UNIVERSITÀ DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Studio e implementazione di Vision-Language Models per il dataset KvasirVQA

TESI DI LAUREA DI
EMANUELE MUZIO

RELATORE
Prof. MARCO LA CASCIA

CONTRORELATORE
Prof. MARCO MORANA

ANNO ACCADEMICO 2023 – 2024

Indice

Capitolo 1

Introduzione

1.1 Contesto

Sin dall'introduzione delle *Convolutional Neural Network* (CNN) e delle architetture basate sui *Transformers*, il modo di affrontare sia i classici task di *Computer Vision* (CV) che di *Natural Language Processing* (NLP) ha subito pesanti rivoluzioni e compiuto notevoli passi in avanti.

Da un lato, infatti, le CNN si sono affermate soprattutto in task di CV come l'*Optical Character Recognition* (OCR), classificazione di immagini, e *Object Detection*, grazie alla loro capacità di catturare le caratteristiche salienti delle immagini senza necessità di feature engineering manuale. Dall'altro, le architetture basate sui transformers hanno rivoluzionato il modo di vedere l'elaborazione testuale grazie a un complesso sistema di *self-attention* che elabora le parole in parallelo piuttosto che sequenzialmente, come le classiche reti *Long Short-Term Memory* (LSTM).

La combinazione delle CNN per l'estrazione di caratteristiche visive e dei transformers per l'elaborazione del linguaggio naturale ha consentito lo sviluppo di modelli multimodali come quelli utilizzati nel *Visual Question Answering* (VQA), un task a metà strada tra i campi di CV ed NLP. I modelli che si occupano di affrontare questo particolare tipo di task si pongono in una più ampia famiglia di modelli multimodali, in particolare nel sottogruppo dei *Vision-Language Models* (VLM).

1.2 Storia del VQA

Il campo di ricerca relativo al VQA è di recente introduzione e in appena una decina di anni è riuscito a fare progressi notevoli grazie allo sfruttamento di tecnologie e approcci trasversali al problema.

I primi passi fatti per la ricerca si pongono come principio cardine la capacità di creare modelli che sappiano ragionare di fronte a domande su immagini, la cui risposta attinga a informazioni contenute nell'immagine stessa, senza necessità di utilizzare una *Knowledge Base* (KB) esterna e focalizzandosi sull'importanza di andare a ricercare le

informazioni necessarie per rispondere all'interno dell'immagine.

Parallelamente, vengono ovviamente compiuti diversi sforzi per raccogliere e formare dei dataset adeguati, come il *Visual Genome* [15] o il dataset rilasciato insieme al primo paper di ricerca sull'argomento [4] che porta lo stesso nome del suo task di riferimento. Un approfondimento sui dataset che hanno acquisito particolare rilevanza per il task verrà fatto nel capitolo della tesi dedicato ai dataset.

A livello tecnico, in particolare, troviamo largo uso inizialmente di approcci quantomeno simili per la fusione multimodale e la comprensione delle informazioni presenti nell'immagine. Introduciamo adesso brevemente alcune tecniche di risoluzione del problema che, seppur superate o rinnovate con ulteriori innovazioni in seguito, hanno costituito al tempo il fondamento della comprensione contestuale di immagini e testo.

1.2.1 Combinazione LSTM e CNN

Viene delineata una prima baseline in [4] che utilizza un per la parte testuale reti LSTM e una rete VGG [28] per le immagini, ottenendo risultati preliminari adeguati per il task. In particolare, viene adottata una fusione multimodale che combina la codifica del testo con l'output dell'ultimo layer nascosto della VGG. Gli embedding risultanti vengono quindi fusi tramite prodotto vettoriale, per poi passare infine da una piccola rete neurale che si occupa di effettuare un task di classificazione multiclasse predicendo una delle risposte possibili. Questo approccio prende quindi il nome di *joint-embeddings*, grazie appunto alla combinazione delle codifiche della componente testuale e visiva che sta alla sua base.

1.2.2 Grad-CAM

Di grande utilità per la comprensione dei modelli basati su CNN è stata sicuramente la tecnica di localizzazione basata sul gradiente *Grad-CAM* [27]. L'approccio si prefissa di usare i gradienti di un concetto obiettivo che confluisce nell'ultimo layer convoluzionale, che mantiene informazioni spaziali sull'immagine, per poi procedere a ritroso tramite retropropagazione per ottenere una *heatmap* da sovrapporre all'immagine.

