ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

VISION-LANGUAGE TRANSFORMER: RETRIEVAL ED EXPLAINABILITY IN AMBITO BIOMEDICO

Elaborato in
Programmazione Di Applicazioni Data Intensive

Relatore
Prof. Gianluca Moro

Co-relatore
Ing. Stefano Salvatori

Presentata da
Lorenzo Zanetti

Terza Sessione di Laurea
Anno Accademico 2022 – 2023

PAROLE CHIAVE

Deep Neural Networks
Transformers
Metric Learning
Retrieval
Explainability

A chiunque mi sia stato vicino,
e mi abbia aiutato a raggiungere questo traguardo.

Introduzione

L'intelligenza artificiale (AI) è una delle principali discipline che negli ultimi tempi sta portando ad una grande rivoluzione tecnologica in innumerevoli settori. Uno dei motivi per cui l'utilizzo di tecniche di AI è aumentato recentemente è la sempre più grande disponibilità di dati in formato digitale ed accessibile. Infatti, assistiamo all'introduzione in sempre più settori e ambiti, di dispositivi elettronici e sensori di ogni genere. Grandi moli di dati permettono poi l'estrazione di conoscenza in modo più accurato da parte degli algoritmi di *machine learning* e, di conseguenza, si può fare utilizzo di queste tecnologie in sempre più ambiti, dove è richiesto un minor margine di errore. In particolare, in ambito biomedico queste tecniche hanno il potenziale di migliorare la salute pubblica, ridurre il numero di risorse impiegate e portare ad un avanzamento più rapido della ricerca.

Una delle difficoltà che si pone in questo campo è la grande quantità di dati non strutturati, come testi e immagini, che possono essere di difficile gestione ed interpretazione anche da parte di medici esperti. Lo scopo di questa tesi è quindi quello di utilizzare tecniche di deep learning allo stato dell'arte per affrontare un problema di analisi multi-modale per l'elaborazione automatica di radiografie (immagini) e referti medici (testi). Come è facile immaginare questi tipi di dato sono difficilmente interpretabili da un essere umano e anche per un esperto radiologo non è sempre semplice analizzare correttamente delle radiografie, considerando che queste si differenziano per piccoli dettagli e le diagnosi non hanno un formato strutturato ma sono fornite in linguaggio naturale, spesso estremamente specifico. E' per questo motivo che l'utilizzo delle ultime tecniche di *deep learning* potrebbe portare alla produzione di applicativi che aiutano uno specialista nello svolgere il proprio lavoro fornendo utili strumenti di supporto. L'obiettivo che ci poniamo è quello di sviluppare un modello di machine learning capace di analizzare il contenuto di radiografie e referti medici riconoscendo e comprendendo le relazioni semantiche che intercorrono tra essi. Questo modello verrà quindi utilizzato, fornendo anche un applicativo web di esempio, in un task di retrieval cross-modale sul dataset di radiografie e referti medici MIMIC-CXR.

All'interno di questo elaborato verranno analizzate e sperimentate recenti tecnologie di *retrieval cross modale* ed *explainability* in ambito biomedico, allo

scopo di identificare una radiografia toracica partendo dal referto che ne riporta le osservazioni e/o la diagnosi (retrieval) e di poter "spiegare" le scelte fatte dal nostro modello evidenziando le parti del referto e della radiografia più rilevanti (explainability).

L'ambito vision-language (quindi l'analisi di testi e immagini) è uno dei più complessi da trattare poiché si occupa di informazioni destrutturate. Questo significa che i dati a disposizione presentano ambiguità e sono difficilmente interpretabili. Nel nostro caso particolare, questo risulta ancora più accentuato, poiché i referti risultano generalmente prolissi ed utilizzano un linguaggio tecnico, mentre le radiografie sono estremamente simili tra loro ed hanno un'alta risoluzione.

Il lavoro di tesi è stato suddiviso nei seguenti capitoli:

- **Capitolo 1** - Panoramica sulle varie tecnologie esistenti in letteratura utilizzabili per il problema posto;
- **Capitolo 2** - Analisi del dataset utilizzato;
- **Capitolo 3** - Modellazione del problema con i primi approcci alle difficoltà riscontrate e alle relative soluzioni;
- **Capitolo 4** - Valutazione dei risultati ottenuti;
- **Capitolo 5** - Panoramica dell'applicativo utilizzabile dall'eventuale utente finale;

Indice

1 Stato dell'arte **1**
1.1 Machine learning e deep learning 1
1.1.1 Reti neurali 2
1.1.2 Reti neurali ricorrenti (RNN) 3
1.2 Meccanismo di Attention nelle RNN 5
1.2.1 Transformers 7
1.2.2 BERT 10
1.2.3 ViT 12
1.3 Vision and language pre-training models 14
1.3.1 ViLT 14
1.4 Deep Metric Learning 16

2 Dataset **19**
2.1 Introduzione al dataset 19
2.2 Analisi del dataset 22

3 Sviluppo **25**
3.1 Architettura del modello 25
3.1.1 Task di addestramento 26
3.1.2 Algoritmi di explainability 28
3.2 Retrieval con metric learning 29
3.3 Librerie utilizzate 31

4 Esperimenti **33**
4.1 Addestramento del modello 33
4.2 Risultati di explainability 35
4.3 Risultati del retrieval 38

5 Applicativo Web **41**

Conclusioni e sviluppi futuri **45**

Ringraziamenti **47**

Bibliografia **49**

Elenco delle figure

1.1 Gerarchia delle tecnologie di intelligenza artificiale. 2

1.2 Rete neurale composta da numerosi strati: uno di input, uno di output e multipli strati intermedi, detti anche hidden layers, ogni strato a sua volta composto da numerosi nodi. 3

1.3 A sinistra una rete neurale ricorrente e a destra una normale rete neurale. 4

1.4 Esempio di RNN che utilizza il modello Seq2Seq: a sinistra tre encoder che elaborano i dati in hidden state e a destra tre decoder che elaborano l'hidden state finale in un output. 4

1.5 Meccanismo di global attention che, dato un hidden state h_t e i precedenti $\overline{h}_s$ produce un context vector c_t e dei pesi a_t su cui fa una media pesata per calcolare il nuovo vettore $\tilde{h}_t$. 6

1.6 Matrice d'attention prodotta nel nostro esempio, la trasparenza corrisponde al valore numerico. 7

1.7 Schema di un generico Transformer. Riporta in rosa gli embedder, con delle sfere i positional encoding, in arancione i meccanismi di attention, in blu le feed forward neural newtwork e in giallo i meccanismi di normalizzazione. 9

1.8 Punteggi BLEU (utilizzato nella traduzione) dei modelli allo stato dell'arte comparati con un Trasformer al tempo della sua ideazione e costo computazionale in numero di operazioni in virgola mobile. 10

1.9 Reppresentazione dell'input utilizzato per BERT, sommando token che rappresentano la parola dell'alfabeto, l'appartenenza ad una delle due frasi e la posizione. 11

1.10 In questa tabella sono riportati i risultati di BioBERT in task di question answering. Le metriche sono: Accuracy (S), Lenient Accuracy (L) e Mean Reciprocal Rank (M), le migliori sono in grassetto e le seconde migliori sono sottolineate. Wiki, Books, PubMed e PMC sono i dataset su cui i modelli sono stati addestrati (i primi due generici, gli altri due in ambito scientifico). 12

1.11 Risultati rappresentati come media e varianza su 3 testset da ViT nella versione L (large) e H (huge) addestrati sul dataset JFT-300M o ImageNet-21k (I21k) contro lo stato dell'arte (BiT-K e Noisy Student). In grassetto i risultati migliori e in basso il costo computazionale del preaddestramento in giorni/core di una TPUv3 13
1.12 Schematizzazione del modello ViT. A destra la struttura del singolo encoder e a sinistra il meccanismo di embedding e di modellazione dell'output dal context prodotto dall'encoder (MLP layer). 13
1.13 Schematizzazione del modello ViLT. Immagine e testo vengono processati e poi passati all'encoder come un unico input, poi l'output può essere elaborato in vari modi in base all'obiettivo. . 15
1.14 Risultati ottenuti da ViLT fine-tuned nei task di text e image retrieval rispetto allo stato dell'arte. E' riportata, per ogni dataset (Flickr30k, MSCOCO), l'accuratezza sulla prima risposta (R@1), le prime 5 (R@5) o 10 (R@10). † indica l'utilizzo aggiuntivo dei dataset GQA, VQAv2, VG-QA in fase di pretraining, ‡ indica l'utilizzo aggiuntivo del dataset Open Images. ⓐ indica l'utilizzo di RandAugment, meccanismo che aggiunge al dataset versioni modificate delle stesse immagini e ⊕ indica un numero maggiore di step di pretraining. 16
1.15 Viene illustrato il funzionamento della triplet margin loss dove, preso un dato X vengono selezionati altri due valori, uno simile e uno differente. Ognuno di questi dati viene elaborato dalla rete neurale e gli "embeddings" creati vengono poi confrontati calcolandone le distanze. 17

2.1 Una radiografia di esempio del dataset, la figura 2.1a, a sinistra, mostra la versione originale della radiografia e nella figura 2.1b, a destra, viene evidenziata la costola rotta. 20
2.2 Viene mostrata la gerarchia in cui sono suddivisi i files. La prima cartella indica le iniziali degli identificatori dei pazienti contenuti al suo interno (p10), la seconda ha come nome l'identificatore del paziente (p10000032) e l'ultima identifica le singole sedute (s50414267, s53189527, s53911762 e s56699142). 21
2.3 Istogramma che riporta la distribuzione della lunghezza dei referti in token (creati con il tokenizer di BERT). 23
2.4 Vi sono nelle figure 2.4a e 2.4b mostrate rispettivamente la distribuzione dell'altezza e della larghezza delle immagini delle radiografie in pixel. 24

3.1 Viene illustrato il modello che, dato in input un referto e un'immagine, li suddivide rispettivamente in token e in patch, poi aggiunge dei valori relativi al loro tipo (token speciali o del testo, patch di padding o dell'immagine) e dei valori relativi alla posizione. Questi embeddings vengono poi elaborati da un transformer encoder che produce un vettore context. Il context viene poi elaborato da un'unità MLMHead per produrre delle predizioni sulle parole mascherate e da un classificatore lineare per decidere se gli input sono o meno correlati tra di loro. 27

3.2 Viene illustrato visivamente il funzionamento del metric learning. Nella figura a si parte dagli embeddings generati dalla rete neurale, che hanno una certa distanza euclidea (figura b). L'obiettivo mostrato in figura c è quindi quello di creare una metrica che "allontana" dati diversi e "avvicina" quelli simili, fino ad arrivare al risultato mostrato in figura e. 30

4.1 Nel grafico viene mostrata la minimizzazione della funzione di loss dei modelli, che risulta molto simile in tutti e tre i casi e tende a stabilizzarsi. 34

4.2 Come si può vedere l'algoritmo di explainability non è molto preciso, ma comunque riesce ad evidenziare come parti più importanti della radiografia (in rosso) i contorni dei polmoni e del cuore. Leggendo la radiografia corrispondente, mostrata in figura 4.3, capiamo infatti che il paziente non presenta particolari anomalie poiché gli organi hanno un volume corretto. 36

4.3 Come si legge dal referto cuore e polmoni sono delle giuste dimensioni e sono sani. Il modello, oltre ad aver correttamente classificato radiografia e referto correlate, evidenzia nel referto le parole più significative per la decisione (in verde): "dimensioni normali", "polmoni", "cuore", "chiari e allargati". 36

4.4 Come si può vedere l'algoritmo di explainability non è molto preciso nemmeno qui, in più vengono evidenziate parti di discutibile rilevanza, per esempio l'angolo dell'immagine che non contiene nulla. 37

4.5 Questo referto, per esempio, non viene analizzato correttamente dal modello, in quanto vengono evidenziate parole come "indicazioni" o degli spazi vuoti nel referto (creati dalla censura del nome del paziente). E anche la predizione fatta "not related" non è corretta. 37

4.6 Nell'immagine 4.6a mostriamo una radiografia evidenziando manualmente la rottura di una costola, indicata anche sul referto. Nell'immagine 4.6b è mostrata la stessa radiografia analizzata con l'algoritmo di attention rollout che in blu evidenzia le parti rilevanti dell'immagine. Anche in questo caso l'attenzione è rivolta alla sesta costola nella parte destra del torace. 38

5.1 Viene mostrato un possibile record, dove il referto viene usato come input e la radiografia è il risultato atteso. 42
5.2 Viene mostrato l'output del modello suddiviso in pagine che riporta le radiografie ordinate per similarità. In questo caso la radiografia attesa è stata restituita come primo risultato. 43
5.3 Viene mostrato il campo di input con una ricerca di esempio. . . 44
5.4 Viene mostrato l'output del modello suddiviso in pagine che riporta le 100 radiografie più coerenti alla ricerca "pleural effusion". 44

Capitolo 1

Stato dell'arte

In questa prima parte verrà fatto un quadro generale dell'ambito di ricerca nel quale abbiamo lavorato, fino ad andare nelle specifiche tecnologie ed algoritmi utilizzati come punti di partenza. Questo viene fatto per dare un contesto e permettere una maggiore comprensione dei risultati e delle soluzioni proposte in questo lavoro.

1.1 Machine learning e deep learning

Il *machine learning* è un settore dell'informatica e più nello specifico dell'intelligenza artificiale (si veda figura 1.1) relativamente giovane che sta sempre di più avendo un impatto forte e positivo in vari settori, dovuto all'esponenziale crescita di dati disponibili in ambito digitale. Per questo motivo ricercatori, aziende e altri enti stanno mettendo a disposizione sempre più risorse nella ricerca e nello sviluppo di nuove tecnologie in quest'ambito.

Alcuni esempi dei recenti obiettivi raggiunti dal machine learning possono essere: diagnosi mediche, assistenti vocali, automazione di servizi, auto a guida autonoma, sentiment analysis ecc.

Il *deep learning* è una disciplina contenuta all'interno di questo più ampio insieme del machine learning. Rispetto ai tradizionali metodi di apprendimento automatico, il deep learning ha permesso di ottenere risultati migliori in svariati task, soprattutto in quelli basati sulla comprensione di dati non strutturati, come testi ed immagini.

Le reti neurali profonde, infatti, che sono alla base del deep learning, non necessitano di preprocessare i dati in un formato strutturato. Sono in grado di determinare autonomamente le feature necessarie per risolvere qualsiasi tipo di problema (e.g., classificazione di testi o immagini, riconoscimento di lettere e numeri scritti a mano, e nei casi più avanzati anche di generare immagini realistiche o testi di senso compiuto).

