UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

Modelli di sequenze aleatore basati sul Transformer

Relatore: Prof. Bilardi Gianfranco

Laureando: Sandron Piergiorgio

Correlatore: Prof. Satta Giorgio

Anno Accademico 2023-2024

Data di laurea 25/09/2024

Abstract

Negli ultimi anni i Transformer basati sull'Attenzione hanno ottenuto un enorme successo in una varietà di discipline. L'obiettivo di questo progetto è studiare la capacità dei Transformer di modellare sequenze aleatorie di dati. Si inizierà descrivendo l'architettura interna di un Transformer e la relativa fase di allenamento, successivamente si eseguiranno dei test variando particolari parametri e si mostreranno nel dettaglio le principali operazioni eseguite dal Transformer. A scopo didattico verrà mostrata anche la principale parte del codice sorgente e, infine, verranno tratte le conclusioni dello studio.

1 Introduzione

I Transformer basati sull'attenzione sono stati all'apice di recenti scoperte in diverse discipline, tra cui le lingue naturali, questo perché dato un testo di n parole (generalmente chiamate token) sono capaci di prevedere la parola $n+1$, iterando questo procedimento sono quindi in grado di simulare il linguaggio umano in maniera sorprendente.
Oltre alla loro architettura innovativa, uno degli ingredienti chiave del loro successo è la procedura di allenamento. Sfruttando l'efficiente elaborazione di dati sequenziali, questi modelli sono stati preaddestrati su grandi gruppi di testi con l'obiettivo di prevedere le parole successive (NTP, Next Token Prediction). Tali modelli addestrati sono generalizzabili a un ampio assortimento di compiti del linguaggio naturale.
Dato il successo riscontrato c'è un enorme interesse e una ricerca attiva nella comprensione del loro funzionamento. Nonostante i sorprendenti risultati già ottenuti, però, non è ancora stata compresa una caratterizzazione precisa riguardo l'obbiettivo NTP e sulle capacità di apprendimento dei Transformer.
Lo scopo di questo testo consiste nell'usare sequenze aleatorie di parole statisticamente indipendenti fra loro, secondo una data distribuzione di probabilità, all'ingresso di un Transformer e verificare se quest'ultimo è in grado di individuare tale distribuzione.
Tutto lo studio è stato condotto usando l'ambiente di sviluppo MATLAB, ma al fine di presentare i dati in maniera più chiara e generica, soprattutto in termini teorici, si è data particolare priorità alla riscrittura di alcune formule, mantenendo ovviamente invariati i risultati ottenuti.

Notazione e definizioni. Gli scalari verranno indicati con le lettere corsive minuscole x, i vettori euclidei verranno rappresentati con lettere minuscole in grassetto $\mathbf{x}$ e le matrici in grassetto maiuscolo $\mathbf{X}$.
Una sequenza ordinata di valori di lunghezza n verrà indicata con $\mathbf{x}^n \triangleq (x_1, \dots, x_n)$.
Viene definito vocabolario $\mathbb{V}$ di taglia v ogni insieme $\mathbb{V} = \{a_1, \dots, a_v\}$ di parole differenti $a_i \neq a_j \ \forall\, i \neq j$ con $i, j = 1, \dots, v$, ma per ragioni pratiche ci si riferirà a esse secondo il loro indice x, ovvero $x = i \iff a = a_i$ e, con abuso di notazione, ci riferirà al vocabolario come $\mathbb{V} = \{x_1, \dots, x_v\} = \{1, \dots, v\}$ e a una sequenza ordinata di n parole con $\mathbf{x}^n \triangleq (x_1, \dots, x_n) \in \mathbb{V}^n$.

2 Descrizione del Transformer

In questo capitolo verrà spiegata la struttura e il funzionamento di un Transformer.
È stato scelto di ricreare un modello basilare, ma dotato di tutte le fasi principali, per permettere una spiegazione dettagliata ed esaustiva, senza trascurare alcun aspetto importante.

2.1 Architettura del Transformer

Sia $\mathbf{x}^n$ una sequenza di parole statisticamente indipendenti generate da una distribuzione di probabilità $\mathbb{P}$ (concretamente vettore $\mathbf{p}$) all'ingresso del Transformer, nella pratica definita come vettore colonna $\mathbf{x} \in \mathbb{V}^n$.
Lo scopo del Transformer è quello di generare in uscita un vettore $\mathbf{log\_probs} \in \mathbb{R}^v$, in questo testo anche chiamato $\mathbf{q}$. Gli elementi di tale vettore rappresenteranno la probabilità che la parola da prevedere sia quella a essi associata: $\mathrm{P}[x_{n+1} = a_i] = q_i$ per ogni $i = 1, \ldots, v$.
Più tale vettore sarà uguale al vettore $\mathbf{p}$ più precisa sarà la quindi la capacità di previsione del Transformer.
In seguito sono riportate le fasi e i parametri principali che compongono la struttura del Transformer, spiegati successivamente nel dettaglio:

$$\begin{aligned}
\mathbf{x}_i &= \mathbf{e}_{x_i} + \mathbf{p}_i && Embedding \\
\mathbf{y}_i &= \mathbf{x}_i + \mathbf{a}_i\mathbf{W}_O + \mathbf{w}_o && Self - Attention \\
\mathbf{z}_i &= \mathbf{y}_i + ReLU(\mathbf{y}_i\mathbf{W}_1 + \mathbf{w}_1)\mathbf{W}_2 + \mathbf{w}_2 && Feed - Forward \\
\mathbf{logit} &= \mathbf{z}_n\mathbf{W}_3 + \mathbf{w}_3 && Linear \\
\mathbf{q} = \mathbf{log_probs} &= softmax(\mathbf{logit}) && Softmax
\end{aligned}$$

i **Word-Embedding:** Ogni parola della sequenza $\mathbf{x}^n$ viene mappata secondo la matrice di Embedding $\mathbf{E} \in \mathbb{R}^{(v+1)\times d}$ generando così una matrice di dimensioni $\mathbf{X} \in \mathbb{R}^{n\times d}$. Ovvero la i-esima parola della sequenza $x_i \in \mathbb{V}$ verrà mappata con la riga corrispondente al valore x_i della matrice $\mathbf{E}$ ($\mathbf{e}_{x_i}$), questo per ogni $i = 1, \ldots, n$. L'utima riga della matrice di Embedding serve per mappare eventuali parole non previste nel vocabolario.
Lo scopo di questa fase è rappresentare ogni parola in un vettore, attribuendo a parole usate ricorrentemente negli stessi contesti vettori con valori simili e viceversa. Per tutte le successive fasi sarà quindi sottinteso che ogni riga della matrice appena ottenuta rappresenta la corrispettiva parola nella sequenza di ingresso.

$$\mathbf{E} = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,d} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ e_{v+1,1} & e_{v+1,2} & \cdots & e_{v+1,d} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_{v+1} \end{pmatrix}$$

ii **Positional-Embedding:** Lo scopo della fase di Positional-Embedding è tenere traccia dell'ordine delle parole nella sequenza in entrata, la matrice $\mathbf{P} \in \mathbb{R}^{n\times d}$ viene quindi sommata alla matrice $\mathbf{X}$ ottenuta nella fase di Word-Embedding.