Grazie all'applicazione del Grad-CAM nel VQA sono stati migliorati sia l'interpretabilità che l'efficacia dei modelli, fornendo uno strumento ulteriore per valutarne l'efficacia.

1.2.3 Meccanismo di attenzione bottom-up e top-down

Viene proposta in [3] una tecnica che si occupa per la prima volta in ambito VQA di proporre un sistema di attenzione a monte della metodologia proposta in [4]. Il meccanismo prevede una prima fase in cui si sfrutta un modello *Faster R-CNN* come *Region Proposal Network* (RPN) per il rilevamento di oggetti nell'immagine. Le features rilevate all'interno dell'immagine in questa fase saranno le nostre features spaziali v, che

verranno in seguito combinate grazie a una rete LSTM con l'input testuale. Questo passaggio permette quindi di andare a calcolare i pesi normalizzati per ogni feature rilevata nell'immagine, cosa che andrà a influenzare il modello per la scelta finale della risposta, proponendo quindi a valle del metodo la classificazione dell'etichetta corretta, cioè della risposta.

La diffusione dell'approccio è stata garantita dall'innovazione rispetto all'uso di griglie fisse per l'estrazione di features dalle immagini che per diversi anni ha rappresentato lo standard per il trattamento delle immagini nel task. Nonostante questa tecnica sia stata lo standard per alcuni anni, recentemente l'utilizzo dei transformers anche nel campo della Computer Vision ha dimostrato di poter non solo superare questo sistema, ma anche di performare con costi computazionali molto più bassi, come dimostrato da [14].

1.2.4 Utilizzo di KB esterna

Nonostante il principio originale del VQA preveda di attingere soltanto all'immagine per le informazioni, vengono proposte alternative come [5] che utilizzano basi di conoscenza esterne per, ad esempio, stabilire relazioni tra oggetti in forma di triplette:

<obj1, rel, obj2>

L'obiettivo dietro questi approcci e dietro la creazione di queste KB è quello di risolvere la necessità, per alcune domande, di conoscenza pregressa del modello o di, banalmente, quello che viene definito nella letteratura relativa come *common sense*, semplicemente buon senso.

1.3 Ambiti di applicazione

Nel tempo, lo spettro di applicazioni del VQA si è ampliato notevolmente, grazie alla possibilità di combinare l'interpretazione contestuale di testo e immagini per fornire informazioni rilevanti.

1.3.1 Analisi di grafici

Tra i casi d'uso che potrebbero sorprendere possiamo trovare l'analisi di grafici a partire non dai dati grezzi, ma bensì dalle immagini: è quello di cui si occupano modelli come Deplot [22] di Google, che possono trovare la loro utilità per esempio nel mondo manageriale come tool di assistenza a analisti e responsabili in ambito commerciale.

Ad esempio, un analista finanziario potrebbe caricare un grafico di andamento dei mercati e chiedere quale sia il valore massimo raggiunto da una particolare curva o quando si verifica un'intersezione.

In ambito scientifico, un ricercatore può utilizzare il VQA per analizzare un grafico sperimentale e fare domande sulla pendenza di un tratto lineare o su un punto di picco.

1.3.2 Ambito Medico

In ambito medico troviamo diverse applicazioni di supporto, come la **diagnosi assistita**, in qualità di strumento di supporto ai medici nell'identificazione di patologie come tumori e polipi e nell'analisi di immagini diagnostiche (radiografie, endoscopie, risonanze magnetiche, etc.).

Troviamo inoltre utilizzi nella **formazione medica**, in aiuto agli studenti che possono ricevere dinamicamente risposte a domande cliniche basate su immagini.

Abbiamo infine il **supporto chirurgico** per l'identificazione di strumenti o aree specifiche in video di procedure mediche, e la **telemedicina** per l'assistenza remota ai pazienti, con spiegazioni dettagliate su ciò che viene visualizzato in tempo reale.

1.3.3 Sicurezza e Sorveglianza

Nel settore della sicurezza, il VQA può essere utilizzato per il **monitoraggio di sistemi di sorveglianza**, ad esempio nell'interpretazione in tempo reale di feed video per il riconoscimento di potenziali minacce. Un'applicazione concreta è il monitoraggio di aeroporti, dove i modelli possono analizzare le telecamere di sicurezza per rilevare bagagli abbandonati o identificare comportamenti sospetti. Inoltre, nel **riconoscimento di oggetti e persone**, il VQA può essere impiegato per individuare veicoli specifici in una registrazione video o identificare individui basandosi su descrizioni fornite dagli operatori di sicurezza.