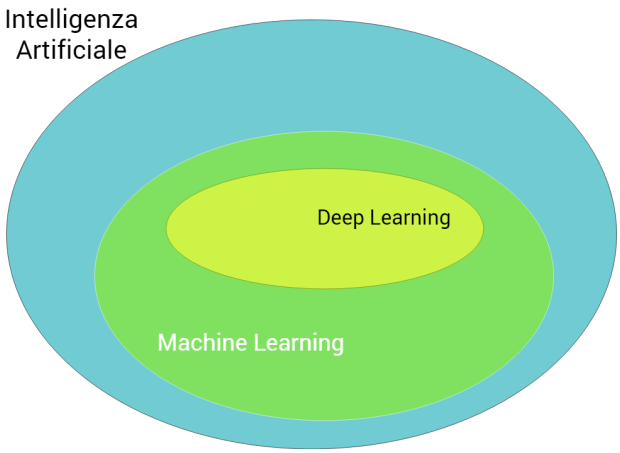


Figura 1.1: Gerarchia delle tecnologie di intelligenza artificiale.

1.1.1 Reti neurali

Le **reti neurali** (neural networks), anche conosciute come artificial neural networks (ANNs), sono dei modelli utilizzati nel machine learning che si ispirano al funzionamento del cervello umano. Questi modelli sono infatti composti da numerosi strati: uno di input, uno di output e multipli strati intermedi, detti anche *nascosti*, ognuno dei quali è composto a sua volta da numerosi nodi (detti anche neuroni).

Per ogni strato ci sono dei valori dati in input che vengono suddivisi e letti dai nodi che lo compongono e, una volta processati, nuovi valori vengono propagati ai nodi dello strato successivo, fino ad arrivare a quello finale che produce un output con un certo significato in base al problema per cui è stata costruita la rete neurale.

Ciascuno dei nodi della rete neurale corrisponde metaforicamente ad un neurone e in pratica ad un'unità di calcolo, che può corrispondere ad un classificatore lineare o non lineare. Una delle prime reti neurali, chiamata *feedforward neural network* (FFNN) o multi-layered perceptron (MLP), utilizza infatti come *unità fondamentale* il perceptron, che è appunto un modello di classificazione lineare.

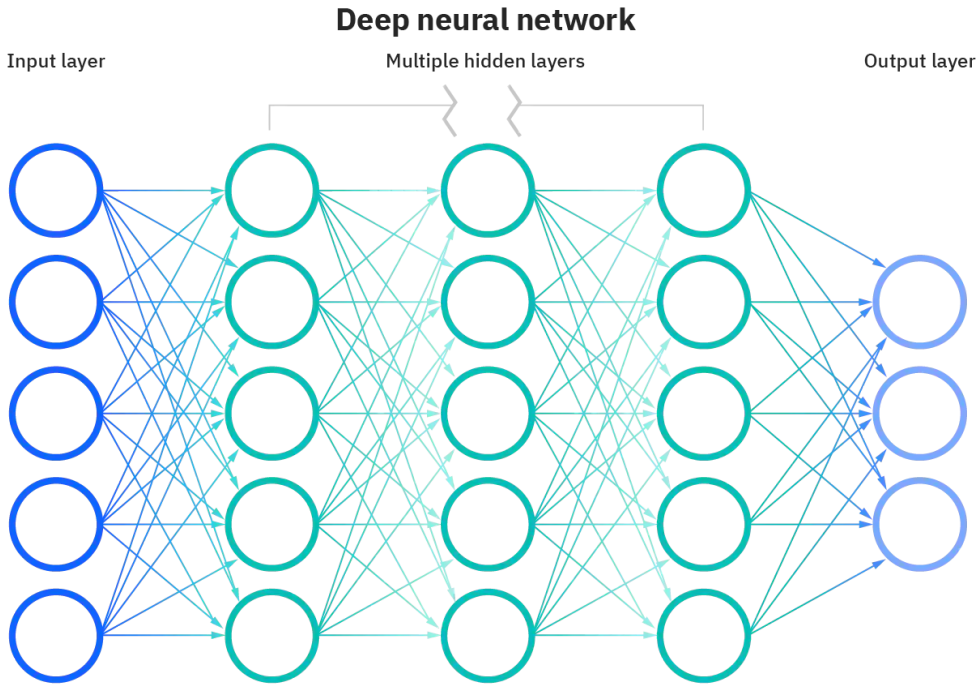


Figura 1.2: Rete neurale composta da numerosi strati: uno di input, uno di output e multipli strati intermedi, detti anche hidden layers, ogni strato a sua volta composto da numerosi nodi.

1.1.2 Reti neurali ricorrenti (RNN)

Nel paragrafo precedente è stato illustrato il concetto generale di rete neurale, ma come molti altri modelli impiegati nel machine learning, esistono diverse versioni implementative di uno stesso concetto. In particolare, le reti neurali, possono implementare diversi modelli ad ogni nodo e possono utilizzare un numero arbitrario di altri meccanismi che ne modificano il comportamento, adattandolo al singolo problema.

Prima sono state citate le feedforward neural network, cioè reti neurali *standard* in cui ogni strato utilizza il solo output dello strato precedente come input per produrre il risultato successivo. Il paradigma delle feedforward network non permette di considerare relazioni sequenziali e/o temporali nei dati. Per questo è possibile invece utilizzare le recurrent neural networks (RNN), nelle quali ogni nodo tiene in considerazione anche gli output prodotti nei *momenti* precedenti (si veda figura 1.3).

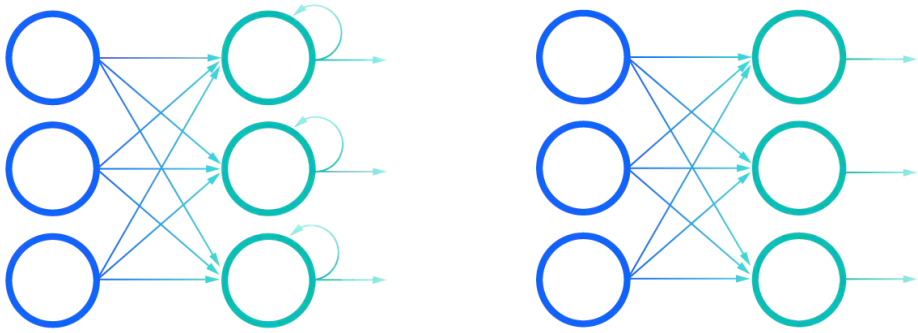


Figura 1.3: A sinistra una rete neurale ricorrente e a destra una normale rete neurale.

Questo meccanismo fa si che il singolo nodo abbia una sorta di *memoria interna*, grazie alla quale può fare predizioni anche in base ad informazioni avute in precedenza, invece che utilizzando soltanto gli input correnti. Questo è particolarmente utile nell'ambito del *natural language processing* (NLP), in task come traduzione, riconoscimento vocale o nel generare didascalie per delle immagini.

Un particolare tipo di RNN molto utilizzato è il modello *Sequence-to-sequence* (Seq2Seq) che utilizza una struttura suddivisa in due componenti principali: un *encoder*, che dato un input ne converte ogni parte in una particolare rappresentazione, chiamata *hidden state* (l'insieme di tutte le rappresentazioni che compongono l'input è poi chiamata *context*), e un *decoder*, che data in input questa rappresentazione la elabora per produrre l'output desiderato. Questo modus operandi è illustrato in figura 1.4 tornerà molto utile per descrivere l'architettura utilizzata in questo progetto.

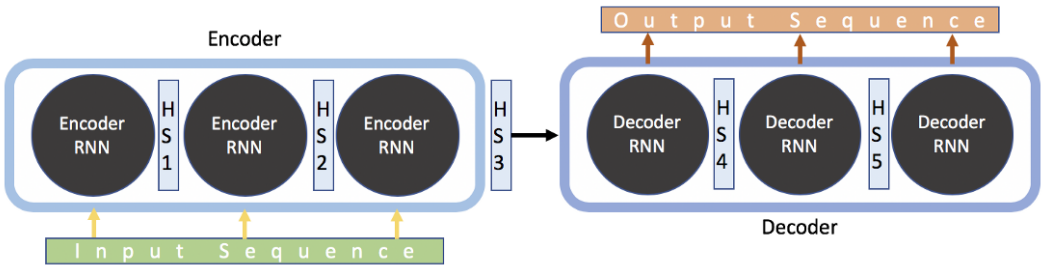


Figura 1.4: Esempio di RNN che utilizza il modello Seq2Seq: a sinistra tre encoder che elaborano i dati in hidden state e a destra tre decoder che elaborano l'hidden state finale in un output.

I modelli RNN standard utilizzano un meccanismo di *back propagation* per l'addestramento, cioè utilizzano l'errore sull'output ottenuto per modificare i pesi lungo tutta la rete. In questo algoritmo l'aggiornamento dei pesi avviene partendo dal layer finale, andando indietro, in modo che l'errore sia sempre meno rilevante andando verso l'inizio della rete (poiché si suppone che la prima trasformazione applicata ai dati avrà avuto meno impatto sul risultato finale). Per ottenere questo risultato, essendo spesso il valore utilizzato per queste modifiche molto piccolo, viene moltiplicato per se stesso muovendosi indietro nella rete, così che possa diminuire. Questo però porta al problema del *vanishing gradient*, cioè al fatto che con reti abbastanza grandi i valori dei primi layer diminuivano sempre di più durante l'addestramento, fino ad essere quasi nulli. Questo problema comporta che i dati passati in input ricevano trasformazioni minime, trascurabili rendendo l'utilizzo dei primi layer inutile, spostando tutto il lavoro sui layer finali, creando un circolo vizioso per il modello. Oltre a questo, un altro problema che hanno le RNN è il fatto che per lunghe sequenze di dati in input, che possono essere testi particolarmente lunghi e/o immagini, le modalità di rappresentazione intermedie non riescono a dare sufficiente importanza a tutte le parti dell'input, perdendo così importanti informazioni.

1.2 Meccanismo di Attention nelle RNN

Per cercare di risolvere questi problemi venne introdotto il meccanismo dell'attention [1], ispirato all'attenzione cognitiva degli essere umani, che permette al modello di capire a quali parti del testo dare maggiore importanza e a quali diminuirla. Questo meccanismo verrà poi adottato da diverse tecnologie, per esempio, nel mondo del natural language processing, darà origine alle architetture dei Transformers [2] o a quella di BERT [3], sviluppata da Google.

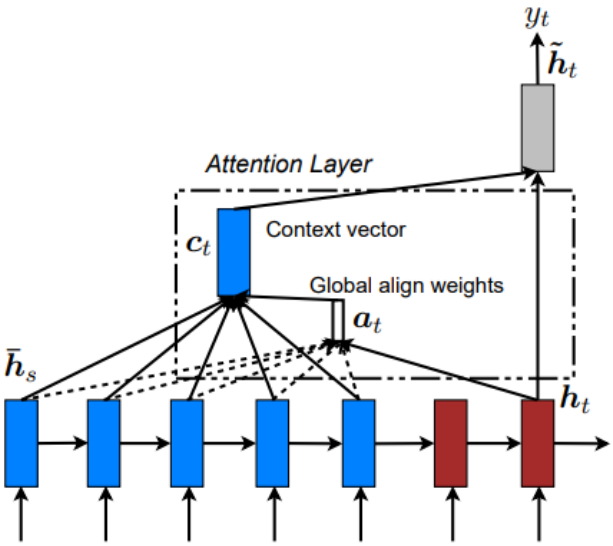


Figura 1.5: Meccanismo di global attention che, dato un hidden state h_t e i precedenti $\overline{h}_s$ produce un context vector c_t e dei pesi a_t su cui fa una media pesata per calcolare il nuovo vettore $\tilde{h}_t$.

Alcuni possibili approcci sono quelli chiamati di *global attention* e *local attention* [1]. Entrambi questi approcci si prefissano di ottenere, per un dato hidden state h_t e dato un *context vector* c_t, una versione più accurata $\tilde{h}_t$ che tiene conto di "alcuni" hidden states precedenti.

$$\tilde{h}_t = tanh(W_c[c_t; h_t]) \tag{1.1}$$

La versione di attention globale (illustrata in figura 1.5)utilizza tutti gli hidden states precedenti e quella locale solo una parte. In particolare, nell'attention globale viene calcolato un vettore di allineamento a_t di dimensione pari al numero degli hidden states dove ogni valore si calcola con la formula 1.2.

$$a_t(s) = \frac{exp(score(h_t, \overline{h}_t))}{\sum_{s'} exp(score(h_t, \overline{h}_t))} \tag{1.2}$$

Una volta calcolati, i valori di a_t vengono utilizzati come pesi, e il vettore di contesto c_t viene calcolato come media pesata partendo dagli hidden states. In questo algoritmo la funzione *score* citata nella formula 1.2 può assumere diverse forme tra cui:

$$h_t^T \overline{h}_s, \quad h_t^T W_a \overline{h}_s, \quad v_a^T tanh(W_a[h_t; \overline{h}_s]) \tag{1.3}$$

Per spiegare meglio il meccanismo dell'attention si può prendere come esempio un modello Seq2Seq, come mostrato in precedenza, applicato ad un

task di traduzione da francese ad inglese. Per prima cosa l'input, cioè la frase in francese, verrà suddivisa in *token* (in questo esempio supponiamo che un token corrisponda esattamente ad una parola, anche se questo non è sempre vero), poi l'encoder del modello produrrà per ciascuno di questi token un hidden state e unendoli formerà un context da passare come input al decoder. In contemporanea a questa codificazione verranno prodotti per ogni hidden state una serie di pesi utilizzati poi dal decoder per capire che importanza dare ad ogni parola dell'input per produrre ciascuno parola dell'output.

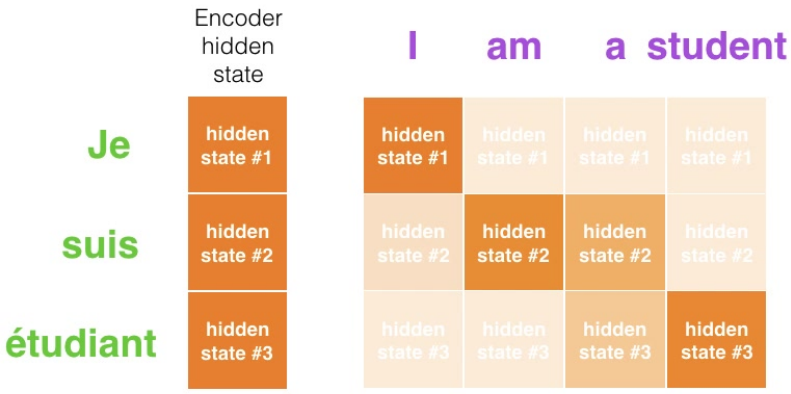


Figura 1.6: Matrice d'attention prodotta nel nostro esempio, la trasparenza corrisponde al valore numerico.

In questo caso particolare si può vedere in figura 1.6 come per le parole "I", "am" e "student" viene fatta praticamente una traduzione uno a uno dal francese, mentre per la parola "a" è necessario considerare sia il verbo che il complemento oggetto della frase originale perché il modello capisca che va inserito questo ulteriore articolo nella lingua di destinazione. In letteratura sono stati proposti vari tipi di attention [4, 1, 5] che hanno permesso di migliorare notevolmente le peformance delle reti neurali, soprattutto nei task di natural language processing. Tuttavia, le reti ricorrenti hanno seri problemi di efficienza in quanto sono difficilmente parallelizzabili e, come già accennato, soffrono del problema del vanishing gradient. Con l'idea di superare queste debolezze dei modelli seq2seq, è stato proposto nel 2017 il Transformer.