Questa operazione permette a ogni vettore riga ottenuto nella fase di Word-Embedding di essere pesato in maniera differente a seconda della sua posizione nella sequenza di ingresso. Idealmente questo passaggio viene fatto perché parole uguali, ma in posizioni diverse, possono avere significati e pesi differenti, come avviene nel linguaggio naturale.
Alla fine di questa fase ogni riga della matrice $\mathbf{X}$ è quindi definita dalla seguente formula: $\mathbf{x}_i = \mathbf{e}_{x_i} + \mathbf{p}_i$ con $i = 1, \ldots, n$.
nota. Nel contesto di questo studio, essendo ogni parola statisticamente indipendente dalle altre, questa fase risulta superflua. È stato scelto di inserirla al fine di comprendere se il modello classico del Transformer è in grado di notare anche l'assenza di tale schema.

iii Self-Attention: In questa fase ogni parola viene messa in relazione con le altre (compresa sé stessa) al fine di consentire al modello di identificare e soppesare l'importanza delle varie parti della sequenza di input.
Il **primo passaggio** richiede di calcolare i seguenti vettori per ogni parola $\mathbf{x}_i$ ottenuta nei passaggi precedenti, con $i = 1, \ldots, n$.

$$\begin{aligned} \mathbf{q}_i &= \mathbf{x}_i \mathbf{W}_Q + \mathbf{w}_q \in \mathbb{R}^m && Query \\ \mathbf{k}_i &= \mathbf{x}_i \mathbf{W}_K + \mathbf{w}_k \in \mathbb{R}^m && Key \\ \mathbf{v}_i &= \mathbf{x}_i \mathbf{W}_V + \mathbf{w}_v \in \mathbb{R}^m && Value \end{aligned}$$

Il **secondo passaggio** comprende l'assegnazione di un punteggio per ogni parola $\mathbf{x}_i$ rispetto a tutte le altre, tale punteggio determina in pratica quanta attenzione porre sulle varie parti della sequenza a seconda della parola considerata. Il calcolo per i punteggi attribuiti alla parola i-esima viene effettuato moltiplicando il vettore Query $\mathbf{q}_i$ per tutti i vettori $\mathbf{k}$, definendo $score_{i,j} = \mathbf{q}_i \cdot \mathbf{k}_j$ il punteggio della parola i in relazione alla parola j. Si anticipa che l'unione di tutti i punteggi genererà una matrice $\mathbf{K}\mathbf{Q}^T \in \mathbb{R}^{n \times n}$.
Il **terzo** e **quarto passaggio** consistono rispettivamente nel dividere ogni punteggio per la radice della dimensione dei vettori di Embedding d (valore standard allo scopo di ottenere gradienti più stabili) ed effettuare un'operazione di softmax per tutti i punteggi inerenti a ciascuna parola $\begin{pmatrix} score_{i,1} & score_{i,2} & \ldots & score_{i,n} \end{pmatrix}$ con $i = 1, \ldots n$. Alla fine di questa fase a ogni vettore di Embedding $\mathbf{x}_i$ risultano quindi attribuiti solo punteggi positivi la cui somma è pari a 1.
Il **quinto passaggio** richiede di moltiplicare tutti i punteggi softmax di ogni parola $\mathbf{x}_i$ per la corrispettiva componente di ogni vettore $\mathbf{v}$, ovvero: $score_{i,j} \cdot v_{j,i}$ con $i, j \in 1, \ldots, n$.
Intuitivamente si vogliono mantenere intatti i valori sui quali concentrarsi ed escludere quelli dove la relazione fra due parole è irrilevante.
Il **sesto passaggio** consiste nel sommare i valori ponderati dai vari punteggi secondo la formula: $a_{i,j} = \sum_k^n score_{i,k} \cdot v_{k,j}$, con $i = 1, \ldots, n$ e $j = 1, \ldots, m$. Questo produce l'output dell'Autoattenzione.
Nell'implementazione pratica ci si può ricondurre al calcolo matriciale condensando i precedenti cinque passaggi in un unica formula

$$\mathbf{A} = softmax\left(\frac{\mathbf{K}\mathbf{Q}^T}{\sqrt{d}}\right)^T \mathbf{V} \in \mathbb{R}^{n \times m}$$

Le matrici $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times m}$ sono state ottenute unendo i vettori riga $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^m$ ottenuti nel primo passaggio (rispettando lo stesso ordine i).
Infine la matrice $\mathbf{W}_O$ e il corrispettivo vettore di bias $\mathbf{w}_o$ servono per ridimensionare la matrice $\mathbf{A}$ portandola alle stesse dimensioni della matrice di ingresso $\mathbf{X} \in \mathbb{R}^{n \times d}$.
All'uscita della fase di Autoattenzione la matrice di output risultante verrà sommata alla stessa matrice $\mathbf{X}$ precedentemente posta in ingresso ottenendo così la nuova matrice $\mathbf{Y} \in \mathbb{R}^{n \times d}$. Ogni riga di tale matrice sarà quindi ottenuta secondo la formula $\mathbf{y}_i = \mathbf{x}_i + \mathbf{a}_i \mathbf{W}_O + \mathbf{w}_o$, con $i = 1, \ldots, n$.
In questo testo si è scelto di creare un modello a testa singola (single-head attention) così da ottenere le 3 matrici $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ e i 3 vettori di bias $\mathbf{w}_q, \mathbf{w}_k, \mathbf{w}_v$, se si fosse adottato un modello a N teste (multi-head attention) si sarebbero ottenute 3N matrici e altrettanti vettori.
nota. Più precisamente la formula teorica è: $\mathbf{A} = softmax(\frac{\mathbf{Q}\mathbf{T}^T}{\sqrt{d}})\mathbf{V} \in \mathbb{R}^{n \times m}$, ma secondo il procedimento con cui MATLAB esegue le varie operazioni e per la forma con cui si sono presentati i dati, la formula sopra riportata è quella corretta in questo contesto.

iv Normalization 1: La fase di normalizzazione permette di addestrare il Transformer in maniera più efficace trasformando i valori della matrice di ingresso su scala simile aiutando il modello ad apprendere i pesi in maniera appropriata. Senza tale fase il Transformer rischia di prestare troppa attenzione ai parametri con intervalli ampi, tralasciando quelli in intervalli ristretti.
L'operazione pratica in cui ogni valore della matrice $y \in \mathbf{Y}$, viene sottoposto è:

$$y'_{i,j} = \gamma_j \cdot \frac{y_{i,j} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta_j \quad i = 1, \ldots, n \;\; j = 1, \ldots, d$$

