0.9439
<i>TSS</i>	0.8761
Recall	0.9305
Specificity	0.9456
Precision	0.6935
Balanced Accuracy	0.9381
F1 score	0.7947
CSI	0.6593
HSS	0.7630

Tabella 4.39: Metriche sul training set per il Test 4

<b>Metriche sul validation set</b>	
Accuracy	0.9270
<i>TSS</i>	0.7804
Recall	0.8494
Specificity	0.9311
Precision	0.3934
Balanced Accuracy	0.8902
F1 score	0.5377
CSI	0.3678
HSS	0.5038

Tabella 4.40: Metriche sul validation set per il Test 4

<b>Metriche sul test set</b>	
Accuracy	0.8872
<i>TSS</i>	0.7462
Recall	0.8580
Specificity	0.8882
Precision	0.2112
Balanced Accuracy	0.8731
F1 score	0.3390
CSI	0.2041
HSS	0.3012

Tabella 4.41: Metriche sul test set per il Test 4

Si può osservare un aumento dei valori delle metriche per il training set, mentre per il validation e il test risultano di poco superiori i rates riguardanti la corretta classificazione dei positivi, mentre per i negativi si ha un miglioramento nel validation set soltanto rispetto al secondo caso. Inoltre, si nota che per il test set, a parità di accuracy 0.8872 con il terzo studio, il *TSS* di questo caso è superiore al precedente, quindi il modello è riuscito a distinguere meglio le due classi, usando meno campioni di classe 0.

## 4.2.5 Analisi dei risultati

In questa sottosezione vogliamo confrontare i risultati ottenuti dei quattro studi precedenti, comparando le metriche ottenute per ciascun set.

Per quanto riguarda il training set, nella seguente tabella si evidenziano i valori più alti per ciascuna delle metriche studiate.

<b>TRAINING SET</b>				
Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.9732	0.9305	0.9452	0.9439
<i>TSS</i>	0.4880	0.7855	0.8651	0.8761
Recall	0.5076	0.8508	0.9187	0.9305
Specificity	0.9804	0.9347	0.9464	0.9456
Precision	0.3099	0.4066	0.4275	0.6935
Balanced Accuracy	0.7440	0.8928	0.9325	0.9381
F1 score	0.3848	0.5503	0.5835	0.7947
CSI	0.2383	0.3796	0.4120	0.6593
HSS	0.3715	0.5177	0.5583	0.7630

Tabella 4.42: Confronto delle metriche sul training set

Si osserva che

- In tutti e quattro i test, i valori di accuracy sono relativamente alti, variando dal 93% al 97%. Il valore più alto è ottenuto nel primo studio, coerentemente con il fatto che sono presenti circa il 98% di campioni negativi durante l'addestramento: il modello, infatti, tende a specializzarsi sui negativi. Per tale motivo, questa metrica, da sola, non è sufficiente a determinare quale modello abbia la performance più alta.