1.2.1 Transformers

Il Transformer [2] è un modello molto interessante che utilizza l'architettura encoder-decoder e fa largo uso del meccanismo di attention. In particolare,

questo modello utilizza la *self-attention* per determinare l'importanza che ogni altra parola ha rispetto ad una parola presa in considerazione, poi "passa" questi valori ad una feed forward neural network e da qui ottiene un vettore detto di contesto (context). Il context viene poi passato ad un decoder che utilizza nuovamente la self-attention, l'attention del decoder e una feed forward neural network per produrre un risultato che viene poi elaborato in modi specifici in base allo scopo per cui si sta utilizzando il modello. Nello specifico la self-attention è un meccanismo che produce un output (Z) calcolato a partire dall'input preprocessato e tre matrici di pesi (W_q, W_k, W_v) determinate durante l'addestramento del modello. Ogni vettore che compone l'input viene moltiplicato per ciascuna di queste matrici producendo tre vettori chiamati rispettivamente *queries*, *keys* e *values*. Per calcolare l'output di una data parola x il suo vettore query viene moltiplicato per i vettori keys di ogni parola (compresa se stessa) per produrre uno *score*, che indica appunto quanto ogni parola sia significativa per x. Ogni punteggio viene poi diviso per la radice della dimensione che avevano i vettori di input (d_k), viene fatta una softmax, ciascuno dei risultati viene moltiplicato per il vettore value di x e questi vettori risultanti vengono sommati per trovare l'output corrispondente di x. Queste operazioni possono essere fatte in parallelo per ogni parola, in particolare se prendiamo una matrice in cui ogni riga indica l'embedding di una delle parole (X) possiamo esprimere tutti i calcoli con queste formule:

$$Q = W_q X, K = W_k X, V = W_v X \tag{1.4}$$

$$softmax(\frac{QK^T}{\sqrt{d_k}})V = Z \tag{1.5}$$

Questo meccanismo di self-attention è molto efficace, ma non tiene conto del fatto che la stessa frase, con le stesse parole, ma un ordine diverso in cui sono poste, può avere un significato diverso. Per questo motivo viene spesso affiancata ai meccanismi di *positional embeddings* e alla *multi-headed attention*. La seconda, la multi-headed attention, consiste semplicemente nell'utilizzare la self-attention più volte in parallelo, al posto di tre matrici di pesi ce ne saranno, per esempio, ventiquattro e produrranno otto diversi output Z. Tutti questi output vengono poi concatenati e moltiplicati per una nuova matrice di pesi (anche questa addestrata con il modello) in modo da avere un'unica matrice Z della dimensione voluta. Il meccanismo dei positional embeddings è invece basato sul sommare agli input iniziali dei valori specifici che ne rappresentano la posizione all'interno della parola. Solitamente un Transformer fa uso di più encoder "impilati" l'uno sull'altro e un egual numero di decoder, la differenza con l'utilizzare un singolo encoder e decoder è che soltanto al primo encoder della pila verrà passato l'input preprocessato, mentre agli encoder successivi verrà

dato l'output dell'encoder precedente. Analogamente il primo decoder riceve in input l'output dell'ultimo encoder, mentre i successivi decoder ricevono l'output del decoder precedente. Nella figura 1.7 è mostrato uno schema riassuntivo di un Transformer che utilizza N blocchi di encoding e N blocchi di decoding per fare traduzione.

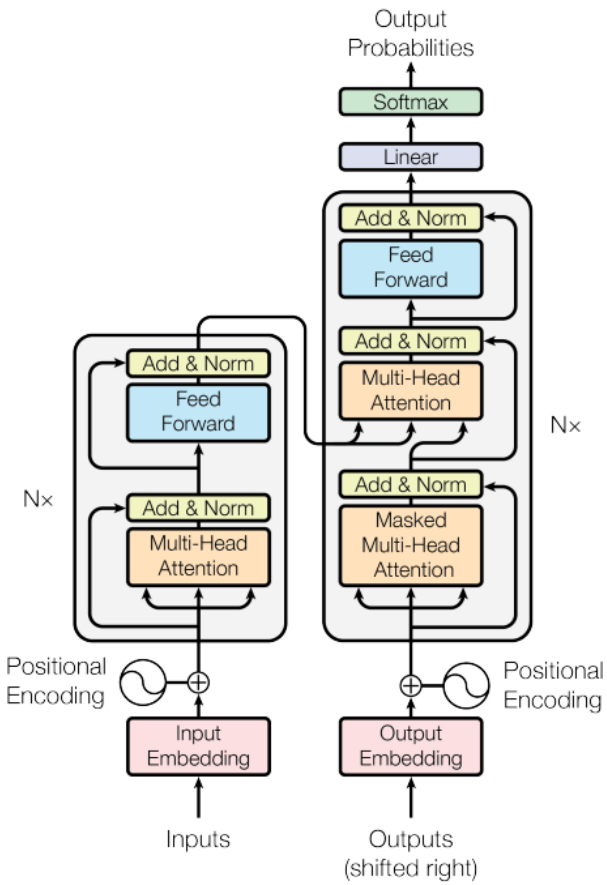


Figura 1.7: Schema di un generico Transformer. Riporta in rosa gli embedder, con delle sfere i positional encoding, in arancione i meccanismi di attention, in blu le feed forward neural newtwork e in giallo i meccanismi di normalizzazione.

Come si può vedere nell'immagine 1.7, i risultati della self-attention e della FFNN vengono sottoposti anche ad una *layer-normalization*, in cui i valori di output vengono sommati a quelli di input e normalizzati. Oltre a questo è importante notare che i decoder aggiungono un passaggio in più rispetto agli encoder, cioè utilzzano l'attention calcolata dall'ultimo encoder. Infine, in questo caso, gli output vengono elaborati da una rete neurale che produce

un punteggio per ogni parola del dizionario di destinazione e una softmax che trasforma questi punteggi in probabilità. Nell'immagine 1.8 si possono osservare i risultati dei Transformers che superano i modelli allo stato dell'arte mantenendo un costo computazionale molto minore.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Figura 1.8: Punteggi BLEU (utilizzato nella traduzione) dei modelli allo stato dell'arte comparati con un Trasformer al tempo della sua ideazione e costo computazionale in numero di operazioni in virgola mobile.

1.2.2 BERT

Nel 2019 viene pubblicato il paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [3], un nuovo modello per la rappresentazione di un linguaggio che segue il paradigma pre-train-and-fine-tune. Questo modello ha infatti l'obiettivo di creare, a partire da un testo, una rappresentazione vettoriale che può poi essere utilizzata e interpretata in vari modi, dipendentemente dallo scopo richiesto. BERT si basa sull'architettura dell'encoder di un Transformer, questo perché il suo obiettivo è soltanto quello di codificare i testi, ed in particolare è bidirezionale, cioè ogni output relativo ad un token del testo viene calcolato sulla base di tutti gli altri token, quindi non in modo sequenziale. Per addestrarlo sono stati utilizzati due task: masked language modeling e next sentence prediction. Il primo consiste nel nascondere casualmente parole del testo al modello per poi fargliele indovinare, mentre il secondo consiste nel, date due frasi, determinare se queste si succedono o no all'interno di un testo che le contiene.

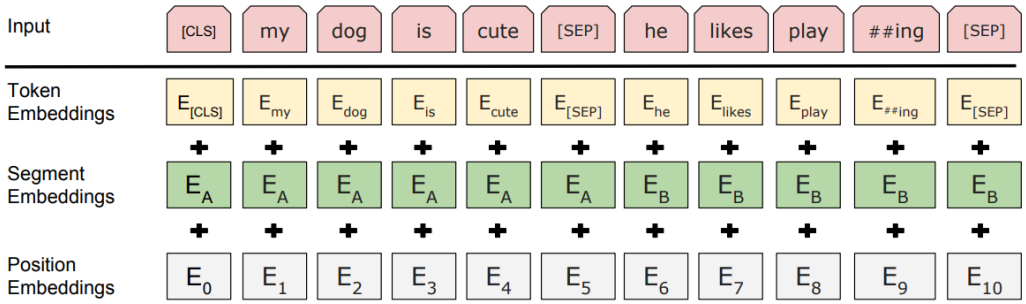


Figura 1.9: Reppresentazione dell'input utilizzato per BERT, sommando token che rappresentano la parola dell'alfabeto, l'appartenenza ad una delle due frasi e la posizione.

Come si nota dall'immagine 1.9, date due frasi, queste vengono suddivise in token e delimitate da dei token speciali: un token chiamato CLS posizionato all'inizio ed un token SEP per dividere i due testi in input. Fatto questo viene preso per ogni token il suo valore numerico (embedding) e gli viene sommato un valore che ne indica l'appartenenza ad una delle due frasi e uno che ne indica la posizione. Sono state addestrate due versioni di BERT: BERTbase, il cui decoder conta 12 layer, 768 hidden states e 12 attention heads e BERTlarge, con 24 layer, 1024 hidden states e 16 attention heads. Questo modello raggiunse ottimi risultati e segnò lo stato dell'arte in numerosi ambiti come sentiment analysis, question answering o question answering. L'efficacia di BERT in svariati task di NLP è dovuta in particolare al paradigma di pretraining e finetuning. Questo paradigma è composto da due fasi: prima di tutto vengono creati modelli *general purpose*, addestrati su dati diversificati e con task generici (come BERT) e, presi questi modelli, si riaddestrano su dati più specifici (a volte anche modificando i modelli iniziali e/o gli obiettivi). A causa della sempre maggiore complessità dei problemi di deep learning, e di conseguenza anche dei suoi modelli, questo nuovo procedimento permette prima di tutto di diminuire lo spreco di risorse per creare modelli simili e, in secondo luogo, aumenta la precisione del modello che, dopo aver acquisito conoscenza generale, è più *robusto* nel passaggio a dati e domini applicativi più specifici. Da BERT nacquero poi alternative come per esempio ALBERT [6] o DistilBERT [7], più leggeri del modello originale e anche modelli adattati a specifici domini, come BioBERT [8], vediamo i risultati ottenuti in figura 1.10.

			BERT	BioBERT v1.0		BioBERT v1.1
Datasets	Metrics	SOTA	(Wiki + Books)	(+ PubMed)	(+ PubMed + PMC)	(+ PubMed)
BioASQ 4b	S	20.01	27.33	25.47	**28.57**	27.95
	L	28.81	44.72	44.72	**47.82**	44.10
	M	23.52	33.77	33.28	**35.17**	34.72
BioASQ 5b	S	41.33	39.33	41.33	44.00	**46.00**
	L	56.67	52.67	55.33	56.67	**60.00**
	M	47.24	44.27	46.73	49.38	**51.64**
BioASQ 6b	S	24.22	33.54	**43.48**	40.37	42.86
	L	37.89	51.55	55.90	**57.77**	**57.77**
	M	27.84	40.88	48.11	47.48	**48.43**

Figura 1.10: In questa tabella sono riportati i risultati di BioBERT in task di question answering. Le metriche sono: Accuracy (S), Lenient Accuracy (L) e Mean Reciprocal Rank (M), le migliori sono in grassetto e le seconde migliori sono sottolineate. Wiki, Books, PubMed e PMC sono i dataset su cui i modelli sono stati addestrati (i primi due generici, gli altri due in ambito scientifico).

1.2.3 ViT

Nonostante i Transformers siano una tecnologia molto utilizzata nell'ambito dei testi, in quello delle immagini hanno avuto un utilizzo limitato. Per molto tempo, l'analisi delle immagini è stata condotta nel deep learning utilizzando le reti convoluzionali o *convolutional neural networks* (CNN), particolari tipi di reti neurali che applicano diversi filtri alle immagini cercando di catturarne il maggior numero di caratteristiche possibili. Spesso i meccanismi di attention, se presenti, vengono utilizzati solamente a supporto di questi particolari tipi di reti neurali. Questo però ha fatto si che la maggior parte dei modelli che lavorano con le immagini siano lenti e poco scalabili. Come è stato mostrato in precedenza, invece, i Transformers sono modelli più semplici che necessitano di un numero minore di calcoli, molti dei quali possono essere svolti parallelamente. ViT (Vision Transformer) [9] nasce infatti per sfruttare le caratteristiche dei Transformer nell'ambito delle immagini. In una prima fase le immagini di input vengono ridimensionate (ad un valore standard) e suddivise in *patch*, cioè in parti rettangolari. In particolare, data un immagine 2D $x_p \in \mathbb{R}^{H \times W \times C}$ dove H, W e C sono altezza, larghezza e canali, questa viene trasformata in una sequenza di "pezzetti" 2D $x_p \in \mathbb{R}^{N \times (P^2 C)}$, dove ogni patch misura P pixel per lato e $N = HW/P^2$ è il numero totale di patch, quindi anche la lunghezza dell'embedding finale. Fatto questo viene aggiunto un token speciale all'inizio, viene sommato un valore che indica la posizione delle singole patch e il risultato viene passato come input ad un Transformer encoder. Questo meccanismo permette un minor numero di operazioni sulle immagini, rendendo il modello molto veloce nei calcoli e mantenendo comunque ottimi risultati. I risultati

delle versioni di ViT Large: 24 layers, 1024 hidden states, 16 attention heads e ViT Huge: 32 layers, 1280 hidden states, 16 attention heads, sono riportati in figura 1.11.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Figura 1.11: Risultati rappresentati come media e varianza su 3 testset da ViT nella versione L (large) e H (huge) addestrati sul dataset JFT-300M o ImageNet-21k (I21k) contro lo stato dell'arte (BiT-K e Noisy Student). In grassetto i risultati migliori e in basso il costo computazionale del preaddestramento in giorni/core di una TPUv3

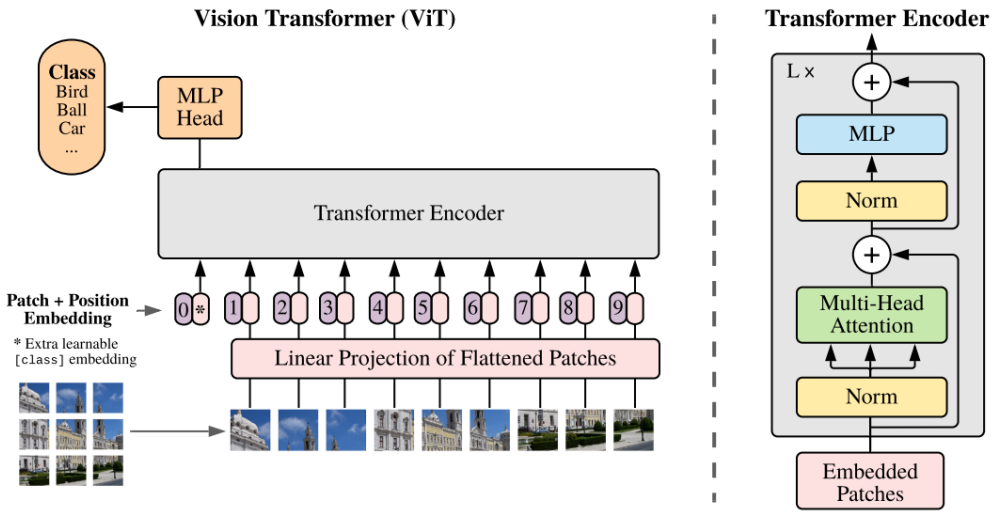


Figura 1.12: Schematizzazione del modello ViT. A destra la struttura del singolo encoder e a sinistra il meccanismo di embedding e di modellazione dell'output dal context prodotto dall'encoder (MLP layer).