I valori μ e σ^2 corrispondono rispettivamente alla media e alla varianza degli elementi della matrice $\mathbf{Y}$, mentre i vettori $\boldsymbol{\gamma}$ e $\boldsymbol{\beta}$ sono chiamati rispettivamente vettori Scale e Offset, infine ϵ è una costante che migliora la stabilità numerica quando la varianza è molto piccola ($1e$-5 di default).

v Feed-Forward (FF): Trasformazione non lineare successiva alla fase di Autoattenzione al fine di consentire al Transformer di apprendere modelli più complessi nei dati.
Può anche essere vista come un modo di perfezionare le funzionalità estratte dalla fase di Autoattenzione, serve inoltre per enfatizzare caratteristiche importanti ed eliminare quelle meno rilevanti, rendendo l'output finale più informativo.
Nella pratica per ogni riga della matrice $\mathbf{y}_i \in \mathbf{Y}$ si ottengono dapprima i vettori $\mathbf{f}_i = \mathbf{y}_i \mathbf{W}_1 + \mathbf{w}_1 \in \mathbb{R}^r$ con $i = 1, \ldots, n$. Successivamente tutti i valori negativi vengono mappati in 0 (ReLU), infine si ricava la matrice $\mathbf{Z} \in \mathbb{R}^{n \times d}$ dove ogni riga è data da: $\mathbf{z}_i = \mathbf{f}_i \mathbf{W}_2 + \mathbf{w}_2 \in \mathbb{R}^d$ con $i = 1, \ldots, n$.

vi Normalization 2: Fase del tutto uguale a quella del punto **iv**, usando però nuovi vettori $\boldsymbol{\gamma}$ e $\boldsymbol{\beta}$, sempre dotati di iperparametri. La matrice in ingresso in questo caso è la matrice $\mathbf{Z} \in \mathbb{R}^{n \times d}$.

nota. Per definizione Le fasi **iii**−**vi** costituiscono un Codificatore del Transformer, generalmente essi sono posti in sequenza nei grandi modelli.

vii **Linear:** Viene presa l'ultima riga della matrice $\mathbf{Z}$, $\mathbf{z}_n \in \mathbb{R}^d$ corrispondente alla previsione della parola $n+1$, viene applicata una trasformazione lineare, tramite la matrice $\mathbf{W}_3$ e il vettore $\mathbf{w}_3$ ottenendo così il vettore $\mathbf{logit} = \mathbf{z}_n\mathbf{W}_3 + \mathbf{w}_3 \in \mathbb{R}^v$ con lunghezza uguale al numero di parole del vocabolario.

viii **Softmax:** Viene effettuata l'operazione di softmax del vettore **logit** ottenendo il vettore di probabilità $\mathbf{q} = \mathbf{log\_probs} = softmax(\mathbf{logit}) \in \mathbb{R}^v$ descritto all'inizio del paragrafo.
Se il Transformer ha imparato correttamente, il vettore $\mathbf{q}$ dovrebbe coincidere con la distribuzione di probabilità $\mathbb{P}$, nella pratica dovrebbe essere simile al vettore $\mathbf{p}$ che ha generato la sequenza di ingresso.

Nella Tabella 1 sono riassunti tutti i parametri interni del Transformer, come si può notare il loro numero totale può essere riassunto maggiormente da 3 variabili: d, m, r. Esse sono le dimensioni di riferimento rispettivamente nella fase di Embedding, Self-Attention e FF, vengono stabilite in fase di costruzione. Tutti gli elementi delle matrici o vettori presenti nella Tabella 1 vengono chiamati iperparametri o parametri di autoapprendimento poichè in fase di allenamento vengono adattati a seconda del tipo di problema.

2.2 Allenamento del Transformer

Inizialmente i parametri di autoapprendimento del Transformer hanno valori predefiniti o casuali, è necessaria quindi una fase di allenamento per permettergli di adattarsi al tipo di incarico previsto.
Per compiti NTP la funzione di perdita usata è l'entropia incrociata (crossentropy), essa viene calcolata secondo la formula:

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{i}^{v} p_i \cdot \log q_i$$

dove $\mathbf{p}$ e $\mathbf{q}$ sono due vettori di probabilità, si noti che la definizione non è simmetrica, $\mathbf{p}$ in questo caso rappresenta la distribuzione teorica ideale, mentre $\mathbf{q}$ è intesa come la distribuzione "non naturale" ottenuta dal Transformer.
In fase di allenamento si sono usati come campioni un numero N di sequenze $\mathbf{x}^n$ e un altrettanto numero di etichette $\mathbf{y}$ corrispondenti alle parole da prevedere, entrambi generati casualmente secondo la distribuzione di probabilità $\mathbb{P}$.
Essendo un problema di predizione e classificazione singola (prediction and single-label classification tasks) tali etichette sono rappresentate dai vettori della base canonica $\mathbf{e}_i \in \mathbb{R}^v$, questo perché è stato supposto che la parola corretta da prevedere possa essere soltanto una.
A scopo illustrativo, data una sequenza di allenamento generica $\mathbf{x}^n$ e la corrispettiva etichetta $\mathbf{y}$ allora $\mathbf{y} = \mathbf{e}_i \iff x_{n+1} = i$, con $i \in \{1, \ldots, v\}$.

Tabella 1: **Iperparametri del Transformer e forma delle matrici**

Layer	**Parametri**	**Dimensione**
W. Embedding	Matrice $\mathbf{E}$	$(v+1) \times d$
P. Embedding	Matrice $\mathbf{P}$	$n \times d$
Self-Attention	Matrice Query $\mathbf{W}_Q$	$d \times m$
Self-Attention	Matrice Key $\mathbf{W}_K$	$d \times m$
Self-Attention	Matrice Value $\mathbf{W}_V$	$d \times m$
Self-Attention	Matrice Output $\mathbf{W}_O$	$m \times d$
Self-Attention	Vettore Bias $\mathbf{w}_q$	$1 \times m$
Self-Attention	Vettore Bias $\mathbf{w}_k$	$1 \times m$
Self-Attention	Vettore Bias $\mathbf{w}_v$	$1 \times m$
Self-Attention	Vettore Bias $\mathbf{w}_o$	$1 \times d$
Normalization 1	Vettore Scale $\boldsymbol{\gamma}$	$1 \times d$
Normalization 1	Vettore Offset $\boldsymbol{\beta}$	$1 \times d$
Feed-Forward	Matrice $\mathbf{W}_1$	$d \times r$
Feed-Forward	Vettore Bias $\mathbf{w}_1$	$1 \times r$
Feed-Forward	Matrice $\mathbf{W}_2$	$r \times d$
Feed-Forward	Vettore Bias $\mathbf{w}_2$	$1 \times d$
Normalization 2	Vettore Scale $\boldsymbol{\gamma}$	$1 \times d$
Normalization 2	Vettore Offset $\boldsymbol{\beta}$	$1 \times d$
Linear	Matrice $\mathbf{W}_3$	$d \times v$
Linear	Vettore Bias $\mathbf{w}_3$	$1 \times v$