1.3.4 Industria e Manutenzione

Nel settore industriale, il VQA trova applicazione nel **controllo qualità**. Ad esempio, in una catena di montaggio automobilistica, il modello può analizzare immagini di componenti per verificare che non siano presenti difetti o anomalie. Può rispondere a domande relative alla presenza di difetti, come crepe su una superficie, o alla conformità di un pezzo rispetto agli standard di produzione.

Per quanto riguarda il **supporto alla manutenzione**, il VQA può essere utilizzato per identificare guasti in macchinari complessi. Ad esempio, un modello addestrato su immagini di turbine eoliche potrebbe rilevare segni di usura o danneggiamento, rispondendo a domande sulla presenza di parti danneggiate o sul rilevamento e locazione di guasti presenti.

1.4 Obiettivi

Il VQA, soprattutto in ambito medico, rappresenta una importante sfida tecnologica. Nonostante siano stati sviluppati negli anni modelli sempre più avanzati, l'applicazione in ambito medico rimane limitata a causa di due fattori principali: la complessità

intrinseca delle immagini diagnostiche e cliniche e la mancanza di dataset specializzati e sufficientemente eterogenei.

Il dataset KvasirVQA[9] cerca di colmare questa lacuna offrendo un insieme unico di immagini, relative domande e risposte. Se tuttavia i dati all'interno del dataset sono predisposti per utilizzi variegati (*Image Captioning*, *Text-based Image Generation* e classificazione di immagini), l'interesse verterà sulle tecniche di VQA.

Gli obiettivi principali della tesi saranno:

Analisi del dataset Un obiettivo fondamentale della tesi è rappresentato dall'analisi approfondita del dataset, con lo scopo di comprenderne le caratteristiche principali e contestualizzarlo adeguatamente nel dominio medico, evidenziando sia le peculiarità rilevanti per le applicazioni del VQA in ambito diagnostico, sia i limiti e le imperfezioni che possono essere smussati per migliorare ulteriormente la qualità dei dati a disposizione.

Valutazione delle tecniche e dei modelli La tesi si propone di valutare le tecniche e i modelli che rappresentano lo stato dell'arte nel campo del VQA, effettuando un confronto dettagliato in termini di accuratezza e capacità di comprensione dei dati, al fine di identificare le soluzioni più efficaci per il dominio medico.

Test di modelli con variazioni procedurali e fine-tuning Verranno testati sia modelli a cui sono state applicate semplici variazioni procedurali, sia modelli che sono stati sottoposti a fine-tuning specifico sui dati, al fine di valutare l'impatto di entrambe le strategie sulle prestazioni complessive nel contesto del dataset analizzato.

Identificazione delle debolezze dei modelli Verranno ricercati insight sulle possibili debolezze dei modelli, analizzando le prestazioni su diversi scenari e identificando i fattori principali che contribuiscono a eventuali limitazioni nei risultati ottenuti.

Proposte per ricerche future Sulla base degli esperimenti e delle analisi condotte, verranno proposti spunti di miglioramento e nuove direzioni di ricerca, con l'obiettivo di avanzare ulteriormente lo stato dell'arte nel campo del VQA, in particolare nel dominio medico.

Per il raggiungimento degli obiettivi descritti, la tesi si propone di approfondire i seguenti punti:

- Prestazioni dei modelli su un dataset medico complesso come il KvasirVQA.
- Aspetti dei dati che possono influenzare maggiormente le prestazioni dei modelli.
- Ottimizzazione dei modelli per migliorarne l'efficacia.
- Effetti di tecniche come il *prompting* per l'arricchimento dell'elemento testuale.

L'analisi svolta contribuirà a migliorare la comprensione delle potenzialità del VQA in ambiente medico, in particolare nelle applicazioni relative al tratto gastrointestinale.