ViT è stato addestrato nella classificazione dell'immagini tra K classi sia in preaddestramento che in fase di fine-tuning. Nel primo caso è stato usato un

multi-layered perceptron con uno stato nascosto (questo caso è quello illustrato dall'immagine 1.12), mentre nel secondo è stata utilizzata una FFNN, con un unico layer, che utilizza un algoritmo di classificazione lineare e ha forma $D \times K$, dove D è la dimentione del context creato dall'encoder.

1.3 Vision and language pre-training models

Il paradigma *pre-train-and-fine-tune* sta prendendo sempre più piede nell'ambito *vision and language*, cioè in tutti quei problemi che lavorano con immagini e/o testi come input. Questo paradigma consiste nell'addestrare inizialmente un modello su quantità di dati molto grandi utilizzando obiettivi generici, in modo che il modello raggiunga una "buona comprensione" del linguaggio e/o del contesto fornito dall'immagine (pre-training). Successivamente questi modelli pre-addestrati, che hanno una conoscenza generica, vengono impiegati per risolvere problemi specifici, per fare ciò si addestrano nuovamente su un dominio ridotto e con obiettivi specifici. In questo modo l'accuratezza delle risposte del modello aumentano e possano soddisfare quel determinato dominio applicativo. Il modello utilizzato in questo progetto è, infatti, preaddestrato su testi e immagini e prende ispirazione da BERT per la parte di text processing e da ViT per la parte di image processing.

1.3.1 ViLT

Dallo stesso obiettivo che ha guidato lo sviluppo di ViT (cioè ridurre il carico computazionale di un modello che utilizza immagini) nasce ViLT [10] un modello per l'encoding di input visivi e testuali. I modelli di questa categoria sono solitamente suddivisi in tre componenti principali: un *textual embedder*, che si occupa del processamento dei testi, un *visual embedder*, che si occupa del processamento delle immagini e un *modality interaction*, che unisce i due embedding e ne produce una rappresentazione congiunta. Molti dei modelli in questo ambito impiegano le tecnologie di miglior successo per entrambe le fasi di processamento, cioè semplici meccanismi di tokenizzazione ed encoding per i testi, come visto con BERT e più complesse reti convoluzionali per le immagini. Questo fa si, però, che la seconda parte "appesantisca" drasticamente il modello. Per questo motivo ViLT (Vision-and-Language Transformer) nasce con l'obiettivo di ridurre questi costi computazionali e, similmente a ViT, processa le immagini tramite una proiezione lineare delle patch dell'immagine, che poi unisce ai testi tokenizzati e li elabora con un Transformer encoder.

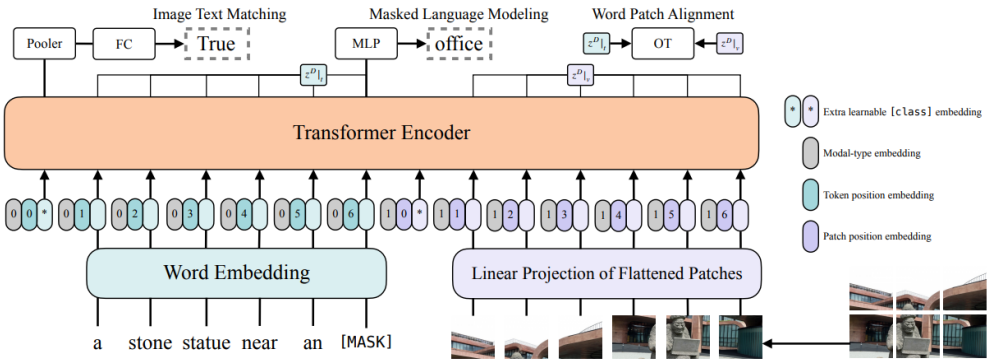


Figura 1.13: Schematizzazione del modello ViLT. Immagine e testo vengono processati e poi passati all'encoder come un unico input, poi l'output può essere elaborato in vari modi in base all'obiettivo.

Come si nota dall'immagine 1.13 vengono aggiunti dei token speciali per segnalare l'inizio dell'input e la separazione tra testo e immagine. Infine, come si vede più in alto l'output può essere elaborato in vari modi, per esempio, per l'addestramento di ViLT sono stati utilizzati obiettivi di *masked language modeling* e *image text matching*. Il primo è lo stesso utilizzato per l'addestramento di BERT, con una probabilità dal 15% vengono mascherate casualmente delle parole nel testo. Poi, grazie ad una rete neurale MLP a due layers viene restituito un output della forma $M \times N$, dove M corrisponde al numero di parole nascoste e N al numero di parole nel vocabolario, tali che i valori rappresentino la probabilità che ha ogni parola di essere quella mascherata. Il secondo obiettivo consiste nel dare in input al modello, data un'immagine del dataset, il testo che la descrive oppure un testo casuale del dataset con una probabilità del 50%. Un unico layer lineare si occupa poi di dare in output due punteggi: uno che rappresenta quanto immagine e testo siano correlati e un altro che rappresenta quanto non siano correlate. Tramite una softmax, per esempio la fungione sigmoidea, si possono trasformare poi questi punteggi in probabilità che i dati rientrino in una classe (correlati) piuttosto che nell'altra (non correlati). ViLT utilizza un encoder con 12 layer, 12 attention head, 768 hidden states e suddivide le immagini in patch di 32x32 pixel, è stato poi *fine-tuned*, cioè è stato specializzato per vari obiettivi e sono stati visti risultati competitivi, e in alcuni casi migliori, di altri modelli più complessi. Questi risultati, nonostante l'architettura minimale, sono ciò che hanno portato alla scelta di utilizzare ViLT come modello preaddestrato da cui partire.

Model	Time (ms)	Text Retrieval						Image Retrieval					
		Flickr30k (1K)			MSCOCO (5K)			Flickr30k (1K)			MSCOCO (5K)		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
w/o VLP SOTA	~900	67.4	90.3	95.8	50.4	82.2	90.0	48.6	77.7	85.2	38.6	69.3	80.4
ViLBERT-Base	~920	-	-	-	-	-	-	58.2	84.9	91.5	-	-	-
Unicoder-VL	~925	86.2	96.3	99.0	62.3	87.1	92.8	71.5	91.2	95.2	48.4	76.7	85.9
UNITER-Base	~900	85.9	97.1	98.8	64.4	87.4	93.1	72.5	92.4	96.1	50.3	78.5	87.2
OSCAR-Base†	~900	-	-	-	70.0	91.1	95.5	-	-	-	54.0	80.8	88.5
VinVL-Base†‡	~650	-	-	-	74.6	92.6	96.3	-	-	-	58.1	83.2	90.1
Pixel-BERT-X152	~160	87.0	98.9	99.5	63.6	87.5	93.6	71.5	92.1	95.8	50.1	77.6	86.2
Pixel-BERT-R50	~60	75.7	94.7	97.1	59.8	85.5	91.6	53.4	80.4	88.5	41.1	69.7	80.5
ViLT-B/32	~15	81.4	95.6	97.6	61.8	86.2	92.6	61.9	86.8	92.8	41.3	72.0	82.5
ViLT-B/32ⓐ	~15	83.7	97.2	98.1	62.9	87.1	92.7	62.2	87.6	93.2	42.6	72.8	83.4
ViLT-B/32ⓐ⊕	~15	83.5	96.7	98.6	61.5	86.3	92.7	64.4	88.7	93.8	42.7	72.9	83.1

Figura 1.14: Risultati ottenuti da ViLT fine-tuned nei task di text e image retrieval rispetto allo stato dell'arte. E' riportata, per ogni dataset (Flickr30k, MSCOCO), l'accuratezza sulla prima risposta (R@1), le prime 5 (R@5) o 10 (R@10). † indica l'utilizzo aggiuntivo dei dataset GQA, VQAv2, VG-QA in fase di pretraining, ‡ indica l'utilizzo aggiuntivo del dataset Open Images. ⓐ indica l'utilizzo di RandAugment, meccanismo che aggiunge al dataset versioni modificate delle stesse immagini e ⊕ indica un numero maggiore di step di pretraining.

1.4 Deep Metric Learning

Una tecnologia molto importante nell'ambito machine learning è il *deep metric learning*[11], che si occupa di determinare una "distanza" concettuale tra due oggetti. Questa distanza che viene calcolata rappresenta quanto questi due oggetti siano concettualmente simili, possiamo pensare, per esempio, a due immagini che rappresentano lo stesso soggetto con un'angolazione diversa, contro due immagini che rappresentano due soggetti diversi. Questo concetto ha avuto molti utilizzi in aree come il riconoscimento facciale o dell'impronta digitale, nella classificazione o anche nel retrieval di immagini. Questi algoritmi si basano sull'utilizzo di una rete neurale (*siamese network*), $f(x) : X \to \mathbb{R}^n$, che trasforma i dati grezzi, X, in rappresentazioni vettoriali e su di una funzione che calcola la distanza effettiva tra due di questi vettori, $D : \mathbb{R}^n \to \mathbb{R}$. Le tecniche di metric learning si basano quindi sullo stabilire una rete neurale e una funzione per calcolare la distanza che dati due input diano come output un valore piccolo, nel caso di due dati simili, e un valore alto nel caso siano molto differenti. Il primo approccio preso nel creare una funzione di loss è quello chiamato *contrastive*, che consiste nel cercare di "allontanare" le coppie di oggetti differenti e di "avvicinare quelle di oggetti simili. Per esempio, se si parla di *contrastive loss* vengono scelte coppie di oggetti casualmente e, in base al fatto che siano simili o meno, y prende il valore 1 o 0 nell'equazione 1.7, che calcola questa loss (dove D è definita dalla formula 1.6). L'iperparametro α,

presente in queste formule, viene chiamato margine e viene utilizzato perchè il modello non degeneri portando tutti i valori nello stesso punto, riducendo tutte le distanze a zero.

$$D(p,q) = \|p-q\|_2 = (\sum_{i=1}^{n}(p_i - q_i)^2)^{\frac{1}{2}} \tag{1.6}$$

$$L_{constrastive} = y \cdot D^2(f(x_1), f(x_2)) + y \cdot max(0, \alpha - D^2(f(x_1), f(x_2))) \tag{1.7}$$

Un'altra loss che utilizza questo approccio ed è molto utilizzata è la *triplet margin loss*[12], che prende in considerazione tre oggetti invece di due, in modo che due siano simili, questi vengono solitamente chiamati ancora e positivo (x_a e x_p), e il terzo sia diverso da loro, chiamato negativo (x_n). La struttura di quest'ultima viene mostrata in figura 1.15, mentre la formula utilizzata è la 1.8.

$$L_{triplet} = max(0, D^2(f(x_a), f(x_p)) - D^2(f(x_a), f(x_n)) + \alpha) \tag{1.8}$$

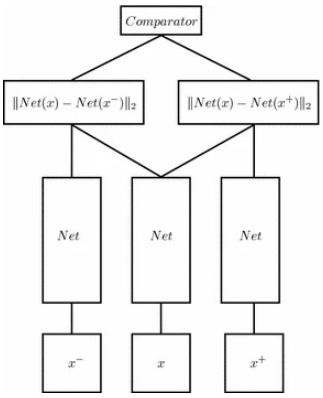


Figura 1.15: Viene illustrato il funzionamento della triplet margin loss dove, preso un dato X vengono selezionati altri due valori, uno simile e uno differente. Ognuno di questi dati viene elaborato dalla rete neurale e gli "embeddings" creati vengono poi confrontati calcolandone le distanze.

A seguito, sono state poi sviluppate loss differenti che utilizzano altri paradigmi, come la *multi similarity loss*[13] o l'*AP loss*[14]. La multi similarity loss, per esempio, nasce dall'idea intuitiva di considerare contemporaneamente tre diversi tipi di similarità: la *self similarity*, la *negative relative similarity* e la *positive relative similarity*. Il primo tipo di similarità è quello già affrontato nel caso della triplet margin loss, cioè dato un certo elemento ancora, un positivo e un negativo si cerca di allontanare il negativo dall'ancora e di

avvicinargli il positivo. La negative relative similarity si occupa invece di stabilire se la distanza tra un ancora e un negativo è "abbastanza" rispetto agli altri negativi. Mettiamoci per esempio nel caso in cui la maggior parte dei negativi si trova ad una distanza compresa tra 10 e 20 dall'ancora, mentre il negativo preso in considerazione si trova a distanza 5, in questo caso il modello dovrebbe necessitare delle modifiche perché quest'ultimo venga ulteriormente allontanato dall'ancora. La positive relative similarity utilizza lo stesso concetto ma prendendo in esame esempi positivi, in questo caso se un elemento positivo si trova troppo distante dall'ancora, rispetto agli altri positivi, deve essere "avvicinato".

Capitolo 2

Dataset

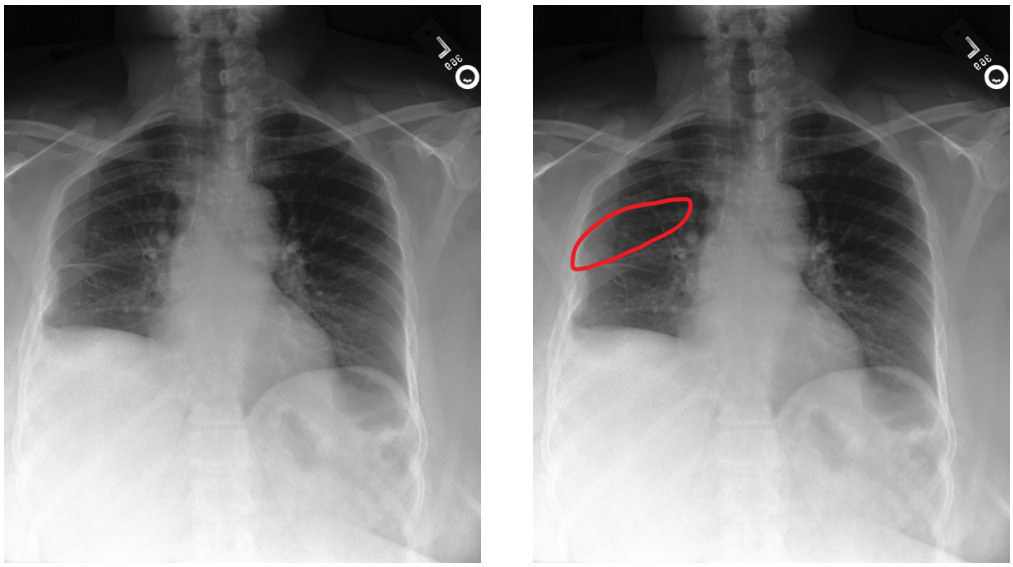
Dopo aver descritto l'insieme delle tecnologie disponibili, che costituiscono lo stato dell'arte nel loro ambito, si procede analizzando il dominio della problematica in questione.