- Rispetto al primo test, in cui il numero di positivi supera di poco l'1.5%, si nota un miglioramento nel valore della *TSS* per i test successivi: dal valore 0.4880 del Test 1 si passa a un valore molto più elevato nell'ultimo test, in cui si ottiene il più alto (0.8761). Ciò indica che il Test 4 ha una capacità di discriminazione tra le classi molto più forte rispetto ai primi test.
- Per la recall, che riflette la capacità del modello di identificare tutti i veri positivi, si osserva di nuovo una crescita significativa: si passa da un recall del 50.76% nel Test 1 a oltre il 93% nel Test 4. Questo implica che l'ultimo modello è in grado di identificare correttamente quasi tutti i casi positivi, riducendo il rischio di falsi negativi.
- Per quanto riguarda il negative rate, come ci si aspettava si nota un leggero calo nei diversi test (da 0.9804 nel Test 1 a 0.9456 nel Test 4), anche se rimane comunque a livelli molto alti, il che indica che i modelli risultano molto efficaci nel riconoscere i veri negativi.
- Per la precision, l'ultimo test continua a mostrare un miglioramento significativo rispetto agli altri, raggiungendo una precisione di 0.6935, molto più alta rispetto al Test 1, che ha un valore di 0.3099. Questo comportamento è dovuto alla diversa percentuale di campioni positivi nei due dataset, dal momento che nell'ultimo studio i positivi raggiungono poco più dell'11% del totale.
- La Balanced Accuracy, che rappresenta la media tra recall e specificity, migliora progressivamente, fino all'ultimo Test 4 che raggiunge un valore di 0.9381, dimostrando un ottimo equilibrio tra la capacità di riconoscere sia i positivi che i negativi.
- Per l'F1 Score, che rappresenta un compromesso tra precision e recall, si osserva lo stesso comportamento: l'ultimo studio è significativamente migliore rispetto agli altri test, suggerendo che il relativo addestramento del modello offre il miglior bilanciamento tra la capacità di rilevare correttamente i positivi e di evitare falsi positivi.
- Il CSI, che valuta l'accuratezza delle previsioni positive riducendo l'impatto di falsi positivi e falsi negativi, passa da 0.2383 nel Test 1 a 0.6593 nel Test 4, un aumento che conferma il miglioramento nella capacità di fare previsioni accurate.

- Infine, ricordando che l'HSS misura quanto il modello performi meglio rispetto a un'assegnazione casuale, si nota che l'ultimo test si distingue dagli altri, con un valore di 0.7630, dimostrando che le sue previsioni sono le migliori rispetto a un modello casuale.

In generale, i risultati ottenuti mostrano un netto miglioramento delle performance del modello durante l'addestramento nei vari test, con il Test 4 che emerge come il più efficace, in particolare nelle metriche più importanti come Recall, Precision, F1 Score e TSS. Questo suggerisce che il modello ha raggiunto un buon equilibrio tra la capacità di identificare correttamente i casi positivi e negativi, minimizzando al contempo sia i falsi positivi che i falsi negativi. Si noti che tali risultati sono dedotti dalla sola analisi del modello sul set di addestramento, quindi è necessario fare uno studio sugli altri due set di dati per determinare il modello migliore. È importante infatti notare che prestazioni così elevate sul training set potrebbero anche indicare un rischio di overfitting, limitando la sua capacità di generalizzare.

Per quanto riguarda il validation set, abbiamo i seguenti risultati.

#### VALIDATION SET

Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.9674	0.9186	0.9336	0.9270
<i>TSS</i>	0.5224	0.8311	0.7473	0.7804
Recall	0.5481	0.9121	0.8117	0.8494
Specificity	0.9743	0.9190	0.9356	0.9311
Precision	0.2589	0.3720	0.1712	0.3934
Balanced Accuracy	0.7612	0.9155	0.8737	0.8902
F1 score	0.3517	0.5285	0.2828	0.5377
CSI	0.2134	0.3591	0.1647	0.3678
HSS	0.3372	0.4924	0.2632	0.5038

Tabella 4.43: Confronto delle metriche sul validation set

Si fanno le seguenti osservazioni sui diversi score:

- L'accuracy raggiunge valori molto elevati per tutti i test, con valori compresi tra 91.86% e 96.74%. Il Test 1 ha la migliore accuracy, seguita dai Test 3 e 4, mentre il Test 2 ha la più bassa. Come detto in precedenza, questo è dovuto all'elevato numero di campioni negativi correttamente previsti del primo test.