Dato il vettore $\mathbf{q}$ in uscita dal Transformer e la corrispettiva etichetta $\mathbf{y} = \mathbf{e}_i$ l'effettiva funzione di perdita calcolata a ogni iterazione è data da:

$$loss_i = -\sum_{j=1}^{v} \left(e_{i,j} \cdot \log q_j \right) = -\log q_i$$

Per verificare l'accuratezza con la quale vengono aggiornati gli iperparametri in fase di allenamento si effettua una media aritmetica delle funzioni di perdita $loss_i$, l'aspettativa è che tale media converga verso il valore teorico $H(\mathbf{p}, \mathbf{p})$ chiamato anche *mean loss* (ML).
A ogni ciclo saranno eseguite quindi N iterazioni, una per ogni sequenza aleatoria di allenamento $\mathbf{x}^n$, prima di ogni nuovo ciclo le coppie campione-etichetta verranno permutate così da evitare che il Transformer si adatti eccessivamente ai campioni di test (fenomeno dell'overfitting).
La fase di allenamento si conclude dopo un numero prestabilito di cicli (Epoche).
Nella Tabella 2 sono riassunti i vari parametri usati in fase di allenamento, il valore di

Batch rappresenta il numero di campioni raggruppati, nel contesto pratico, per eseguire le varie operazioni in parallelo al fine di ottimizzare i tempi di allenamento.

Tabella 2: **Parametri di allenamento**

Ottimizzatore	Adam ($\beta_1 = 0.9, \beta_2 = 0.95$)
Tasso Apprendimento	0.001
Iterazioni	8000
Batch Size	16
Epoche	3
Lunghezza Sequenze	$n \in \{16, 32, 64, 128\}$
Dimensione Embedding	$d \in \{8, 16, 32, 64, 128\}$
Dimensione Autoattenzione	$m \in \{4, 8, 16, 32, 64, 128\}$
Teste dell'Autoattenzione	1
Dimensione FF	$r \in \{16, 32, 64, 128, 256, 512\}$

3 Studio del Transformer

3.1 Risultati empirici

Lo scopo dello studio consiste nel creare e allenare vari Transformer con parametri diversi, osservare il loro funzionamento e la loro struttura interna, confrontare i vettori di output $\mathbf{q}$ con i rispettivi vettori di probabilità $\mathbf{p}$.
Sono state considerate sequenze di lunghezza $n \in \{16, 32, 64, 128\}$ con un numero di parole differenti $v \in \{2, 4, 8\}$ generate da una distribuzione di probabilità $\mathbb{P}$ diversa solo per ogni taglia v del vocabolario. Mantenere la stessa distribuzione $\mathbb{P}$ per sequenze originate dallo stesso vocabolario permette di avere un paragone oggettivo per confrontare i vari risultati.
I parametri strutturali del Transformer sono la dimensione dei vettori di Embedding e delle matrici (coi rispettivi vettori bias) relative alla fase di Autoattenzione, rispettivamente $d, m \in \{4, 8, 16, 32, 64, 128\}$. La dimensione relativa al FF r è stata calcolata moltiplicando per 4 o per 2 il corrispondente valore d.
Essendo le sequenze di ingresso generate da una sorgente senza memoria, una delle aspettative principali è quella che a ogni iterazione il Transformer ideale generi in uscita lo stesso vettore di probabilità $\mathbf{q}$. I risultati dei test hanno dimostrato che ogni Transformer allenato segue questo comportamento.
Le probabilità riportate nelle Tabelle 3, 4 e 5 si possono quindi interpretare come il vettore $\mathbf{q}$ medio in uscita per qualsiasi sequenza in entrata al relativo Transformer. È stata calcolata anche l'entropia incrociata $H(\mathbf{p}, \mathbf{q})$ come criterio di confronto col valore teorico ideale $H(\mathbf{p}, \mathbf{p})$ (uguale alla ML). Sono stati inoltre trascritti il numero totale di iperparametri (Learnables) del Transformer e il massimo scostamento dalla probabilità teorica definito secondo la seguente formula $err = \max |p_i - q_i| \ \forall \ i = 1, \ldots, v$.

Tabella 3: **Test relativi a sequenze generate da un vocabolario di 2 parole**

% Prob Teorica		ML	Parametri				Learn	% Err
75.00	**25.00**	**0.56234**	**n**	**d**	**m**	**r**		
78.95	21.05	0.56682	16	8	4	16	630	3.95
74.88	25.12	0.56234	16	8	8	16	770	0.12
74.09	25.91	0.56255	16	8	4	32	902	0.91
74.86	25.14	0.56234	16	8	8	32	1042	0.14
71.93	28.07	0.56472	16	16	8	64	3082	3.07
75.30	24.70	0.56236	16	16	16	64	3618	0.3
75.71	24.29	0.56247	32	8	4	16	758	0.71
76.87	23.13	0.56330	32	16	8	32	2282	1.87
72.97	27.03	0.56340	32	32	16	64	7634	2.03
72.66	27.34	0.56374	32	32	16	128	11794	2.34
70.68	29.32	0.56698	32	32	32	128	13890	4.32
73.28	26.72	0.56310	64	8	4	16	1014	1.72
71.97	28.03	0.56466	64	16	8	32	2794	3.03
76.70	23.30	0.56313	64	32	16	64	8658	1.7
72.58	27.42	0.56383	64	64	16	256	41970	2.42
77.43	22.57	0.56399	64	64	32	256	46114	2.43
75.71	24.29	0.56247	64	64	64	256	54402	0.71
73.46	26.54	0.56295	128	8	4	16	1526	1.54
77.56	22.44	0.56418	128	16	8	32	3818	2.56
73.90	26.10	0.56265	128	32	16	64	10706	1.1
69.95	30.05	0.56861	128	64	32	128	33698	5.05
72.65	27.35	0.56375	128	64	32	256	50210	2.35
74.13	25.87	0.56254	128	128	64	512	182338	0.87
68.17	31.83	0.57358	128	128	128	512	215298	6.83