2.1 Introduzione al dataset

Per questo progetto è stato utilizzato il dataset MIMIC Chest X-ray (MIMIC-CXR[15, 16, 17]) alla sua versione 2.0.0. Si tratta di un insieme di dati pubblici (e anonimizzati) riguardanti radiografie toraciche in formato DICOM (o jpg) e i relativi referti in formato testuale. La versione che è stata utilizzata utilizza le immagini in formato jpg.

Questo dataset è nato con l'idea che le radiografie toraciche siano ampiamente utilizzate in campo medico per diagnosticare problemi legati al torace e agli organi contenuti al suo interno. Il numero di radiologi negli Stati Uniti (dove è stato creato il dataset), però, è in calo e la loro distribuzione geografica è fortemente concentrata nelle grandi città. Per questo motivo, unito a ritardi burocratici e lunghe code di attesa, la qualità dell'apparato sanitario è in decrescita. Perciò, un sistema automatizzato, con alta accuratezza, che analizza le radiografie avrebbe un enorme potenziale nel facilitare il lavoro dei radiologi e nel migliorare il servizio sanitario, negli Stati Uniti, ma ancora di più in paesi in via di sviluppo.

Il dataset contiene 377,110 immagini corrispondenti a 227,835 diversi casi clinici condotti al Beth Israel Deaconess Medical Center in Boston, MA. Ogni dato è completamente anonimo per ragioni di privacy, ed è stato assegnato un numero identificativo ("subject id", compreso tra 10,000,000 e 19,999,999) ad ogni paziente, ad ogni caso di studio ("study id", compresto tra 50,000,000 e 59,999,999) e un codice hash di 40 caratteri ad ogni immagine. Nell'immagine 2.1 si può osservare un esempio di radiografia del dataset, che è accompagnata



(a) Radiografia originale (b) Costola rotta evidenziata

Figura 2.1: Una radiografia di esempio del dataset, la figura 2.1a, a sinistra, mostra la versione originale della radiografia e nella figura 2.1b, a destra, viene evidenziata la costola rotta.

dal referto:

"*FINAL REPORT: REASON FOR EXAMINATION: Evaluation of the patient with history of carcinoid with intermittent dyspnea on exertion. PA and lateral upright chest radiographs were reviewed in comparison to ___. Heart size is normal. Mediastinum is normal. The post-surgical changes in the right hemithorax are stable including thickening of the pleura along the costal surface and blunting of the costophrenic sulcus. The surgical fracture of the right sixth rib is redemonstrated. No new abnormalities are demonstrated within the limitations of the chest radiograph technique. Lung volumes are preserved.*".
In breve, il referto riporta che la radiografia è stata eseguita come controllo a seguito di una operazione per carcinoide (forma tumorale) che causava sforzi respiratori, ma al momento della radiografia è tutto in buono stato, anche la frattura della sesta costola destra (che è stata rotta per eseguire l'operazione). Come capiamo già dal semplice esempio nella figura 2.1b, anche un problema che teoricamente dovrebbe essere semplice da individuare ad occhio nudo, una costola rotta, risulta molto difficile da individuare per una persona senza esperienza nel campo.

MIMIC-CXR-JPG v2.0.0 contiene:

- Una cartella *files* che contiene 10 cartelle, contenenti ciascuna circa 6500

sotto cartelle corrispondenti ai pazienti. Ogni cartella contiene poi, per ogni visita un referto in formato txt e un'altra cartella contenente le immagini di quella radiografia.

- *mimic-cxr-2.0.0-metadata.csv.gz* - un documento in formato csv (compresso) contenente metadati come il momento dell'acquisizione (in forma anonima), posizione e orientazione del paziente durante la radiografia.
- *mimic-cxr-2.0.0-split.csv.gz* - un documento in formato csv (compresso) contenente la suddivisione in train, validation e test set suggerita.
- *mimic-cxr-2.0.0-chexpert.csv.gz* - un documento in formato csv (compresso) contenente dei label generati da CheXpert[18].
- *mimic-cxr-2.0.0-negbio.csv.gz* - un documento in formato csv (compresso) contenente dei label generati da NegBio[19].

```
files
└── p10
    └── p10000032
        ├── s50414267
        │   ├── 02aa804e-bde0afdd-112c0b34-7bc16630-4e384014.dcm
        │   └── 174413ec-4ec4c1f7-34ea26b7-c5f994f8-79ef1962.dcm
        ├── s50414267.txt
        ├── s53189527
        │   ├── 2a2277a9-b0ded155-c0de8eb9-c124d10e-82c5caab.dcm
        │   └── e084de3b-be89b11e-20fe3f9f-9c8d8dfe-4cfd202c.dcm
        ├── s53189527.txt
        ├── s53911762
        │   ├── 68b5c4b1-227d0485-9cc38c3f-7b84ab51-4b472714.dcm
        │   └── fffabebf-74fd3a1f-673b6b41-96ec0ac9-2ab69818.dcm
        ├── s53911762.txt
        ├── s56699142
        │   └── ea030e7a-2e3b1346-bc518786-7a8fd698-f673b44c.dcm
        └── s56699142.txt
```

Figura 2.2: Viene mostrata la gerarchia in cui sono suddivisi i files. La prima cartella indica le iniziali degli identificatori dei pazienti contenuti al suo interno (p10), la seconda ha come nome l'identificatore del paziente (p10000032) e l'ultima identifica le singole sedute (s50414267, s53189527, s53911762 e s56699142).

Nell'immagine 2.2 viene illustrata la struttura dei documenti all'interno della cartella *files*. In particolare, come prima cosa si nota la cartella *p10*

che è una delle 10 macro-cartelle in cui è suddiviso il dataset. All'interno di questa cartella si trovano tutti i pazienti con un identificatore che inizia per "10", nell'esempio il paziente *p10000032* ha eseguito quattro radiografie (con identificatori *s50414267*, *s53189527*, *s53911762* e *s56699142*) e per ciascuna di esse ci sono all'interno due immagini jpg.

2.2 Analisi del dataset

Ai veri e propri dati del dataset, come abbiamo visto, sono poi affiancati alcuni documenti che tengono traccia della correlazione tra immagini, studi e pazienti ed un file (presente in due versioni diverse:) che tiene traccia di alcuni label, cioè della presenza o meno di alcune parole (e insiemi di parole) chiave:

- *Atelectasis*: "atelectasia", mancata distensione di un organo.
- *Cardiomegaly*: "cardiomegalia", aumento di volume del muscolo cardiaco.
- *Consolidation*: rigonfiamento, indurimento di tessuti normalmente morbidi
- *Edema*: accumulo di liquidi.
- *Enlarged cardiomediastinum*: "ingrossamento del mediastino", strutta centrale del torace compresa tra i polmoni
- *Fracture*: "Frattura".
- *Lung lesion*: "lesione polmonare".
- *Lung opacity* : "opacità polmonare" o "nodulo" polmonare.
- *Pleural effusion*: "effusione pleurica", occupazione dello spazio pleurico (tra i polmoni e il torace) a causa di un eccesso di liquido.
- *Pneumonia*: "polmonite".
- *Pneumothorax*: "pneumotorace", presenza di aria nello spazio pleurico.
- *Pleural other*: "*altro* pleurico", altri problemi legati allo spazio pleurico.
- *Support devices*: "dispositivi medici".
- *No finding*: "nessun risultato", non è strato trovato nulla che si riferisse a problemi ne della lista, ne di un'altra lista più ampia.

Per ciascun record sono quindi riportati tutti i label appena elencati con un valore assegnato ad ogni parola che è stata incontrata in quel referto. I possibili valori che ogni label può assumere sono:

- Mancante, se la parola chiave non è stata citata nel referto.
- 1, se la parola è presente ed è citata in modo positivo ("è presente un grosso edema"), per cui dovrebbe essere visibile in almeno una delle immagini.
- 0, se la parola è presente ma è citata in modo negativo ("non vi sono segni di polmonite"), per cui ci si aspetta non sia presente in nessuna delle radiografie corrispondenti.
- -1, se il problema è menzionato in modo non sicuro ("la presenza di noduli non può essere valutata"), oppure se viene citato in modo ambiguo rispetto al linguaggio ("la situazione del polmone destro è stabile").

Questo ultimo documento è stato particolarmente utile nel definire quando un immagine e un testo sono correlati, poiché anche se questi non riguardano lo stesso studio possono presentare caratteristiche molto simili, o addirittura identiche.

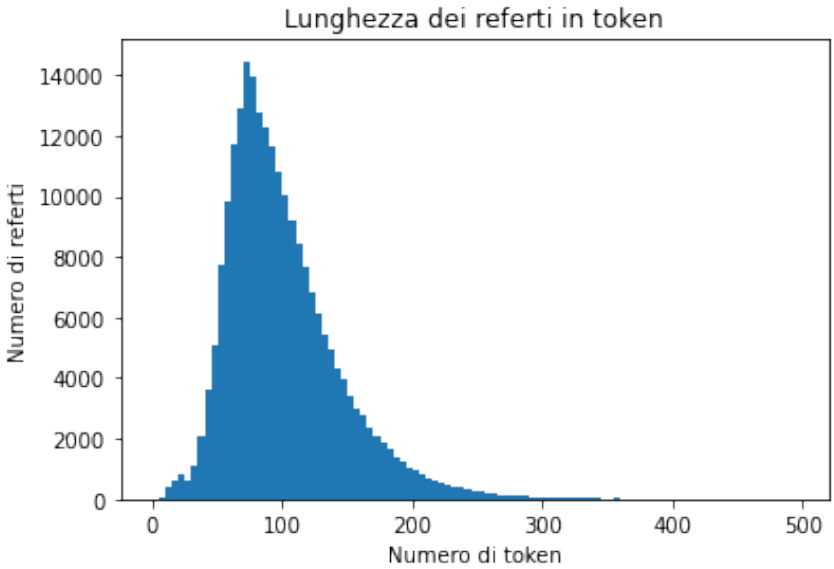
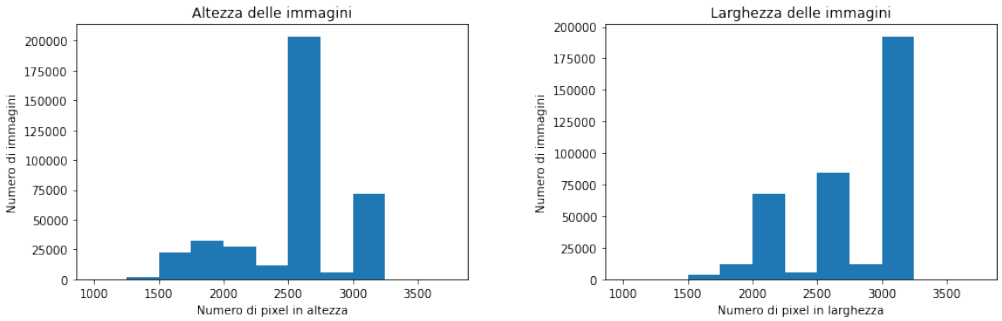


Figura 2.3: Istogramma che riporta la distribuzione della lunghezza dei referti in token (creati con il tokenizer di BERT).

Oltre al contenuto dei referti se ne è analizzata anche la lunghezza, che in media è sicuramente più alta di normali descrizioni di immagini, vista la



(a) Altezza delle immagini (b) Larghezza delle immagini

Figura 2.4: Vi sono nelle figure 2.4a e 2.4b mostrate rispettivamente la distribuzione dell'altezza e della larghezza delle immagini delle radiografie in pixel.

precisione con cui devono essere fatti dei referti medici. In particolare circa l'1% dei referti supera la soglia di 256 tokens, mentre soltanto 6 referti in tutto il dataset superano i 512 tokens. La stessa cosa si può dire anche per le immagini poiché, essendo radiografie toraciche, la risoluzione in media è molto alta. Come si vede in figura 2.4 la risoluzione media si aggira intorno ai 3000x2500 pixel.

Capitolo 3

Sviluppo

Si procede con la descrizione del lavoro svolto sul dataset e sullo sviluppo del nostro modello basato sulle tecnologie descritte nei capitoli precedenti.

3.1 Architettura del modello

Per risolvere il problema in questione abbiamo deciso di basarci su un modello ViLT preaddestrato che favorisce la semplicità e la "leggerezza" del modello al raggiungimento di accuratezze molto elevate. I vantaggi di ViLT, come è stato mostrato in precedenza, stanno infatti nell'utilizzo di un preprocessore particolarmente leggero, questo permette di rendere il modello molto più veloce di altri suoi concorrenti (che utilizzano reti convoluzionali per il preprocessing di immagini) sia in fase di addestramento che di predizione. In particolare, si è scelto di mantenere il meccanismo di preprocessing delle immagini invariato, si utilizza cioè un *ViLT feature extractor*, a sua volta ispirato a ViT [10] che ridimensiona le immagini a 384 pixel lungo il lato più corto e limita il lato più lungo ad un massimo di 640 preservando le proporzioni, poi le suddivide in patch di dimensione 32x32. Per il preprocessing dei testi si utilizza invece il tokenizer di BERT [3], come per ViLT, ma è stato deciso, vista la lunghezza elevata dei testi del nostro dataset, di aumentare la lunghezza massima da 40 a 512 tokens poiché circa l'1% dei referti supera questa soglia di default. Ci è stato possibile reperire direttamente il modello di ViLT originale preaddestrato in task di masked language modeling grazie ad Hugging Face [20][1], poi abbiamo modificato le configurazioni sopra citate, i task di addestramento e abbiamo fatto addestrato il modello utilizzando il nostro dataset. Si può vedere un'illustrazione della struttura del modello in figura 3.1.

[1] https://huggingface.co/dandelin/vilt-b32-mlm

3.1.1 Task di addestramento

Il modello, come visto in precedenza, accetta in input delle coppie testo-immagine e, grazie ad un transformer, produce come output un vettore context che può poi essere interpretato in vari modi in base al task. Nel nostro caso abbiamo deciso di utilizzare il meccanismo di masked language modeling già implementanto per ViLT (ViltMLMHead) e aggiungere un nostro task di image text matching. La "ViltMLMHead" è composta da: un layer lineare denso, cioè una rete neurale che prende in input H valori degli hidden states e ne restituisce altri H usando ogni valore di input per calcolare ciascuno dei valori di output, un layer non lineare, un layer di normalizzazione e infine un layer lineare che trasforma gli H valori in M valori, dove M è il numero delle parole del dizionario (questo per ogni parola mascherata).