1.5 Struttura della tesi

La struttura della tesi è la seguente:

1. Il primo capitolo è stato pensato per fornire una overview del lavoro svolto e delle motivazioni dietro la scelta degli argomenti;
2. Il secondo capitolo affronterà origine, problematiche e peculiarità legate al dataset KvasirVQA;
3. Il terzo capitolo esplorerà lo stato dell'arte del mondo del VQA;
4. Il quarto capitolo presenterà le modifiche sperimentali applicate ai modelli baseline;
5. Il quinto capitolo analizzerà i risultati sperimentali e confronterà le prestazioni delle diverse architetture testate;

Nel prossimo capitolo verrà introdotto il dataset KvasirVQA, insieme alle analisi condotte per comprenderne le caratteristiche principali.

Capitolo 2

Dataset

Procediamo a fare un'introduzione ai dataset storici che hanno contribuito alla ricerca in campo VQA prima di presentare e analizzare il KvasirVQA.

2.1 Dataset storici

2.1.1 VQA v1

Il dataset *VQA v1* viene proposto contestualmente al primo paper di ricerca sull'argomento [4], combinando immagini sia astratte, ovvero raffiguranti paesaggi artificiali e stilizzati, sia reali e provenienti dal dataset *Microsoft Common Objects in Context* (MSCOCO) [21] con domande e risposte generate da annotatori umani per la sua componente testuale, proponendo una vasta gamma di domande, dalle risposte binarie (Sì/No) a quesiti aperti e numerici. Parallelamente viene proposta una prima metrica per la valutazione di modelli specializzati in questo task che definisce l'*accuracy* del modello con la seguente formula:

$$\text{accuracy} = \min\left(\frac{\#\text{humans that provided that answer}}{3}, 1\right) \tag{2.1}$$

Basando la valutazione del modello in base alla percentuale di annotatori che hanno risposto allo stesso modo alla domanda.

Nonostante la sua importanza, il dataset *VQA v1* presentava alcune limitazioni significative, come una distribuzione non bilanciata delle domande, permettendo ai modelli di ottenere buoni risultati sfruttando bias nei dati, senza una reale comprensione visiva.

Il *VQA v1* ha gettato le basi per lo sviluppo di dataset più avanzati, come il *VQA v2*, che ha affrontato molte delle sue criticità per fornire una valutazione più rigorosa delle capacità dei modelli.

2.1.2 Dataset VQA v2

Il dataset *VQA v2* [11] è stato rilasciato come evoluzione della versione originale *VQA v1*, ed è stato progettato per andare incontro ad alcune delle limitazioni emerse dalla prima versione.

Il dataset *VQA v2* contiene circa 265.000 immagini provenienti da *MSCOCO*, ognuna associata a domande e risposte multiple. Ogni domanda è bilanciata da coppie di immagini che condividono la stessa domanda ma hanno risposte diverse, al fine di ridurre il bias nei dati. Ad esempio, una domanda sulla quantità di animali presente nell'immagine può avere risposte diverse a seconda della scena mostrata.

Il dataset include oltre 1,1 milioni di domande e circa 11 milioni di risposte raccolte tramite annotazioni umane. Le domande spaziano tra diverse tipologie, come:

- **Domande binarie**: Richiedono una risposta *Sì* o *No*.
- **Domande numeriche**: Prevedono risposte sotto forma di conteggio.
- **Domande aperte**: Richiedono risposte descrittive, come nomi di oggetti o azioni.

Il bilanciamento introdotto in *VQA v2* ha migliorato significativamente la qualità delle valutazioni dei modelli, riducendo la possibilità di ottenere alte performance sfruttando semplici bias nei dati. Questo rende il dataset particolarmente adatto per testare modelli di VQA che si basano su una comprensione reale delle relazioni tra testo e immagine.

2.1.3 Visual Genome

Il dataset *Visual Genome* [15] contiene circa 108.000 immagini annotate con una vasta gamma di informazioni, incluse relazioni tra oggetti, attributi e domande con risposte generate da annotatori umani.

Oltre alle domande e risposte, Visual Genome offre annotazioni dettagliate a livello di oggetto e di scena, includendo descrizioni, relazioni semantiche e attributi, fornendo un contesto più ricco rispetto a dataset come *VQA v2*. Questa granularità lo rende particolarmente utile per addestrare modelli in grado di catturare relazioni complesse tra gli elementi visivi.

Grazie alla combinazione di annotazioni visive e testuali, Visual Genome è stato ampiamente utilizzato per migliorare le capacità di ragionamento visivo nei modelli di VQA, soprattutto per il loro addestramento su domande che richiedono una comprensione contestuale delle immagini.