3.2 Esempio pratico

In questa sezione sono riportati i passaggi più significativi eseguiti dal Transformer con lo scopo di evidenziare maggiormente il suo funzionamento quando all'ingresso sono presenti sequenze aleatorie con parole statisticamente indipendenti fra loro.
I parametri stabiliti in fase di costruzione sono:

- Vettore di probabilità $\mathbf{p} = (1/2, 1/4, 1/8, 1/8)$
- Lunghezza delle sequenze aleatorie $n = 8$
- Dimensione dei vettori di Embedding $d = 4$
- Dimensione relativa all'Autoattenzione $m = 4$
- Dimensione relativa al FF $r = 16$

Il numero totale di iperparametri del Transformer è 316.
Gli ulteriori parametri impostati in fase di allenamento sono gli stessi presenti nella Tabella 1.
Il valore ML teorico è 1.213 mentre la media relativa all'ultimo 10% delle *loss* calcolate

Tabella 4: **Test relativi a sequenze generate da un vocabolario di 4 parole**

% Prob Teorica				ML	Parametri				Learn	% Err
50.00	**25.00**	**12.50**	**12.50**	**1.213**	**n**	**d**	**m**	**r**		
51.90	24.36	12.47	11.27	1.2141	16	8	4	16	664	1.9
49.64	23.57	13.18	13.61	1.2141	16	8	8	16	804	1.43
50.54	24.08	11.38	14.00	1.2146	16	8	4	32	936	1.5
46.61	27.72	12.24	13.43	1.2160	16	8	8	32	1076	3.39
48.89	25.69	12.00	13.42	1.2137	16	16	8	64	3148	1.11
51.60	25.80	12.47	10.13	1.2160	16	16	16	64	3684	2.37
44.75	29.78	13.53	11.94	1.2206	32	8	4	16	792	5.25
52.75	22.80	12.55	11.90	1.2149	32	16	8	32	2348	2.75
49.65	24.63	10.11	15.61	1.2190	32	32	16	64	7764	3.11
53.43	21.16	13.98	11.43	1.2187	32	32	16	128	11924	3.84
47.89	24.34	14.14	13.63	1.2150	32	32	32	128	14020	2.11
51.00	24.39	11.92	12.69	1.2133	64	8	4	16	1048	1
48.34	25.60	14.76	11.30	1.2158	64	16	8	32	2860	2.26
47.03	30.91	11.38	10.68	1.2220	64	32	16	64	8788	5.91
49.17	24.73	13.63	12.47	1.2136	64	64	16	256	42228	1.13
48.12	20.45	16.71	14.72	1.2257	64	64	32	256	46372	4.55
59.03	21.59	11.85	7.53	1.2367	64	64	64	256	54660	9.03
45.48	28.59	13.30	12.63	1.2178	128	8	4	16	1560	4.52
51.99	22.89	11.80	13.32	1.2148	128	16	8	32	3884	2.11
51.21	24.18	11.90	12.71	1.2135	128	32	16	64	10836	1.21
38.86	31.16	13.63	16.35	1.2396	128	64	32	128	33956	11.14
54.70	19.39	12.48	13.43	1.2228	128	64	32	256	50468	5.61
50.34	27.58	8.23	13.85	1.2245	128	128	64	512	182852	4.27
49.69	24.25	12.50	13.56	1.2136	128	128	128	512	215812	1.06

in fase di allenamento è 1.2165, il grafico completo è riportato nella Figura 1.
La sequenza aleatoria all'ingresso del Transformer è $\mathbf{x}^8 = \begin{pmatrix}1 & 1 & 1 & 4 & 1 & 2 & 1 & 4\end{pmatrix}^T$.
In seguito sono stati trascritti i principali dati estratti dal Transformer secondo i parametri appena riportati.
Matrici relative alla fase di Embedding

$$\mathbf{E} = \begin{pmatrix} -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ -0.0439 & 0.0019 & 0.0153 & 0.0027 \\ 0.0030 & 0.0113 & -0.0154 & 0.0339 \\ 0.0038 & 0.0031 & -0.0153 & -0.0181 \\ -0.0139 & -0.0187 & -0.0078 & 0.0240 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} -0.0495 & -0.0153 & 0.0177 & -0.0284 \\ -0.0350 & -0.0233 & 0.0169 & -0.0440 \\ -0.0536 & -0.0186 & 0.0312 & -0.0414 \\ -0.0311 & -0.0008 & 0.0162 & -0.0391 \\ -0.0634 & -0.0272 & 0.0316 & -0.0126 \\ -0.0400 & -0.0006 & 0.0233 & -0.0230 \\ -0.0354 & -0.0165 & 0.0186 & -0.0396 \\ -0.0305 & 0.0407 & -0.0135 & 0.1786 \end{pmatrix}$$

Tabella 5: **Test relativi a sequenze generate da un vocabolario di 8 parole**

% Prob Teorica								ML	Parametri				Learn	% Err
25.00	**25.00**	**12.50**	**12.50**	**12.50**	**6.25**	**3.125**	**3.125**	**1.8628**	**n**	**d**	**m**	**r**		
24.91	24.48	14.34	12.28	12.33	5.66	3.28	2.72	1.8648	16	8	4	16	732	1.84
24.38	25.21	12.76	12.57	12.57	6.76	2.55	3.20	1.8638	16	8	8	16	872	0.62
26.92	24.21	12.86	11.48	12.28	6.49	2.81	2.95	1.8644	16	8	4	32	1004	1.92
25.12	23.22	14.99	11.65	12.07	6.56	3.46	2.93	1.8664	16	8	8	32	1144	2.49
27.04	23.69	12.89	12.14	11.62	6.17	3.28	3.17	1.8645	16	16	8	64	3280	2.04
24.92	23.73	15.45	11.38	13.72	3.84	4.29	2.67	1.8757	16	16	16	64	3816	2.95
22.27	26.47	12.77	13.37	12.81	6.03	3.42	2.86	1.8655	32	8	4	16	860	2.73
27.96	25.47	12.44	9.74	11.92	6.37	3.31	2.79	1.8685	32	16	8	32	2480	2.96
24.80	25.08	11.66	12.21	13.12	7.59	2.01	3.53	1.8674	32	32	16	64	8024	1.34
28.67	25.90	11.16	13.07	10.36	5.42	2.85	2.57	1.8697	32	32	16	128	12184	3.67
26.42	26.08	11.22	10.63	14.17	5.36	3.37	2.75	1.8678	32	32	32	128	14280	1.87
24.84	26.91	10.50	12.86	12.89	6.09	2.80	3.11	1.8656	64	8	4	16	1116	2
26.13	22.33	12.63	13.68	13.15	5.89	2.96	3.23	1.8655	64	16	8	32	2992	2.67
25.80	23.08	15.28	10.95	12.74	6.09	3.21	2.85	1.8677	64	32	16	64	9048	2.78
28.38	21.84	15.70	11.11	11.63	6.96	2.10	2.28	1.8757	64	64	16	256	42744	3.38
24.21	21.77	12.52	13.44	15.81	6.96	3.49	1.80	1.8739	64	64	32	256	46888	3.31
30.65	16.16	10.70	14.54	15.41	5.95	2.47	4.12	1.8972	64	64	64	256	55176	8.84
24.40	23.14	14.61	11.30	12.90	7.09	3.29	3.27	1.8665	128	8	4	16	1628	2.11
28.42	20.36	10.85	13.12	13.40	7.34	3.00	3.51	1.8727	128	16	8	32	4016	4.64
19.28	23.22	12.94	18.45	15.28	4.99	3.58	2.26	1.8881	128	32	16	64	11096	5.95
20.27	23.51	14.59	13.55	15.61	5.90	3.42	3.15	1.8740	128	64	32	128	34472	4.73
25.14	27.89	12.34	14.30	10.85	5.25	2.21	2.02	1.8720	128	64	32	256	50984	2.89
20.10	31.62	9.08	14.40	12.25	6.10	2.97	3.48	1.8831	128	128	64	512	183880	6.62
22.14	22.57	14.58	11.64	14.90	8.17	2.85	3.15	1.8724	128	128	128	512	216840	2.86