Il primo task consiste, infatti, nel sostituire ogni token [MASK] presente all'interno del testo di input con uno dei token presenti nel dizionario del tokenizer. Di conseguenza l'output è una matrice $N \times M$ dove N è il numero di token mascherati, M il numero di parole del dizionario e i valori della matrice sono punteggi che indicano, dato un token mascherato, quanto è corretto sostituirlo con una data parola del dizionario. Su questi punteggi si potrebbe poi ottenere una vera e propria probabilità con una funzione di softmax.

Il secondo task, di image-text matching, consiste nel determinare quanto una data immagine e un dato testo siano correlati tra loro. In particolare, abbiamo utilizzato un layer lineare che dato in input un context di lunghezza H produce due valori che indicano rispettivamente quanto radiografia e referto siano correlati e quanto non lo siano. Grazie a questi due punteggi si può direttamente scegliere il più alto per capire la predizione fatta dal modello, oppure si può eseguire una softmax per ottenere, come prima, le probabilità.

Per produrre degli esempi su cui addestrare il modello non è quindi sufficiente utilizzare semplicemente i dati presenti nel dataset poiché questi formerebbero solo coppie "correlate". Inoltre i report testuali non presenterebbero nessun mascheramento . Per questo motivo abbiamo implementato due operazioni di preprocessing che vengono effettuate dinamicamente durante il caricamento dei dati:

- Mascheramento di ogni token in input con una probabilità del 15% (questo valore è stato scelto in quanto corrisponde a quello usato nel paper originale di BERT);
- Sampling di coppie negative associando in maniera casuale testi ed immagini dal dataset.

La scelta delle coppie non-correlate richiede una discussione più approfondita in quanto non è banale scegliere quando un testo ed una data immagine possono

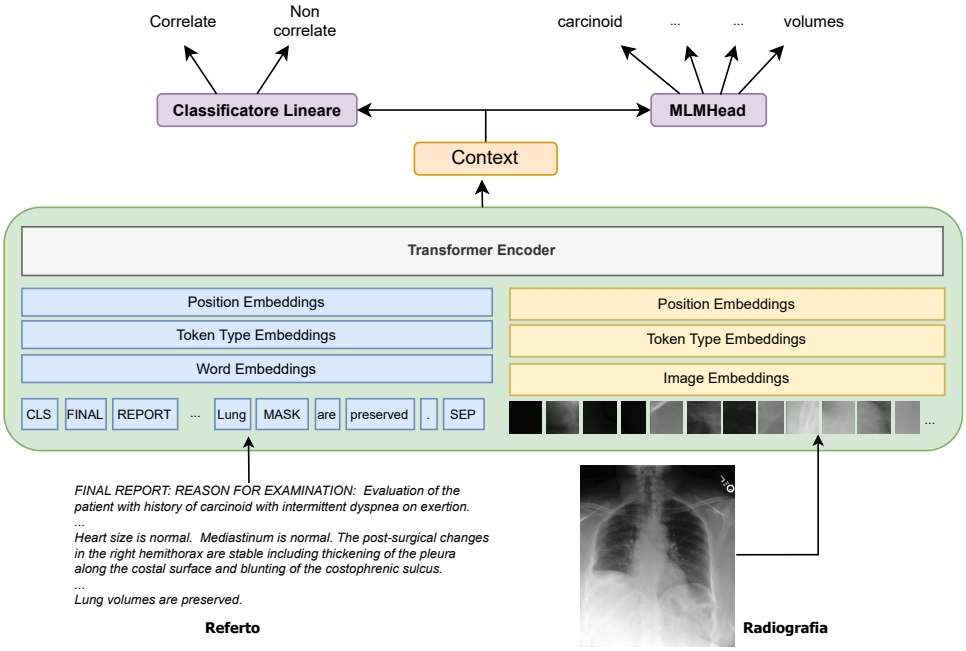


Figura 3.1: Viene illustrato il modello che, dato in input un referto e un'immagine, li suddivide rispettivamente in token e in patch, poi aggiunge dei valori relativi al loro tipo (token speciali o del testo, patch di padding o dell'immagine) e dei valori relativi alla posizione. Questi embeddings vengono poi elaborati da un transformer encoder che produce un vettore context. Il context viene poi elaborato da un'unità MLMHead per produrre delle predizioni sulle parole mascherate e da un classificatore lineare per decidere se gli input sono o meno correlati tra di loro.

essere considerate negative. La prima idea utilizzata è stata quella di dare in input al modello dei dati "corretti" (cioè immagine e testo correlati) nel 50% dei casi, mentre, nell'altro 50%, veniva fornito un testo del dataset e un'immagine estratta a caso, sempre dal dataset, che non corrispondesse a quel testo.

Questo approccio è il più semplice, però non tiene conto di alcuni casi particolari. E' infatti possibile che vengano "estratti" dal dataset un testo che descrive un dato problema (es. polmonite) e un'immagine in cui è presente quello stesso problema, ma in un paziente differente e quindi in una visita diversa. In questo caso particolare potrebbero quindi venire date al modello un testo e un immagine che noi consideriamo "corrette", correlate, ma che vengono etichettate come non correlate, perché relativi a due record diversi.

Per risolvere questo problema abbiamo implementato due varianti, entrambe che producono comunque 50% di dati in input positivi e 50% negativi, che non estraggono gli esempi negativi in maniera totalmente casuale, ma fanno delle considerazioni basate sui metadati visti nel capitolo di analisi del dataset (delle label che indicano quali problemi sono citati nel referto). In un caso, gli esempi non correlati richiedono che le visite corrispondenti al testo e all'immagine non abbiano alcuna label in comune, nell'altro che possono avere label in comune ma non devono essere tutte identiche.

Una volta implementato il funzionamento dei due task di masked language modeling e image text matching, è stato deciso di utilizzare come funzione di loss (da minimizzare) una somma delle *cross entropy loss* dei due task. La cross entropy loss dato un vettore x di lunghezza C, che contiene i punteggi assegnati ad ogni classe c e una classe da predire y_n è definita nella formula 3.1. Per aggregare la loss di più predizioni si è deciso di calcolarne la media.

$$l = -log(\frac{exp(x_{y_n})}{\sum_{c=1}^{C} exp(x_c)}) \tag{3.1}$$

3.1.2 Algoritmi di explainability

Una volta creato il modello, e fatto fine tuning sul dataset MIMIC-CXR, abbiamo deciso di applicare degli algoritmi di explainability per cercare di interpretare che cosa avesse appreso. In particolare, abbiamo deciso di utilizzare due tecniche diverse: *integrated gradients* [21] e *attention rollout* [22]. Il meccanismo degli integrated gradients è molto semplice e consiste nel considerare il modello utilizzato come una funzione F(x) dove $F : \mathbb{R}^N \Rightarrow R^M$. N è la dimensione dell'input, che nel nostro caso sarebbe uguale al numero dei token, sommato al numero delle patch dell'immagine, sommato agli altri token speciali di separazione; M indica la dimensione dell'output, 2, se consideriamo il task di image text matching. Data questa "funzione" si prende un certo input x

e un input "nullo" x', nel nostro caso un immagine completamente bianca e un vettore di token tutti uguali a zero. L'integrated gradient di x lungo una certa dimensione i è poi definito con la formula matematica 3.2. Non potendo utilizzare un calcolatore per quantificare esattamente un integrale la formula utilizzata nell'algoritmo è quella approssimata (3.3).

$$IntegratedGrads_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \tag{3.2}$$

$$IntegratedGrads_i^{approx}(x) = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m} \tag{3.3}$$

Grazie a questo procedimento si ottengono dei veri e propri punteggi per ogni token del testo e ogni patch dell'immagine che indicano quanto ciascuno abbia contribuito nel determinare il risultato. Questi valori vengono poi normalizzati e utilizzati in qualche tipo di visualizzazione.

Il secondo metodo utilizzato, di attention rollout, non si basa soltanto su input e output del modello ma sui valori di attention prodotti. Come visto in precedenza, infatti, l'attention dovrebbe essere una vera e propria indicazione dell'importanza di ciascun dato in input nel determinare un risultato. Il meccanismo di attention rollout consiste nel moltiplicare in modo ricorsivo le matrici di attention dal primo layer all'ultimo seguendo la formula 3.4.

$$\tilde{A}(l_i) = \begin{cases} A(l_i)\tilde{A}(l_{i+1}) & \text{if} \quad i > j \\ A(l_i) & \text{if} \quad i = j \end{cases} \tag{3.4}$$

Nell'algoritmo implementato sono state poi aggiunte alcune modifiche, per esempio non avendo un'unica matrice di attention per layer, ma un attention a 12 "teste" dobbiamo in qualche modo "fonderle" in un'unica matrice. Per fare questo, abbiamo provato prendendo una media dei valori, il massimo o il minimo. Oltre a questo abbiamo cercato di ridurre il rumore e azzerare valori molto bassi, per questo abbiamo aggiunto un valore di *discard ratio* che, ad ogni passo ricorsivo della moltiplicazione tra matrici, azzera una certa percentuale di valori che sono più bassi degli altri.

3.2 Retrieval con metric learning

Una volta ottenuto il modello specializzato nella classificazione di coppie radiografia-referto abbiamo deciso di utilizzare delle tecniche di metric learning per permettere di sviluppare un applicazione di retrieval. Con retrieval si intende la possibilità di presentare al modello una "query" che può essere un

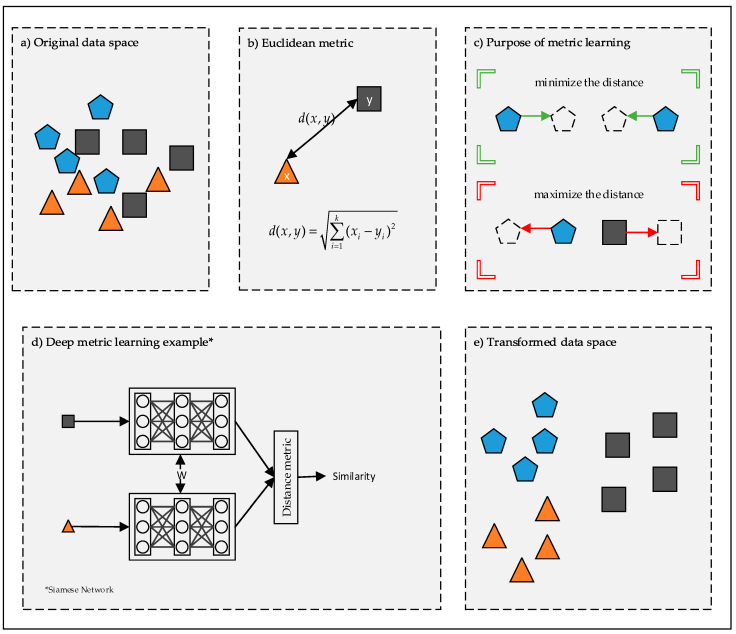


Figura 3.2: Viene illustrato visivamente il funzionamento del metric learning. Nella figura a si parte dagli embeddings generati dalla rete neurale, che hanno una certa distanza euclidea (figura b). L'obiettivo mostrato in figura c è quindi quello di creare una metrica che "allontana" dati diversi e "avvicina" quelli simili, fino ad arrivare al risultato mostrato in figura e.

referto o una radiografia di riferimento in modo tale che ci venga restituito un elenco rispettivamente di radiografie o di immagini ordinate per similarità a quella query. Per esempio, vorremmo che, idealmente, inserendo come input la query "tumore ai polmoni" ci vengano restituite prima una seria di radiografie che presentano chiaramente quella patologia, poi una serie di casi incerti e solo dopo i casi che invece non hanno nulla a che vedere con essa. Per fare questo il metric learning, solitamente, utilizza una coppia di reti neurali "siamesi" per determinare da due dati grezzi (nel nostro caso un'immagine e un testo) i loro embeddings e poi utilizza una funzione che calcola la distanza tra i due elementi di input. Una distanza maggiore indica che i due sono poco correlati, mentre una distanza minore che sono molto correlati. Questa idea è illustrata in figura 3.2.

Grazie al lavoro svolto precedentemente risulta semplice generare gli embeddings per immagini e testi direttamente dal modello addestrato in precedenza (senza dover addestrare nuove rete neurali), mentre la funzione per calcolare le distanze andrà creata ed addestrata da zero. Questo va fatto perché, nonostante gli embeddings possano essere già visti come punti in uno spazio multidimen-

sionale e risulti immediato calcolare la distanza tra due di questi punti, non è detto che gli embeddings generati dal nostro modello siano effettivamente già disposti vicini nel caso siano correlati e lontani nel caso non lo siano. Per lo sviluppo ci siamo appoggiati alla libreria di metric learning di PyTorch e abbiamo testato diverse funzioni di loss, con diversi iperparametri, per poi valutarle utilizzando la mean average precision at k. Questa metrica consiste nel calcolare, dato un valore k, quanti valori corretti sono stati restituiti tra i primi k elementi fratto il numero totale di elementi corretti presenti nella lista di elementi restituiti. Questo discorso verrà poi approfondito nella sezione 4.3.

3.3 Librerie utilizzate

In questa sezione viene fatto un elenco delle principali librerie utilizzate per realizzare questo progetto.

- Numpy [23] e Pandas [24], utilizzate per l'analisi del dataset, l'elaborazione dei metadati, quindi label e indici del dataset e per una piccola parte dei calcoli.
- PyTorch [25], utilizzata per la maggior parte delle operazioni tra matrici, per ottimizzare i processi su GPU e per gestire l'accesso al dataset.
- Hugging Face [20, 26], utilizzata come framework per reperire modelli preaddestrati ed eseguire fine tuning.
- PyTorch Metric learning [27], utilizzata per la parte di metric learning.
- Captum [28] utilizzato per la visualizzazione dei risultati ottenuti nel campo dell'explainability e per l'implementazione dell'algoritmo Layer Integrated Gradients.
- Weights & Biases[2], utilizzata per tenere traccia di tutti i test fatti, degli iperparametri e dei risultati ottenuti nei task di addestramento.

[2]https://wandb.ai/site

Capitolo 4

Esperimenti

Gli esperimenti sono stati condotti su di un server avente 48 processori, 128GB di RAM e 4 GPU Nvidia GeForce RTX 3090. Nello specifico la maggior parte dei calcoli per l'addestramento del modello sono stati fatti utilizzando una di queste GPU, che conta 24GB di memoria, 1695MHz come frequenza di clock e 10496 CUDA cores, unità fondamentale per parallelizzare i calcoli. Per motivi di occupazione di risorse è stato deciso di non utilizzare l'intero dataset MIMIC-CXR (più di 800GB), ma di utilizzarne inizialmente il 10% per testare alcuni iperparametri e successivamente il 50% per addestrare il modello.

4.1 Addestramento del modello

Per l'addestramento del modello sul nostro dataset (quindi fine tuning) è stato prima valutato l'impatto di alcuni iperparametri su un sottoinsieme del dataset. Dai risultati ottenuti in questa fase preliminare abbiamo deciso di procedere utilizzando i seguenti iperparametri:

- *learning rate* $= 2{\cdot}10^{-5}$, poiché altri valori più alti rendevano la diminuzione della loss meno regolare.
- *weigth decay* $= 1 \cdot 10^{-4}$, con una diminuzione del learning rate sinusoidale, in modo che i parametri del modello si stabilizzino verso la fine del training.
- 2500 step di *warmup* $= 1 \cdot 10^{-4}$, in modo che i parametri appresi dal modello in pretraining non venissero subito stravolti dai nuovi dati.
- *max position embeddings* $= 512$, perché come già detto i referti tendono ad essere molto lunghi.