2.1.4 Graph-based Question Answering (GQA)

Il dataset *(GQA)* [19] è progettato per domande strutturate logicamente, basandosi su rappresentazioni grafiche delle scene visive. Contiene oltre 1,7 milioni di domande

generate automaticamente, derivanti da 113.000 immagini annotate.

Ogni immagine è rappresentata da un grafo di scene che include oggetti, attributi e relazioni spaziali. Le domande sono costruite per riflettere percorsi logici che attraversano i nodi del grafo, rendendo il task particolarmente complesso.

GQA è stato progettato per migliorare le capacità di ragionamento strutturato dei modelli, rendendolo un riferimento importante per valutare come il ragionamento basato su strutture come i grafi si posizioni a livello di performance rispetto ad altre tecniche.

2.1.5 PathVQA

Il dataset *PathVQA* [12] è uno dei pochi dataset specificamente progettati per il VQA in ambito medico. È composto da circa 32.800 immagini patologiche accompagnate da domande che richiedono competenze diagnostiche specialistiche.

Le domande riguardano anomalie, caratteristiche visive e dettagli clinici presenti nelle immagini. La struttura del dataset è pensata per valutare la capacità dei modelli di rispondere correttamente a domande basate su conoscenze mediche.

Volendo fare un confronto critico, PathVQA rappresenta un riferimento diretto per il KvasirVQA, condividendo un focus simile sul contesto medico. Tuttavia, KvasirVQA si distingue per il suo focus sull'endoscopia gastrointestinale.

2.1.6 ScienceQA

Nel contesto del VQA nel dominio scientifico, è sicuramente utile menzionare il dataset *ScienceQA* [24], sviluppato per affrontare il problema del ragionamento multimodale combinando informazioni provenienti da testo, immagini e conoscenze strutturate per rispondere a domande educative.

ScienceQA contiene oltre 21.000 domande tratte da materiali educativi di scienze, organizzate in 26 categorie (biologia, chimica, fisica, ecc.) e distribuite su tre livelli educativi (elementare, medio, avanzato). Ogni domanda è accompagnata da:

- Un'immagine correlata che arricchisce il contesto visivo.
- Risposte multiple.
- Spiegazioni dettagliate che delineano il ragionamento necessario per raggiungere la risposta corretta.

Il lavoro presentato si concentra sull'integrazione di fonti multimodali per rispondere alle domande. In particolare, il modello proposto utilizza tecniche avanzate come il **Multimodal Chain-of-Thought Reasoning**, che simula il processo di ragionamento umano combinando contenuti testuali e visivi con conoscenze esterne, tecnica che verrà approfondita nel prossimo capitolo e che sarà impiegata per lo studio delle prestazioni di alcuni modelli generativi sul KvasirVQA.

Ad esempio, il modello potrebbe integrare informazioni testuali e visive per rispondere a domande relative alla funzione di un organo evidenziato in un'immagine, fornendo risposte basate su conoscenze scientifiche consolidate.

ScienceQA offre un banco di prova per testare la capacità dei modelli di risolvere problemi complessi che richiedono non solo la comprensione visiva, ma anche il ragionamento e la conoscenza scientifica. Questo lo distingue da dataset generici di VQA, focalizzandosi su domande che richiedono una fusione profonda delle modalità e un metodo di approccio al problema sicuramente più simile a quello umano.

Il concetto di **Multimodal Chain-of-Thought Reasoning** esplorato in ScienceQA rappresenta una direzione promettente per l'applicazione al dominio medico. In particolare, le tecniche utilizzate per guidare il ragionamento attraverso spiegazioni strutturate possono ispirare nuove strategie per migliorare le performance su dataset che sfruttano tecniche simili.

2.2 KvasirVQA

Passiamo adesso ad un'introduzione del KvasirVQA con a seguire un'analisi approfondita per la contestualizzazione.

2.2.1 Introduzione a KvasirVQA

Il dataset KvasirVQA è stato creato per affrontare una lacuna significativa nella ricerca sul VQA medico, offrendo dati specifici per immagini diagnostiche. Derivato dai dataset HyperKvasir [7] e Kvasir-Instrument [7], il KvasirVQA [9] si distingue per l'integrazione di immagini mediche e quesiti strutturati, concepiti per supportare la diagnosi e il rilevamento di anomalie in un contesto clinico altamente specializzato. Il KvasirVQA è stato utilizzato in questa tesi come punto di partenza per lo studio sperimentale, senza interventi diretti nella sua raccolta o annotazione, per indagare gli effetti delle variazioni sui modelli affermati nel campo del VQA.