Matrice $\mathbf{X}$ in uscita dalla fase di Embedding

$$\mathbf{X} = \begin{pmatrix} -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ 0.0038 & 0.0031 & -0.0153 & -0.0181 \\ -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ -0.0439 & 0.0019 & 0.0153 & 0.0027 \\ -0.0410 & 0.0119 & 0.0098 & 0.0307 \\ 0.0038 & 0.0031 & -0.0153 & -0.0181 \end{pmatrix} + \mathbf{P} = \begin{pmatrix} -0.0906 & -0.0034 & 0.0275 & 0.0023 \\ -0.0760 & -0.0114 & 0.0267 & -0.0133 \\ -0.0946 & -0.0067 & 0.0410 & -0.0106 \\ -0.0274 & 0.0023 & 0.0009 & -0.0572 \\ -0.1044 & -0.0153 & 0.0414 & 0.0181 \\ -0.0838 & 0.0012 & 0.0386 & -0.0202 \\ -0.0764 & -0.0047 & 0.0283 & -0.0089 \\ -0.0268 & 0.0438 & -0.0288 & 0.1605 \end{pmatrix}$$

Matrici interne inerenti alla fase di Autoattenzione

$$\mathbf{Q} = \begin{pmatrix} 0.0642 & 0.0533 & 0.0021 & 0.0994 \\ 0.0662 & 0.0240 & 0.0067 & 0.0902 \\ 0.0756 & 0.0323 & 0.0129 & 0.0931 \\ 0.0564 & -0.0235 & -0.0012 & 0.0924 \\ 0.0686 & 0.0595 & 0.0072 & 0.0903 \\ 0.0734 & 0.0220 & 0.0099 & 0.0937 \\ 0.0637 & 0.0308 & 0.0027 & 0.0918 \\ -0.0538 & 0.2110 & -0.1157 & 0.0989 \end{pmatrix} \quad \mathbf{K} = \begin{pmatrix} -0.0363 & 0.0391 & 0.0759 & -0.0584 \\ -0.0388 & 0.0249 & 0.0717 & -0.0486 \\ -0.0497 & 0.0331 & 0.0907 & -0.0664 \\ -0.0221 & -0.0100 & 0.0609 & -0.0101 \\ -0.0458 & 0.0482 & 0.0757 & -0.0720 \\ -0.0468 & 0.0254 & 0.0912 & -0.0605 \\ -0.0371 & 0.0274 & 0.0723 & -0.0511 \\ 0.0666 & 0.0871 & -0.0783 & -0.0154 \end{pmatrix}$$

$$\mathbf{V} = \begin{pmatrix} 0.0448 & 0.1373 & -0.0780 & 0.1057 \\ 0.0524 & 0.1207 & -0.0828 & 0.0888 \\ 0.0581 & 0.1338 & -0.0968 & 0.1018 \\ 0.0650 & 0.0995 & -0.0604 & 0.0231 \\ 0.0426 & 0.1360 & -0.0912 & 0.1285 \\ 0.0635 & 0.1311 & -0.0921 & 0.0848 \\ 0.0521 & 0.1249 & -0.0792 & 0.0885 \\ -0.0494 & 0.1370 & 0.0784 & 0.1233 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 0.0411 & 0.1275 & -0.0626 & 0.0931 \\ 0.0411 & 0.1275 & -0.0627 & 0.0931 \\ 0.0411 & 0.1275 & -0.0627 & 0.0931 \\ 0.0411 & 0.1275 & -0.0627 & 0.0931 \\ 0.0411 & 0.1275 & -0.0626 & 0.0931 \\ 0.0411 & 0.1275 & -0.0627 & 0.0931 \\ 0.0411 & 0.1275 & -0.0627 & 0.0931 \\ 0.0410 & 0.1276 & -0.0625 & 0.0931 \end{pmatrix}$$

Matrice e vettore di output attinenti alla fase di Autoattenzione

$$\mathbf{W}_O = \begin{pmatrix} -0.3952 & -0.5285 & -0.6704 & 0.6683 \\ -0.2380 & 0.1196 & -0.5715 & 0.3333 \\ -0.1172 & -0.3465 & 0.5597 & -0.4401 \\ -0.5406 & -0.7400 & -0.7868 & 0.2903 \end{pmatrix} \quad \mathbf{w}_0 = \begin{pmatrix} -0.0278 & 0.0405 & -0.0146 & 0.1571 \end{pmatrix}$$

Matrice in $\mathbf{A}_O$ in uscita dall'Autoattenzione confrontata con la matrice $\mathbf{Y} = \mathbf{X} + \mathbf{A}_O$ (non normalizzata)

$$\mathbf{A}_O = \begin{pmatrix} -0.1174 & -0.0131 & -0.2233 & 0.2816 \\ -0.1173 & -0.0131 & -0.2233 & 0.2816 \\ -0.1173 & -0.0131 & -0.2233 & 0.2816 \\ -0.1173 & -0.0131 & -0.2233 & 0.2816 \\ -0.1174 & -0.0131 & -0.2233 & 0.2816 \\ -0.1173 & -0.0131 & -0.2233 & 0.2816 \\ -0.1173 & -0.0131 & -0.2233 & 0.2816 \\ -0.1174 & -0.0131 & -0.2232 & 0.2815 \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} -0.2079 & -0.0165 & -0.1958 & 0.2839 \\ -0.1934 & -0.0245 & -0.1966 & 0.2683 \\ -0.2120 & -0.0198 & -0.1823 & 0.2710 \\ -0.1447 & -0.0107 & -0.2224 & 0.2244 \\ -0.2218 & -0.0284 & -0.1819 & 0.2997 \\ -0.2012 & -0.0119 & -0.1847 & 0.2614 \\ -0.1938 & -0.0178 & -0.1949 & 0.2728 \\ -0.1442 & 0.0306 & -0.2520 & 0.4420 \end{pmatrix}$$