- Per quanto riguarda la TSS, si nota una notevole variazione tra i test. Il Test 2 raggiunge il punteggio più alto (0.8311), indicando una buona capacità di distinzione tra le classi. Il Test 1 ha il valore più basso (0.5224), dal momento che fatica a distinguere i campioni positivi.
- Nella Recall vediamo un miglioramento significativo nei test successivi. Il Test 2 raggiunge il valore più alto (0.9121), dimostrando una buona capacità di identificare i positivi. Come per la TSS, il Test 1 ha il valore di recall più basso (0.5481).
- La specificity è relativamente alta in tutti i test, con il Test 1 che raggiunge il valore più alto (0.9743). I Test 2, 3 e 4 hanno valori leggermente inferiori ma comunque buoni. In generale, il modello è molto efficace nell'identificare i negativi, anche nei test con recall più alto.
- Riguardo la precision, il Test 4 ha il valore più alto (0.3934), mentre il Test 3 mostra un valore molto basso (0.1712), suggerendo che il modello addestrato con quel dataset ha difficoltà a mantenere l'accuratezza nelle previsioni positive, producendo molti falsi positivi.
- La Balanced Accuracy raggiunge il massimo nel Test 2 (0.9155); tuttavia anche il Test 4 si comporta bene con un valore di 0.8902, mentre il Test 1 ha un valore più basso (0.7612), suggerendo un maggiore squilibrio tra recall e specificity.
- Per l'F1 Score il Test 4 mostra un buon equilibrio, con un punteggio di 0.5377, indicando che riesce a mantenere un buon compromesso tra l'identificazione dei positivi e la riduzione dei falsi positivi. Al contrario, il Test 3 ha un F1 score basso (0.2828), il che riflette la scarsa precisione nonostante un recall relativamente alto.
- Riguardo il CSI, vediamo che il Test 4 ha il valore più alto (0.3678), il che significa che produce previsioni positive più accurate rispetto agli altri test, mentre il Test 3 ha il punteggio più basso (0.1647), confermando le sue difficoltà nel fare previsioni precise.
- Infine il punteggio dell'HSS emerge nel Test 4 come il migliore (0.5038), seguito dal Test 2. Il Test 1 e soprattutto il Test 3 mostrano valori inferiori, confermando le prestazioni meno soddisfacenti del modello addestrato su quei dataset.

In generale, la tabella 4.43 mostra che le metriche relative ai positivi variano notevolmente a seconda del dataset utilizzato per addestrare il modello, mentre quelle relative ai negativi sono sempre abbastanza alte. Il Test 4, come nel caso del training, si distingue per la sua performance bilanciata in termini di precision, recall, F1 score, risultando uno dei migliori modelli per quanto riguarda la generalizzazione sui dati di validazione. Il Test 2 mostra risultati molto buoni per il recall e il TSS, ma potrebbe presentare un numero relativamente alto di falsi positivi, come suggerito dalla precision. Il Test 3, invece, ha difficoltà nel mantenere alta la precision, risultando meno affidabile nelle previsioni positive. Nel complesso, il Test 4 sembra offrire il miglior compromesso tra le diverse metriche, suggerendo che il dataset utilizzato per questo modello ha fornito il miglior addestramento per generalizzare bene sui dati di validazione.

Tuttavia si nota un comportamento diverso andando ad analizzare le metriche relative ai test set. Ricordiamo che, a differenza del training e del validation set, qui il test set è lo stesso per tutti i modelli, consentendo un confronto diretto delle loro capacità di generalizzare su dati completamente nuovi.

#### TEST SET

Metriche	Test 1	Test 2	Test 3	Test 4
Accuracy	0.9352	0.8778	0.8872	0.8872
<i>TSS</i>	0.4640	0.7597	0.7095	0.7462
Recall	0.5140	0.8820	0.8200	0.8580
Specificity	0.9500	0.8777	0.8895	0.8882
Precision	0.2641	0.2011	0.2057	0.2112
Balanced Accuracy	0.7320	0.8798	0.8547	0.8731
F1 score	0.3489	0.3275	0.3289	0.3390
CSI	0.2113	0.1958	0.1968	0.2041
HSS	0.3186	0.2884	0.2907	0.3012

Tabella 4.44: Confronto delle metriche sul test set

- Per l'accuracy si osservano di nuovo valori elevati per tutti i test, con la migliore performance ottenuta, come per il training e validation, nel Test 1 (0.9352), mentre gli altri test si assestano intorno a 0.88, leggermente più bassi. Nonostante tutti i modelli abbiano buone capacità di

classificazione generale, il Test 1 risulta leggermente migliore in termini di previsione complessiva.