- *adam* $\beta_1 = 0.9$ e *adam* $\beta_2 = 0.999$, iperparametri dell'algoritmo di ottimizzazione AdamW che utilizza il trainer di HuggingFace utilizzato (lasciati al valore di default).

- *batch size* = 20, per permettere la massima parallelizzazione dell'addestramento sulla GPU descritta in precedenza.

- *seed* = 42, per permettere la riproducibilità degli esperimenti.

Con questi iperparametri si è scelto di addestrare 3 modelli differenti che variano per il meccanismo che utilizzano nel selezionare coppie immagine-testo non correlate. In particolare, un modello seleziona questi esempi "negativi" in modo del tutto casuale (ViLT-R), un altro sceglie come coppie negative soltanto radiografie e referti che non hanno label identici (ViLT-NE) e l'ultimo seleziona come coppie negative soltanto quelle che non hanno alcun label in comune (ViLT-NI). Questi modelli sono stati addestrati per 5 epoche, cioè per poco più di 3 giorni nell'ambiente in cui abbiamo lavorato e l'andamento della loro funzione di loss è molto simile ed è descritto nell'immagine 4.1. Come si può vedere la funzione è molto ripida all'inizio (probabilmente a causa del dominio applicativo molto diverso rispetto al preaddestramento) e una volta diminuita tende a stabilizzarsi verso la fine dell'addestramento. Questo fa pensare che senza altri dati e senza modificare i modelli non si possano raggiungere performance migliori.

I valori di accuratezza ottenuti nel validation set (1496 record) per i due task di addestramento sono riportate in tabella 4.1.

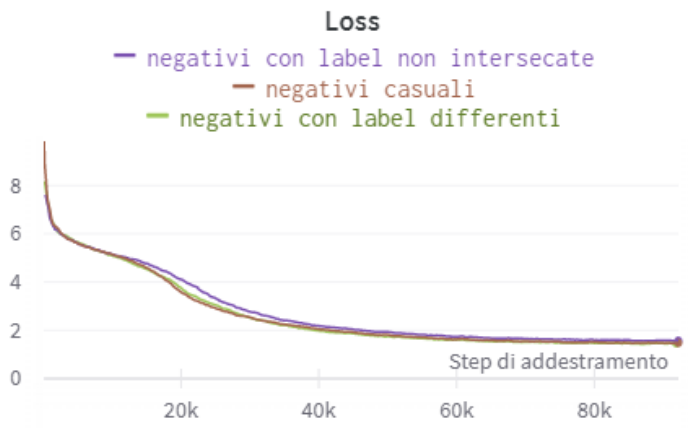


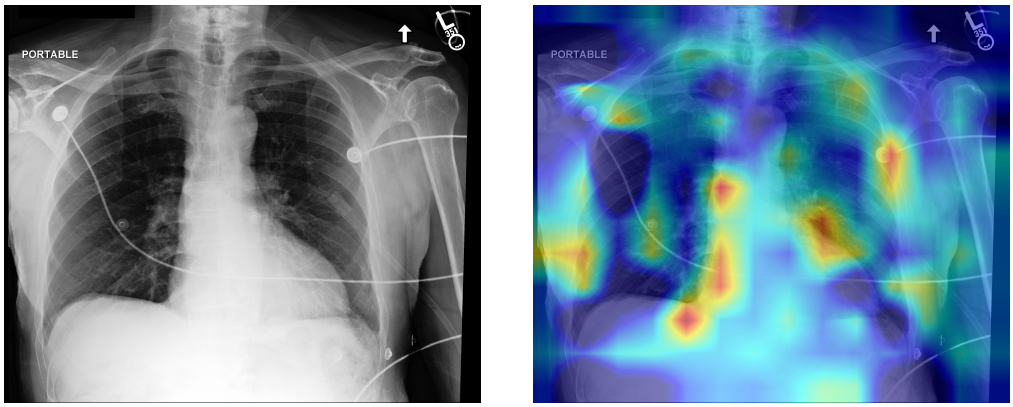
Figura 4.1: Nel grafico viene mostrata la minimizzazione della funzione di loss dei modelli, che risulta molto simile in tutti e tre i casi e tende a stabilizzarsi.

Modelli	MLM Acc	ITM Acc	Loss
ViLT-R	3.206	87.07	1.493
ViLT-NE	3.220	86.66	1.468
ViLT-NI	3.134	86.30	1.603

Tabella 4.1: In questa tabella vengono indicate l'accuratezza nei task di Masked Language Modeling e di Image Text Matching e la loss di ciascun modello. In ordine i modelli utilizzano esempi negativi: casuali, con almeno una label differente e con ogni label differente.

4.2 Risultati di explainability

Come anticipato nella sottosezione 3.1.2 abbiamo svolto anche un lavoro di explainability sul modello addestrato. In particolare, si è cercato di evidenziare sia nelle radiografie che nei referti le parti più importanti e significative, in modo da poter poi interpretare e "giustificare" le decisioni prese dall'algoritmo, per esempio nel classificare una coppia immagine-testo come correlata. Per questo scopo sono stati utilizzati principalmente due soluzioni basate su due algoritmi di base: i *layer integrated gradients* e l'*attention rollout*. La prima soluzione che utilizza un algoritmo più semplice e diretto, ha portato ad alcuni risultati interessanti sia nell'interpretazione delle immagini che del testo, seppur non fosse sempre affidabile e preciso. Mostriamo nell'immagine 4.2 un buon esempio e nell'immagine 4.4 un altro esempio meno buono che infatti evidenzia anche parti dell'immagine o del testo senza significato, come l'angolo della radiografia, in cui non si trova nulla o parole poco rilevanti come "indicazioni".



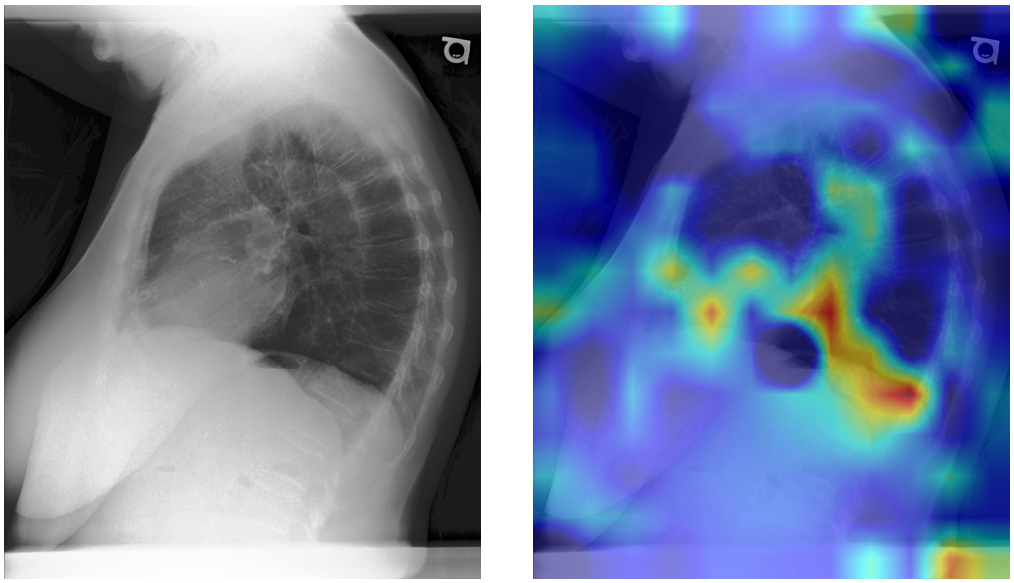
(a) Radiografia originale (b) Radiografia con parti evidenziate

Figura 4.2: Come si può vedere l'algoritmo di explainability non è molto preciso, ma comunque riesce ad evidenziare come parti più importanti della radiografia (in rosso) i contorni dei polmoni e del cuore. Leggendo la radiografia corrispondente, mostrata in figura 4.3, capiamo infatti che il paziente non presenta particolari anomalie poiché gli organi hanno un volume corretto.

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
Related	**Related (1.00)**	**Related**	**3.63**	[CLS] final report indication : _ _ _ m with am ##s / / acute process ? technique : portable ap upright view of the chest comparison : none findings : heart is normal size and card ##iom ##ed ##ias ##tina ##l con ##tour ##s are within normal limits . lungs are symmetrical ##ly expanded and clear . there is no pl ##eur ##al e ##ff ##usion or p ##ne ##um ##otho ##ra ##x . no pulmonary ed ##ema . impression : no acute intra ##thor ##ac ##ic process . [SEP]

Figura 4.3: Come si legge dal referto cuore e polmoni sono delle giuste dimensioni e sono sani. Il modello, oltre ad aver correttamente classificato radiografia e referto correlate, evidenzia nel referto le parole più significative per la decisione (in verde): "dimensioni normali", "polmoni", "cuore", "chiari e allargati".



(a) Radiografia originale (b) Radiografia con parti evidenziate

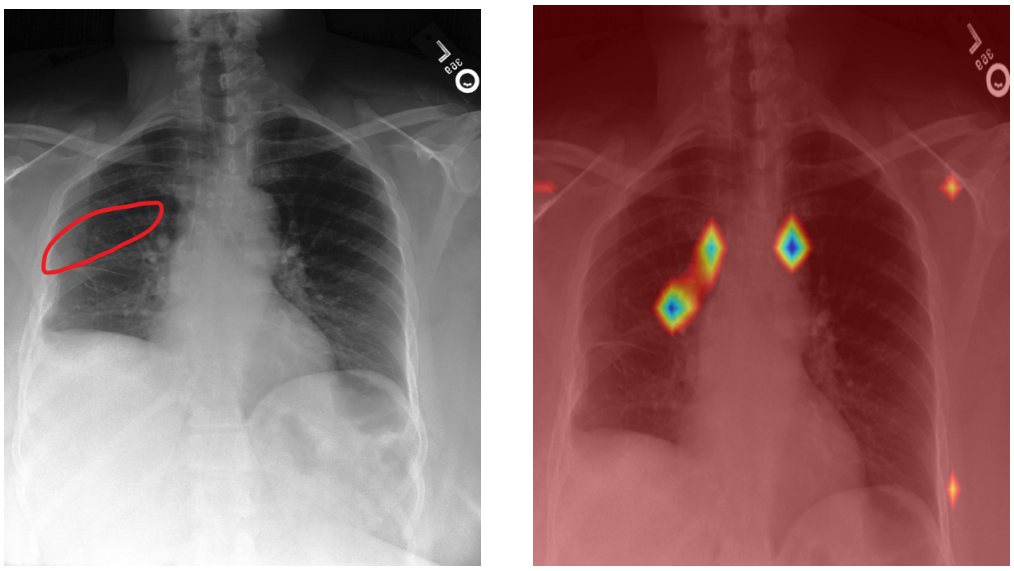
Figura 4.4: Come si può vedere l'algoritmo di explainability non è molto preciso nemmeno qui, in più vengono evidenziate parti di discutibile rilevanza, per esempio l'angolo dell'immagine che non contiene nulla.

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
Related	**Not related (0.00)**	**Related**	**1.03**	[CLS] final report examination : chest radio ##graph . indication : history : _ _ _ f with cough and back pain / / evaluate for pneumonia technique : chest pa and lateral comparison : comparison is made to chest radio ##graphs dated _ _ _ . findings : as compared to the prior examination dated _ _ _ , there has been no significant interval change . there is no evidence of focal consolidation , pl ##eur ##al e ##ff ##usion , p ##ne ##um ##otho ##ra ##x , or frank pulmonary ed ##ema . the card ##iom ##ed ##ias ##tina ##l silhouette is within normal limits . there is persistent thor ##ac ##ic ky ##ph ##osis with mild wed ##ging of a mid thor ##ac ##ic ve ##rte ##bra ##l body . impression : no evidence of acute card ##io ##pu ##lm ##ona ##ry process . [SEP]

Figura 4.5: Questo referto, per esempio, non viene analizzato correttamente dal modello, in quanto vengono evidenziate parole come "indicazioni" o degli spazi vuoti nel referto (creati dalla censura del nome del paziente). E anche la predizione fatta "not related" non è corretta.

Successivamente all'algoritmo basato su layer integrated gradients ne abbiamo testato uno basato sull'attention rollout, che si basa appunto sui valori di attention calcolati dal modello. In questo caso abbiamo raggiunto grafici più precisi per le immagini (dominio in cui questa tegnologia era effettivamente stata sviluppata), mentre per i testi non siamo riusciti a produrre risultati



(a) Radiografia originale (b) Radiografia con parti evidenziate

Figura 4.6: Nell'immagine 4.6a mostriamo una radiografia evidenziando manualmente la rottura di una costola, indicata anche sul referto. Nell'immagine 4.6b è mostrata la stessa radiografia analizzata con l'algoritmo di attention rollout che in blu evidenzia le parti rilevanti dell'immagine. Anche in questo caso l'attenzione è rivolta alla sesta costola nella parte destra del torace.

soddisfacenti. Un esempio di questo è mostrato in figura 4.6 utilizzando la radiografica di esempio che veniva analizzata nel capitolo 2.

Dai risultati ottenuti in questo elaborato, ma in generale in ambito scientifico, possiamo dire che le tecniche di explainability fino ad ora implementate non producono sempre risultati affidabili. Per questo motivo, ci aspettiamo che in futuro in questo campo possano esserci nuovi algoritmi più affidabili e che anche questo lavoro possa essere migliorato sotto questo aspetto.

4.3 Risultati del retrieval

Successivamente all'addestramento del modello sono stati svolti diversi test con l'utilizzo del metric learning per poter utilizzare il modello in task di *image-to-text* e *text-to-image retrieval*. In particolare, il task che abbiamo utilizzato consiste nel selezionare casualmente un testo come query e 100 immagini che non corrispondono a quel testo; poi viene aggiunta l'immagine che invece corrisponde alla query e, dato in input il testo al modello e questo insieme di immagini verranno ordinate dalla più simile alla meno simile, secondo il modello

(text-to-image). La stessa cosa è stata fatta utilizzando delle radiografie come query e dei referti come documenti da ordinare per similarità (image-to-text). Questo test è stato fatto sul dataset di validation, è stato ripetuto 1000 volte e sono state calcolate la precisione a 1, 5, 10, 30, 50, 100. Cioè, è stato calcolato quante volte l'immagine "esatta" è stata riportata rispettivamente come prima, tra le prime cinque, tra le prime dieci eccetera, sui 1000 casi estratti. Per questo obiettivo sono stati valutati diversi tipi di loss: la *multisimilarity loss*, la *triplet margin loss* e la *fast AP loss*, con diversi iperparametri. Per la multisimilarity loss sono stati provati diversi valori di *alpha* e *beta*, per la triplet margin loss sono stati presi diversi gruppi di triple dal miner, mentre per la fast AP loss sono stati usati i parametri di default. I risultati migliori per ciascuna loss sono riportati nelle tabelle 4.2 e 4.3.