Matrice $\mathbf{Z}$ (già normalizzata) corrispondente all'uscita della fase **(vi)** in riferimento alla sezione 2.1

$$\mathbf{Z} = \begin{pmatrix} -0.8464 & 0.8430 & -0.8696 & 1.4187 \\ -0.8214 & 0.8303 & -0.8904 & 1.4272 \\ -0.8745 & 0.8535 & -0.8449 & 1.4112 \\ -0.6815 & 0.7987 & -1.0053 & 1.4336 \\ -0.8841 & 0.8508 & -0.8349 & 1.4134 \\ -0.8559 & 0.8501 & -0.8621 & 1.4134 \\ -0.8255 & 0.8340 & -0.8873 & 1.4246 \\ -0.6941 & 0.7932 & -0.9939 & 1.4402 \end{pmatrix} \quad \mathbf{z}_8 = \begin{pmatrix} -0.6941 & 0.7932 & -0.9939 & 1.4402 \end{pmatrix}$$

In questo particolare esempio, essendo $d = v$, il passaggio attraverso la matrice $\mathbf{W}_3$ e il vettore $\mathbf{w}_3$ si sarebbe potuto eliminare, ma anche in questo caso si è preferita usare la struttura generale del modello di Transformer.

$$\mathbf{W}_3 = \begin{pmatrix} 0.2584 & 0.8435 & -0.6804 & 0.6328 \\ 0.3017 & -0.5149 & -0.0316 & -0.9219 \\ 0.0496 & 0.0567 & 0.4545 & -0.6791 \\ 0.0840 & 0.4179 & -0.7622 & -0.3784 \end{pmatrix} \quad \mathbf{w}_3 = \begin{pmatrix} 0.0937 & 0.0033 & -0.0699 & -0.0930 \end{pmatrix}$$

Vettore $\mathbf{logit} = \mathbf{z}_8 \cdot \mathbf{W}_3 + \mathbf{w}_3 = \begin{pmatrix} 0.2254 & -0.4450 & -1.1721 & -1.1334 \end{pmatrix}$
Infine si calcola il vettore $\mathbf{q} = softmax(\mathbf{logit}) = \begin{pmatrix} 0.4961 & 0.2538 & 0.1226 & 0.1275 \end{pmatrix}$
La parola prevista dal Transformer corrispettiva all'indice $n + 1 = 9$ della sequenza di ingresso verrà scelta secondo il vettore di probabilità $\mathbf{q}$, ovvero $\mathrm{P}[x_9 = a_i] = q_i$.
Confrontando i vettori $\mathbf{p}$ e $\mathbf{q}$ si ottiene $\mathbf{e} = |\mathbf{p} - \mathbf{q}| = \begin{pmatrix} 0.0039 & 0.0038 & 0.0024 & 00.25 \end{pmatrix}$ che evidenzia uno scostamento massimo dalla probabilità teorica del 0.39%.

4 Codice Sorgente

Per completezza è stato reso disponibile il codice sorgente più significativo usato nella fase di creazione e allenamento del Transformer.
Al fine di evidenziare solamente i passaggi essenziali e garantire una lettura del codice più fluida, sono state eliminate tutti le operazioni secondarie, per la maggior parte inerenti alla memorizzazione e creazione di variabili. Le costanti sono riportate in maiuscolo senza indicare il loro preciso valore.

4.1 Creare un Transformer

```
function transformer = getTransformer(v_size, seq_length, embed_d, attention_d, ff_d)
  numHeads = 1;
  transformerLayers = [
    sequenceInputLayer(1,Name="input"); % input vettore colonna di lunghezza n
    wordEmbeddingLayer(embed_d, v_size, Name="embedding"); % matrice n × d
    positionEmbeddingLayer(embed_d, seq_length, Name="position")
    additionLayer(2, Name="embed_add"); % somma w. e p. embedding, n × d
    selfAttentionLayer(numHeads, attention_d); % matrice A_O, n × d
    additionLayer(2, Name="attention_add"); % somma Y = X + A_O, n × d
    layerNormalizationLayer(Name="attention_norm");
    fullyConnectedLayer(ff_d); % operazioni con W_1 e w_1, output n × r
    reluLayer; % ReLU sulla matrice precedentemente ottenuta
    fullyConnectedLayer(embed_d); % operazioni con W_2 e w_2, in uscita F, n × d
    additionLayer(2, Name="feedforward_add"); % somma Z = Y + F, dim: n × d
    layerNormalizationLayer(Name="encoder1_out");
    indexing1dLayer(); % seleziona solo l'ultima riga della matrice, 1 × d
    fullyConnectedLayer(v_size); % operazioni con W_3 e w_3, output dim: 1 × v
    softmaxLayer; % vettore q corrispondente alla previsione, dim 1 × v
  ];
  transformer = dlnetwork(transformerLayers,Initialize=false);
  transformer = connectLayers(transformer,"embedding","embed_add/in2");
  transformer = connectLayers(transformer,"embed_add","attention_add/in2");
  transformer = connectLayers(transformer,"attention_norm","feedforward_add/in2");
  transformer = initialize(transformer);
end
```

4.2 Allenare un Transformer

```
XTrain; % matrice di allenamento, ogni riga raprresenta una sequenza (campioni)
YTrain = categorical(YTrain); % vettore colonna inerente alla parola n+1 (etichette)
classes = categories(YTrain); % valori rappresentanti indici delle parole
numObservations = numel(YTrain); % numero di test effettuati
numIterationsPerEpoch = floor(numObservations./BATCH_SIZE); % test per ciclo
numIterations = EPOCH * numIterationsPerEpoch; % numero di test totali
avGrad = [ ], avSqGrad = [ ]; % parametri imparabili del gradiente
monitor = trainingProgressMonitor(Metrics="Loss", Info="Epoch", XLabel="Iteration");
iteration = 0, epoch = 0;
while epoch < EPOCH && ~monitor.Stop
  epoch = epoch + 1;
  idx = randperm(size(YTrain, 1)); % permuta l'ordine delle osservazioni
  XTrain = XTrain(idx, :);
  YTrain = YTrain(idx);
  i = 0;
```

```
  while i < numIterationsPerEpoch && ~monitor.Stop
    i = i + 1, iteration = iteration + 1;
    idx = (i-1) * BATCH_SIZE + 1 : i * BATCH_SIZE; % 1 × BATCH_SIZE
    X = XTrain(idx, :)'; % matrice n × BATCH_SIZE
    X = dlarray(single(X), "TBC"); % formato MATLAB per interpretare i dati
    E = zeros(VOCAB_SIZE, BATCH_SIZE, "single"); % etichette Y → e
    for c = 1:VOCAB_SIZE % le colonne di E saranno vettori della base canonica
      E(c, YTrain(idx) == classes(c)) = 1;
    end
    % Valuta il modello di perdita e i gradienti
    [loss, gradients, Y] = dlfeval(@modelLoss, net, X, E); % aggiorna i parametri
    [net, avGrad, avSqGrad] = adamupdate(net, gradients, avGrad, avSqGrad, iteration);
    recordMetrics(monitor, iteration, Loss=loss);
    updateInfo(monitor, Epoch=epoch + " of " + EPOCH);
    monitor.Progress = 100 * iteration/numIterations;
  end
end

function [loss, gradients, Y] = modelLoss(net, X, E)
  Y = forward(net, X); % matrice di vettori probabilità q, v× BATCH_SIZE
  loss = crossentropy(Y, E); % media delle crossentropy su coppie vettore-etichetta
  gradients = dlgradient(loss, net.Learnables);
end
```