- Il *TSS* varia tra il primo e i successivi test. Il Test 2 raggiunge il punteggio più alto (0.7597), seguito dal Test 4 (0.7462). Questo indica che i modelli addestrati sui dataset del Test 2 e del Test 4 riescono a separare meglio le classi rispetto al Test 1 (0.4640), il quale, nonostante la buona accuracy, presenta più difficoltà a gestire correttamente entrambe le classi.
- Come per il *TSS*, anche la recall mostra un miglioramento significativo nei Test 2 e Test 4, con valori di 0.8820 e 0.8580 rispettivamente. Il Test 1, invece, ha il valore più basso (0.5140), lasciando un numero più alto di falsi negativi.
- Per la specificity, si ottiene come negli studi precedenti il miglior risultato nel Test 1 (0.9500), mentre i Test 2, 3 e 4 hanno performance simili, leggermente inferiori ma comunque solide. Ne deduciamo che il Test 1 eccelle nell'identificazione dei negativi anche per nuovi dataset, presentando più difficoltà con i positivi, come indicato dal basso recall.
- La precision è piuttosto bassa in tutti i test, con il valore più alto nel Test 1 (0.2641), seguito dal Test 4 (0.2112), da cui deduciamo che tutti i modelli tendono a generare un numero significativo di falsi positivi.
- La Balanced Accuracy, combinando recall e specificity, fornisce una visione più completa della performance del modello su entrambe le classi. Osserviamo che il Test 2 ha il valore più alto (0.8798), seguito dal Test 4 (0.8731), mentre il primo, nonostante la buona accuracy, presenta il valore più basso (0.7320), a causa del basso recall.
- I valori dell'*F1-score* sono tutti molto simili, suggerendo che non esiste un modello che eccelle particolarmente nell'equilibrio tra precision e recall.
- Anche per il *CSI* si osservano valori molto simili, con il Test 1 che presenta il valore più alto (0.2113), seguito dal Test 4 (0.2041), confermando la difficoltà complessiva dei modelli nel fare previsioni positive accurate.
- Per l'*HSS* notiamo il Test 1 ottenere il valore più alto (0.3186), anche se tutti i test mostrano valori relativamente vicini. Questo suggerisce che il

Test 1 riesce a comportarsi leggermente meglio rispetto agli altri modelli nella classificazione generale, ma non in maniera nettamente superiore.

Analizzando i risultati ottenuti, osserviamo che il modello addestrato sul Test 4, che presentava ottimi risultati sia sul training che sul validation set, presenta un calo delle performance quando si passa al test set. Questo comportamento potrebbe essere indicativo di un overfitting ai dati di addestramento, ottenendo metriche molto elevate su training e validation, non ottenibili su dati realmente nuovi. Una spiegazione a questo comportamento potrebbe derivare dal modo in cui è stato utilizzato il modello sui diversi Test: il modello costruito, infatti, presenta un elevato numero di parametri, che non porta ad overfitting nel caso del Test 1 e Test 3, in cui il numero di campioni del dataset è molto alto (69189 e 70973), mentre porta a specializzazione sul training nel caso in cui è stato ridotto il numero di campioni negativi, ovvero nel Test 2 e Test 4. In effetti, i due dataset relativi a questi Test presentano un numero molto inferiori di campioni nel training (23600 per il Test 2 e 25384 per il Test 4). Per la teoria vista nella sezione 2.4, potrebbe essere interessante costruire un modello con un numero di parametri ancora più elevato, in modo che il modello possa entrare nel regime finale della double-descent curve.