Nel presente capitolo verranno analizzate in dettaglio le caratteristiche principali del dataset, la sua rilevanza per il task e il suo ruolo specifico nello studio sperimentale condotto.

2.2.2 Origine

Rilasciato a settembre 2024, il dataset KvasirVQA combina immagini provenienti dai dataset HyperKvasir e Kvasir-Instrument, integrando le caratteristiche di entrambi per fornire un insieme completo e diversificato di dati visivi.

HyperKvasir comprende immagini e video di ispezioni del tratto gastrointestinale corredati da annotazioni relative sia a ritrovamenti anomali che a landmark anatomici, utili per la localizzazione durante le procedure endoscopiche. Tuttavia, presenta alcune

limitazioni, tra cui uno sbilanciamento nella distribuzione delle classi e la mancanza di annotazioni dettagliate per molte categorie. Un'analisi preliminare ha rivelato che la maggior parte delle immagini annotate appartiene alla classe "Polipo", riflettendo la maggiore frequenza di questi ritrovamenti rispetto ad altre anomalie. Inoltre, le annotazioni relative alla posizione degli oggetti (ad esempio, bounding-box) sono disponibili solo per i polipi, a causa delle difficoltà nell'isolare altre tipologie di anomalie e della presenza di etichette che descrivono l'immagine nel suo complesso, come l'indice Boston Bowel Preparation Scale (BBPS).

Kvasir-Instrument, invece, si concentra su immagini contenenti strumenti medici come pinze da biopsia e sonde, fornendo annotazioni precise tramite bounding-box per localizzare questi strumenti. Questa caratteristica arricchisce il dataset KvasirVQA con informazioni visive fondamentali per applicazioni diagnostiche e interventistiche.

La combinazione di queste risorse rende il dataset KvasirVQA particolarmente adatto a task multimodali come il Visual Question Answering, grazie alla varietà e complessità delle immagini e delle annotazioni fornite. Tuttavia, alcune limitazioni ereditarie, come lo sbilanciamento tra le classi e la disponibilità limitata di bounding-box per determinate categorie, hanno influenzato la configurazione degli esperimenti descritti in questa tesi. In particolare, alcuni degli approcci adottati si basano sull'uso di features globali, evitando l'introduzione di meccanismi di attenzione complessi.

2.2.3 Statistiche e Caratteristiche

La componente testuale, composta da domande e risposte, è stata sviluppata con il contributo di esperti nel campo della gastroenterologia, includendo una varietà di formati, come domande a risposta multipla, binaria (Sì/No) e altre tipologie.

La distribuzione delle immagini divisa per origine è la seguente:

Categoria	**N**	**Sorgente**
Normal	2500	HyperKvasir
Polyps	1000	HyperKvasir
Esophagitis	1000	HyperKvasir
Ulcerative Colitis	1000	HyperKvasir
Instrument	1000	Kvasir-Instrument
Totale	6500	

Table 2.1: Distribuzione immagini ricavata da [9] con relativo dataset di origine e cardinalità

La *label* "Normal", in particolare, fa riferimento a delle immagini in cui non sono presenti anormalità o segni distintivi di alcune patologie, ma che semplicemente danno delle informazioni sulla posizione specifica in cui ci si trova all'interno del tratto GI. Tra queste, troviamo ad esempio *cecum* e *ileum*, ovvero l'intestino cieco e ileo. La presenza

di immagini etichettate come "Normal" consente di addestrare i modelli a distinguere scenari diagnostici complessi da quelli standard, migliorando la loro capacità di generalizzazione e identificazione di patologie.

Per ciascuna delle 6500 immagini presenti, sono state generate sei coppie di domande e risposte, distribuite tra le principali macro-categorie considerate. A seguire è riportata una tabella riassuntiva, tratta da [9], che offre una panoramica generale sulla componente testuale del dataset.