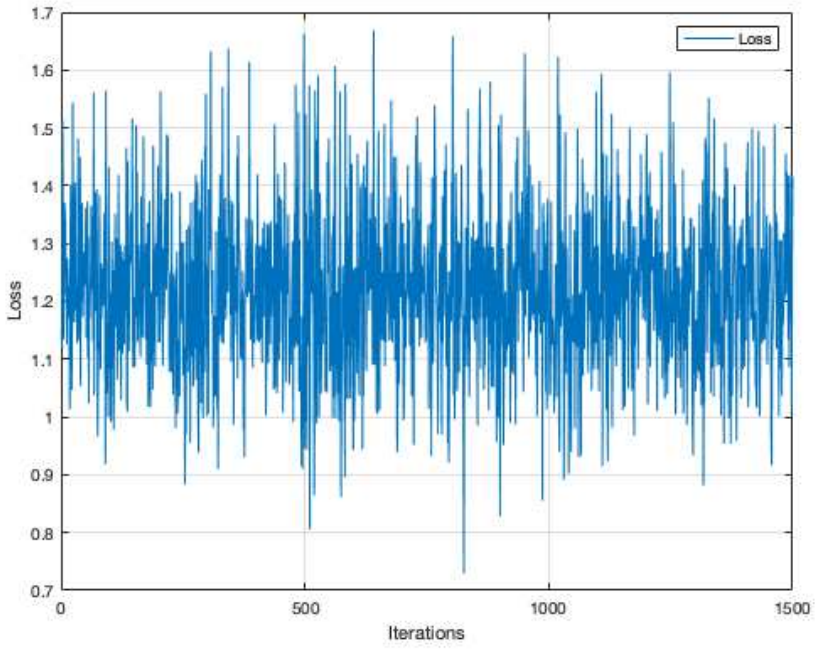

Figura 1: Funzioni di perdita calcolate in fase di allenamento

5 Conclusioni

Nel paragrafo 3.2 si può notare l'importanza della fase di Autoattenzione poiché la matrice $\mathbf{A}_O$ presenta valori superiori oltre un ordine di grandezza rispetto alla matrice $\mathbf{X}$ influendo maggiormente nella matrice $\mathbf{Y}$. Tale caratteristica è presente in tutti i Transformer studiati in questo testo.
Un secondo punto fondamentale, in accordo con le aspettative, è dato dalla somiglianza fra le righe della matrice $\mathbf{Y}$ e, di conseguenza, $\mathbf{Z}$. Per costruzione ogni riga della matrice $\mathbf{Z}$, $\mathbf{z}_i$ corrisponde alla previsione della parola $i+1$, con $i \in \{1, \dots, n\}$. Ricordando che le parole delle sequenze sono tutte statisticamente indipendenti, quindi non correlate fra loro, il fatto che tutte le righe di tale matrice siano approssimativamente uguali è la prova che il Transformer abbia individuato l'assenza di memoria della sorgente.
Una conseguenza di questo fatto è stata riscontrata nella sezione 3.1 poiché ogni Transformer allenato ha generato in uscita vettori di probabilità $\mathbf{q}$ strettamente simili fra loro (intervalli di variazione minori dell'1% rispetto ai valori nelle Tabelle 3, 4 e 5).
Sebbene sia vero che una volta allenato il Transformer generi in uscita lo stesso vettore di probabilità $\mathbf{q}$ per qualsiasi sequenza in ingresso, è sorto però un problema dato dalla precisione con cui il vettore medio $\mathbf{q}$ approssima il vettore $\mathbf{p}$ (visibile nelle Tabelle 3, 4 e 5). Tale imprevisto sorge in fase di allenamento, come si può notare dalla figura 1, nonostante il modello riesca a far convergere la media delle funzioni di perdita *loss* verso valore teorico $H(\mathbf{p}, \mathbf{p})$ già dalle prime iterazioni. Il Transformer verrà di conseguenza indirizzato sin dal principio verso la distribuzione $\mathbf{p}$, ma a causa dell'aleatorietà delle sequenze in ingresso, esso tenderà sempre a oscillare verso tale media. Se la fase di allenamento dovesse terminare durante un periodo di oscillazione allora tutti i vettori di probabilità $\mathbf{q}$ in uscita dal Transformer, anche se sempre simili fra loro, presenteranno valori discostati dalla probabilità teorica $\mathbf{p}$.
Per ridurre il rischio di incorrere in tale errore è consigliato adattare il numero di iperparametri del Transformer in proporzione alla taglia del problema. Sempre in riferimento alle Tabelle 3, 4 e 5 si noti come in media un minor numero di parametri di autoapprendimento permettano di avvicinarsi alla distribuzione teorica con maggior precisione, mentre aumentando tale quantità aumenti anche l'intervallo di oscillazione.

Riferimenti bibliografici

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin *I. Attention is all you need. In Advances in Neural Information* Processing Systems, pp. 5998–6008, 2017.

[2] Ashok Vardhan Makkuva, Marco Bondaschi, Adway Girish, Alliot Nagle, Martin Jaggi, Hyej Kim, and Michael Gastpar *Attention with Markov: A Framework for Principled Analysis of Transformers via Markov Chains* 2024

[3] Diederik P. Kingma, Jimmy Ba *Adam: A Method for Stochastic Optimization* 2017

[4] Nello Cristianini *Machina sapiens. L'algoritmo che ci ha rubato il segreto della conoscenza* 2024

[5] The MathWorks Inc. (2024). Optimization Toolbox version: 24.1.0.2603908 (R2024a), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com