### 4.3 Conclusioni

Nel complesso, il modello costruito basato sul Transformer ha dimostrato prestazioni abbastanza soddisfacenti nel contesto multiclasse, in cui bisogna segnalare che non sono state applicate tecniche di bilanciamento dei dati, le quali avrebbero potuto fornire al modello più campioni delle classi minoritarie da cui imparare. Se consideriamo l'articolo da cui è nata questa tesi [2], gli autori hanno applicato il modello a classi completamente bilanciate (4 classi  $Q$ ,  $C$ ,  $M$  e  $X$  ciascuna rappresentante il 25% del totale) e hanno ottenuto buoni risultati. Nel loro studio, si è raggiunto un valore di F1-score macro di 0.8275, contro il valore massimo da noi ottenuto di 0.6252. In questo lavoro di Tesi, si è voluto esplorare come le performance di apprendimento e di generalizzazione cambino modificando principalmente il bilanciamento del training set. Tuttavia, tali modifiche sono state effettuate mantenendo delle percentuali coerenti con il contesto operativo, nel quale il problema rientra nella categoria dei rare-event. L'obiettivo è stato comprendere l'impatto che un diverso bilanciamento dei dati di addestramento può avere su modelli che si trovano ad affrontare

eventi rari, senza compromettere la rappresentatività della situazione reale. Ne deduciamo, comunque, che il Transformer ha un grande potenziale nel riconoscimento di pattern e dipendenze nelle serie temporali anche nel caso di dataset fortemente sbilanciati.

Nel caso binario, invece grazie all'utilizzo di una funzione di perdita che cerca di massimizzare lo sbilanciamento tra le classi, i risultati ottenuti sono migliori rispetto al caso multiclasse. Il modello riesce a discriminare bene la classe negativa e quella positiva, nonostante la bassa rappresentazione di quest'ultima. Nell'articolo sopra citato, gli autori hanno applicato nuovamente il modello ad un dataset completamente bilanciato, con la label negativa rappresentate le classi  $Q, B, C$  al 50% e quella positiva rappresentate le classi  $M, X$  (sempre al 50%). In questo caso hanno ottenuto un  $TSS$  sullo 0.83, di poco superiore al valore massimo ottenuto dal nostro studio, che è di 0.7597

Ulteriori studi potrebbero portare a considerare lo stesso modello con diversi numeri di parametri, aumentando i blocchi dell'attenzione o utilizzando una rete di classificazione finale più grande. Inoltre, data la grande quantità di dati, il numero di epoche durante i Test è stato fissato non troppo alto, ma in ulteriori ricerche si potrebbe addestrare il modello per un numero maggiore di epoche. Il teorema di convergenza del transformer, inoltre, garantisce la convergenza del modello in un regime di over parametrizzazione, ma si ricorda che tale teorema è dimostrato solo per il caso di regressione in cui viene utilizzata la least squared loss function. Potrebbe dunque essere interessante approfondire lo studio di tale teorema anche per problemi di classificazione binaria e multiclasse utilizzando sia funzioni di loss classiche come la cross-entropy o funzioni innovative come la Score-Oriented loss function.

# Bibliografia

- [1] Charu C Aggarwal et al. *Neural networks and deep learning*, volume 10. Springer, 2018.
- [2] Khaznah Alshammari, Kartik Saini, Shah Muhammad Hamdi, and Soukaina Filali Boubrahimi. End-to-end attention/transformer model for solar flare prediction from multivariate time series data. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 558–565. IEEE, 2023.
- [3] Rafal A Angryk, Petrus C Martens, Berkay Aydin, Dustin Kempton, Sushant S Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, et al. Multivariate time series dataset for space weather data analytics. *Scientific data*, 7(1):227, 2020.
- [4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [6] Ovidiu Calin. *Deep learning architectures*. Springer, 2020.
- [7] National Research Council, Division on Engineering, Physical Sciences, Space Studies Board, Committee on the Societal, Economic Impacts of Severe Space Weather Events, and A Workshop. *Severe space weather events: Understanding societal and economic impacts: A workshop report*. National Academies Press, 2009.
- [8] Stéphane d’Ascoli. *Overparametrization, architecture and dynamics of deep neural networks: from theory to practice*. PhD thesis, Université Paris sciences et lettres, 2022.