Retrieval da testo ad immagine

Loss	P@1	P@5	P@10	P@30	P@50	P@100
MultiSim	0.052	0.23	0.384	0.804	0.95	1.0
Triplet	0.374	0.743	0.887	0.983	0.998	1.0
Fast AP	0.009	0.053	0.096	0.287	0.488	0.989

Tabella 4.2: In questa tabella vengono indicate l'accuratezza at k nel task di retrieval txt2img (quindi da testo ad immagine) utilizzando diverse loss.

Retrieval da immagine a testo

Loss	P@1	P@5	P@10	P@30	P@50	P@100
MultiSim	0.058	0.23	0.403	0.805	0.951	1.0
Triplet	0.387	0.743	0.88	0.987	0.998	1.0
Fast AP	0.015	0.053	0.097	0.31	0.504	0.991

Tabella 4.3: In questa tabella vengono indicate l'accuratezza at k nel task di retrieval img2txt (quindi da immagine a testo) utilizzando diverse loss.

Dalle tabelle possiamo notare che i risultati ottenuti utilizzando la triplet margin loss sono molto buoni, mentre quelli ottenuti utilizzando le altre loss rispetto a quest'ultima risultano molto bassi, relativamente a quest'ultima. Data questa discrepanza dei risultati ottenuti con le diverse metriche andrebbe fatta un'analisi più approfondita degli iperparametri utilizzati.

Capitolo 5

Applicativo Web

A scopo dimostrativo è stato realizzato un applicativo web (utilizzabile all'indirizzo https://disi-unibo-nlp.github.io/projects/radiography_retrieval/) che permette di provare il modello da noi addestrato sul dataset. In una pagina è possibile fare manualmente il test che abbiamo svolto per valutare l'accuratezza del modello in un task di retrival. Come si vede in figura 5.1 è infatti possibile selezionare un dato di esempio, in automatico al modello vengono date 99 radiografie casuali oltre a quella corretta e vengono mostrate nell'ordine stabilito dal modello (vedi 5.2). A questo punto la radiografia corretta è evidenziata in verde, in modo da potere più facilmente capire come è stata valutata.

Figura 5.1: Viene mostrato un possibile record, dove il referto viene usato come input e la radiografia è il risultato atteso.

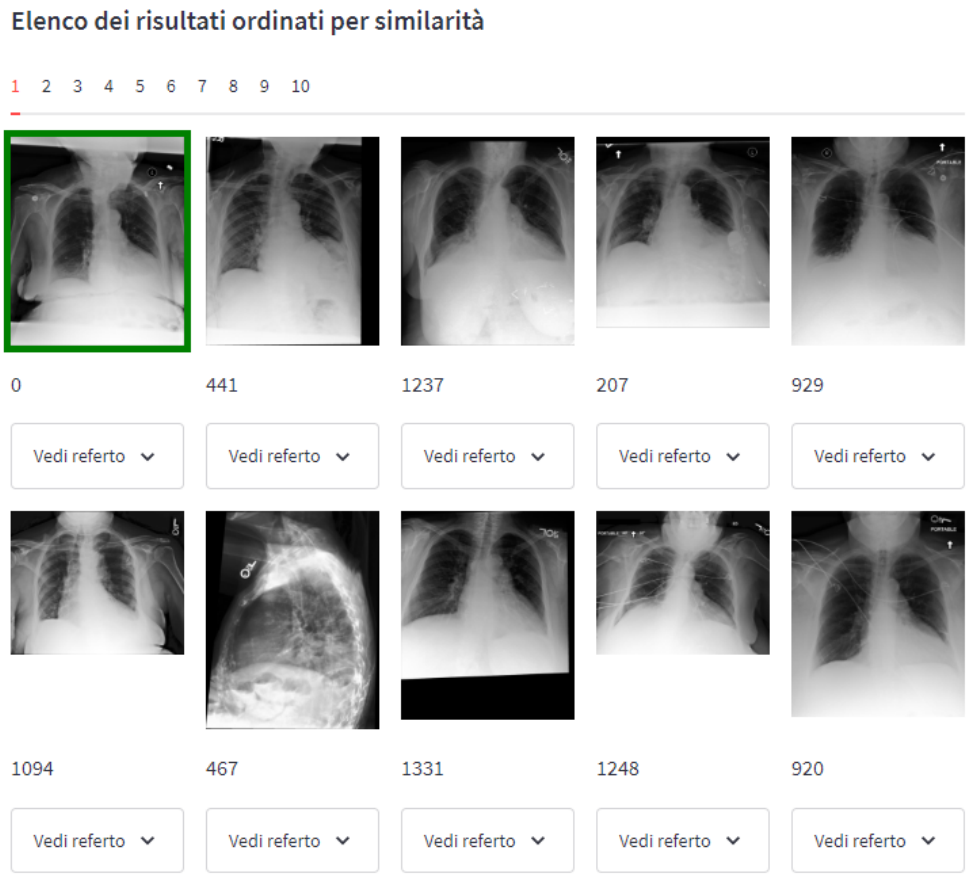


Figura 5.2: Viene mostrato l'output del modello suddiviso in pagine che riporta le radiografie ordinate per similarità. In questo caso la radiografia attesa è stata restituita come primo risultato.

Nella seconda pagina, quella principale, è stata invece implementata una possibile applicazione per utente finale in cui è possibile inserire un testo arbitrario. Il modello, a questo punto, va a valutare tutte le radiografie a nostra disposizione per trovare le 100 radiografie più inerenti al testo inserito che vengono poi mostrate in ordine di rilevanza. Una ricerca di esempio è mostrata in figura 5.3 e l'output restituito in figura 5.4.

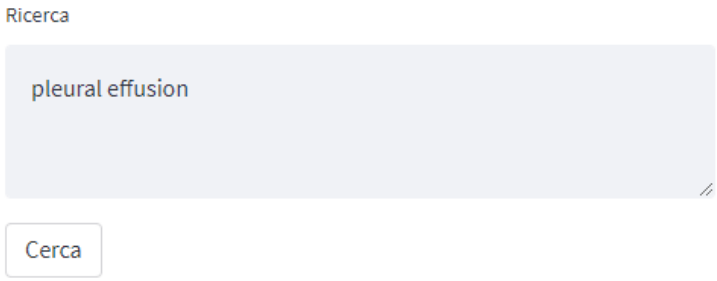


Figura 5.3: Viene mostrato il campo di input con una ricerca di esempio.

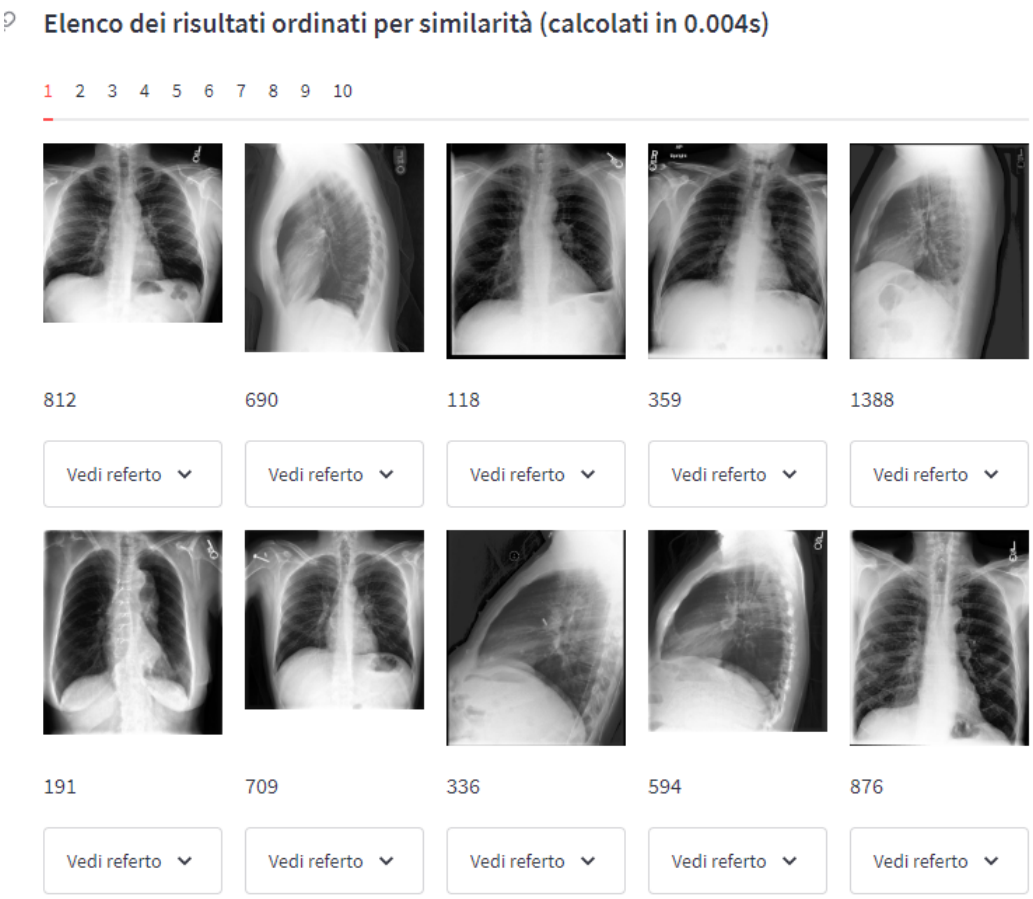


Figura 5.4: Viene mostrato l'output del modello suddiviso in pagine che riporta le 100 radiografie più coerenti alla ricerca "pleural effusion".

Conclusioni e sviluppi futuri

L'obiettivo di questo elaborato era quello di sviluppare un modello di machine learning in ambito biomedico capace di elaborare testi ed immagini relativi a radiografie toraciche e referti stilati da medici professionisti. E' stata quindi implementata una soluzione di retrieval cross-modale in grado di restituire le immagini di radiografie più pertinenti rispetto ad una qualsiasi query testuale e viceversa. Inoltre, sono state analizzate le principali tecniche di explainability in letteratura per poter spiegare e descrivere il comportamento del modello ottenuto. La soluzione sviluppata sfrutta un modello di tipo Vision-Language Transaformer basato sull'architettura di ViLT[10]. I risultati ottenuti, in fase di addestramento, in task di image-text matching, raggiungono l'87% di precisione, mentre nel retrieval raggiungono il 37% sul primo risultato in una query con 100 elementi errati e uno corretto, il 74% sui primi cinque e l'89% sui primi dieci. Quantitativamente i risultati ottenuti dal modello sono perciò molto buoni e grazie al metric learning ed all'utilizzo di rappresentazioni latenti abbiamo un sistema efficiente che permette di eseguire delle query su moltissimi dati in pochi millisecondi. Abbiamo creato una web app che sfrutta il modello e mostra le sue potenzialità in un task di retrieval (utilizzabile all'indirizzo https://disi-unibo-nlp.github.io/projects/radiography_retrieval/). Oltre a questo, i risultati degli algoritmi di explainability del modello sono interessanti, pongono le basi per qualcosa che potrà essere molto utile in ambito medico seppur al momento siano ancora instabili e imprecisi. Questo ambito è, infatti, il primo su cui è chiaro che vadano fatti ulteriori ricerche in ambito scientifico e dove gli applicativi sviluppati devono essere migliorati soprattutto dal punto di vista della spiegabilità e della trasparenza dei modelli. Per quanto riguarda il modello addestrato, si potrebbero sviluppare delle migliorie per esempio prendendo in maggior considerazione i metadati forniti dal dataset, cioè una lista di patologie che possono essere citate all'interno dei singoli referti. Questo potrebbe aiutare sia nel migliorare le performance del modello nella classificazione di coppie simili, ma soprattutto in task di retrieval per comprendere meglio quali siano le somiglianze tra radiografie diverse e quanto queste siano simili.

Ringraziamenti

Il primo ringraziamento va al relatore di questa tesi, il Prof. Gianluca Moro, che si è reso disponibile nella sua realizzazione e, prima di tutto, ha acceso il mio interesse nei confronti di questa disciplina.

Grazie a Stefano che mi ha seguito nello sviluppo del progetto da vicino.

E infine, grazie alla mia famiglia, ai miei amici e a Lucia per essermi stati accanto durante questo viaggio.

Bibliografia

[1] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics, 2015.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[4] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: first results. *CoRR*, abs/1412.1602, 2014.

[5] Pongchanun Luangpaiboon and Keerati Chinda. Computer-based management of interactive data transformation systems using taguchi's robust parameter design. *Int. J. Comput. Integr. Manuf.*, 28(10):1030–1045, 2015.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised

learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[8] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240, 2020.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[10] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5583–5594. PMLR, 2021.

[11] Mahmut Kaya and Hasan Sakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[12] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In Aasa Feragen, Marcello Pelillo, and Marco Loog, editors, *Similarity-Based Pattern Recognition - Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015, Proceedings*, volume 9370 of *Lecture Notes in Computer Science*, pages 84–92. Springer, 2015.

[13] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5022–5030. Computer Vision Foundation / IEEE, 2019.

[14] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. Ap-loss for accurate one-stage object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(11):3782–3798, 2021.

[15] Alistair E. W. Johnson, Tom J. Pollard, Seth J. Berkowitz, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih-ying Deng, Roger G. Mark, and Steven Horng. MIMIC-CXR: A large publicly available database of labeled chest radiographs. *CoRR*, abs/1901.07042, 2019.

[16] Alistair E. W. Johnson, Tom J. Pollard, Seth J. Berkowitz, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih-ying Deng, Roger G. Mark, and Steven Horng. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific Data*, 6(1):317, Dec 2019.

[17] Isaac C. Henry, Ary L. Goldberger, George B. Moody, and Roger G. Mark. Physionet: An NIH research resource for physiologic datasets and open source software. In *14th IEEE Symposium on Computer-Based Medical Systems (CBMS 2001), 26-27 July 2001, Bethesda, MD, USA*, page 245. IEEE Computer Society, 2001.

[18] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn L. Ball, Katie S. Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 590–597. AAAI Press, 2019.

[19] Yifan Peng, Xiaosong Wang, Le Lu, Mohammadhadi Bagheri, Ronald M. Summers, and Zhiyong Lu. Negbio: a high-performance tool for negation and uncertainty detection in radiology reports. *CoRR*, abs/1712.05898, 2017.

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art

Natural Language Processing. pages 38–45. Association for Computational Linguistics, 10 2020.

[21] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017.

[22] Samira Abnar and Willem H. Zuidema. Quantifying attention flow in transformers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4190–4197. Association for Computational Linguistics, 2020.

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[24] The pandas development team. pandas-dev/pandas: Pandas.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[26] Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Mario Šaško, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. Datasets: A Community Library for Natural Language

Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184. Association for Computational Linguistics, 11 2021.

[27] Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. Pytorch metric learning. *ArXiv*, abs/2008.09164, 2020.

[28] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.