- [9] Benjamin L. Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms, 2022.
- [10] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview, 2020.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [13] Chuanfeng Liu, Darong Liu, and Lin Mu. Improved transformer model for enhanced monthly streamflow predictions of the yangtze river. *IEEE Access*, 10:58240–58253, 2022.
- [14] F. Marchetti, S. Guastavino, M. Piana, and C. Campi. Score-oriented loss (sol) functions. *Pattern Recognition*, 132:108913, 2022.
- [15] Quynh Nguyen and Marco Mondelli. Global convergence of deep networks with one wide layer followed by pyramidal topology, 2020.
- [16] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.
- [17] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. B2t connection: Serving stability and performance in deep transformers. *arXiv preprint arXiv:2206.00330*, 2022.
- [18] John Thickstun. The transformer model in equations. *University of Washington: Seattle, WA, USA*, 2021.
- [19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [20] Ž Vujović et al. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, 12(6):599–606, 2021.

- [21] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.
- [22] Yongtao Wu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher. On the convergence of encoder-only shallow transformers. *Advances in Neural Information Processing Systems*, 36, 2024.
- [23] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.
- [24] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019.



# Ringraziamenti

In quest'ultima parte del mio lavoro, vorrei provare a ringraziare tutte le persone che hanno avuto un ruolo fondamentale nella mia crescita personale e professionale, accompagnandomi lungo questo percorso.

In primo luogo, un sentito ringraziamento alla mia relatrice, Sabrina, che con pazienza e dedizione mi ha guidata in questo bellissimo viaggio accademico, aiutandomi a trovare la mia strada e a crescere all'interno di un mondo che ho imparato ad amare.

Un pensiero speciale va alla mia famiglia, per tutti i momenti condivisi che mi hanno fatto sentire parte di qualcosa di meraviglioso. Ogni momento trascorso insieme, ogni insegnamento e ogni valore che mi hanno trasmesso hanno contribuito a farmi diventare la persona che sono.

Grazie ai miei genitori, che mi hanno accompagnata con tanto amore e dedizione, aiutandomi a crescere e maturare. A mia madre, la roccia della mia vita: grazie per la tua forza incredibile, per essere sempre stata pronta a rialzarmi quando cadevo e per aver sempre creduto in me anche quando non riuscivo a farlo io stessa. A mio padre, che è la forza motrice dietro molte delle mie scelte, che mi sprona nei momenti di incertezza e che mi ha spinto a intraprendere e credere in questo percorso che ha cambiato la mia vita.

Un ringraziamento speciale va ai miei nonni. A mio nonno Sergio, che purtroppo è venuto a mancare all'inizio di questo cammino, ma che sono felice abbia potuto vedermi felice nelle mie scelte. A mia nonna Giovanna, che mi è sempre stata vicina e mi ha insegnato ad affrontare con coraggio le sfide che la vita ci pone davanti.

Un pensiero speciale a Matteo, la persona che più mi sopporta da anni, che ha condiviso con me i momenti più importanti della mia vita, che mi ha aiutata a gestire e superare tutti i momenti di crisi, e mi spinge ad essere la migliore versione di me stessa. Ti sono grata per come mi sostieni nelle scelte

della vita e per mettermi sempre al centro delle tue attenzioni.

Grazie anche alle mie cugine, Marta e Dodi, le sorelle che avrei sempre voluto e che si sono rivelate tali, che mi aiutano a sorridere e a vivere la vita con gioia e leggerezza. Vi ringrazio perchè siete sempre state al mio fianco e so che ci sarete sempre.

Un pensiero va a tutte le amiche che ho avuto nella vita, che mi hanno accompagnata in modi diversi lungo questo cammino. Grazie per il supporto, l'affetto e la capacità di rendere ogni momento più leggero e piacevole.

In particolare, desidero ringraziare le mie amiche di sempre: Alice, Irene e Rebecca. Grazie per tutti i momenti felici e spensierati trascorsi insieme, per i consigli preziosi e per essere sempre presenti, pronte a condividere gioie e difficoltà.

Infine, un ringraziamento sincero alle nuove amicizie nate durante il periodo universitario, che hanno arricchito il mio percorso. In particolare, un grazie di cuore a Miriana, compagna di mille avventure, con cui ho condiviso gioie, fatiche ed esami, rendendo questo viaggio ancora più speciale.