Type	Question	Answer Options
Yes/No	Have all polyps been removed? / Is this finding easy to detect? / Is there text? / Is there a green/black box artefact? / Does this image contain any finding? / What type of polyp is present?	No / Yes / Not relevant
Choice (Single)	What type of procedure is the image taken from?	Paris Ip / Paris IIa / Paris Is / Capsule Endoscopy / Colonoscopy / Gastroscopy
Choice (Single)	What is the size of the polyp?	<5mm / 5-10mm / 11-20mm / >20mm
Choice (Multiple)	Are there any abnormalities in the image?	Oesophagitis, Ulcerative Colitis, Short-Segment Barretts, Barretts, Polyp, Hemorrhoids
Choice (Multiple)	Are there any anatomical landmarks in the image?	Ileum, Z-Line, Cecum, Pylorus
Choice (Multiple)	Are there any instruments in the image?	Tube, Metal Clip, Polyp Snare, Injection Needle, Biopsy Forceps
Choice (Color)	What color is the abnormality?	landmark: grey, flesh, pink, black, orange, etc.
Choice (Color)	What color is the anatomical landmark?	pink, red, etc.
Location	Where in the image is the abnormality? / Where in the image is the instrument? / Where in the image is the anatomical landmark?	Upper-Left / Upper-Center / Upper-Right / Center-Left / Center / Center-Right / Lower-Left / Lower-Center / Lower-Right
Numerical Count	How many findings are present? / How many polyps are in the image? / How many instruments are in the image?	[0-inf]

Table 2.2: Domande, categorizzate per tipologia, e relative risposte dal KvasirVQA[9]

Le domande presenti nel dataset sono diversificate e comprendono formati che spaziano da risposte binarie al conteggio delle anomalie visibili nell'immagine, fino al rilevamento di artefatti specifici, come scritte o riquadri generati dal software utiliz-

zato durante le ispezioni del tratto gastrointestinale. Questa varietà di domande riveste un ruolo cruciale, poiché consente di valutare la capacità dei modelli di comprendere il contenuto visivo, identificare gli elementi di interesse e analizzare il processo attraverso cui il modello osserva e interpreta le immagini.

2.3 Conclusioni

Il KvasirVQA è un passo in avanti importante in ambito medico grazie al suo contributo nel dominio dei dati relativi al tratto gastrointestinale, sia per quanto riguarda il solo task del VQA, che è stato il focus della tesi, sia per altri task come l'*Image Captioning* e la generazione di immagini sintetiche. Non escludiamo che possa avere una rilevanza anche maggiore in futuro, con il rilascio di versioni ulteriormente arricchite e maggiormente bilanciate, con un supporto maggiore da parte di esperti gastroenterologi e con l'applicazione da parte di modelli che utilizzino architetture e tecnologie innovative e più evolute rispetto a quelle attualmente esistenti. Il dataset KvasirVQA rappresenta un passo avanti rispetto ai dataset generici come VQA v2 [11] o Visual Genome [15], grazie alla sua specializzazione in ambito medico e alla diversificazione delle domande. Tuttavia, questa unicità pone sfide specifiche, come l'interpretazione di immagini diagnostiche complesse e la gestione di categorie sbilanciate, che richiedono approcci modellistici avanzati.

Capitolo 3

Stato dell'arte

3.1 Introduzione

Alla luce delle peculiarità del dataset KvasirVQA, è fondamentale analizzare le tecniche e i modelli che hanno dominato il campo del VQA negli ultimi anni. In questo capitolo verranno spiegati i dataset principali che hanno contribuito allo sviluppo del VQA, evidenziando brevemente le loro caratteristiche principali e il ruolo avuto nella definizione degli standard attuali, per una migliore contestualizzazione del KvasirVQA nel panorama dei dataset disponibili.

A seguire, verranno introdotti i modelli e le tecniche applicate al VQA che negli anni hanno provato a essere particolarmente efficaci per la comprensione visiva e testuale.

3.2 Tecniche VQA

Sebbene la maggior parte dei progressi tecnologici della ricerca in ambito VQA siano concentrati sulle immagini, una parte importante della ricerca si è concentrata sullo studio di tecniche da applicare alla componente testuale, grazie soprattutto all'impiego di modelli basati su transformers. Alcune tecniche si occupano per esempio di simulare un processo più simile al ragionamento umano per migliorare la capacità di ragionamento del modello stesso, o semplicemente per ricevere una spiegazione contestuale di come il modello sia arrivato a fornire una risposta a una determinata domanda. Stiamo parlando in particolare di tecniche di *prompting*, ovvero tentativi di manipolazione dell'input testuale dei modelli linguistici per guidarlo nella generazione della risposta. Recentemente, per esempio, OpenAI ne ha fatto uso per la creazione del modello *o1* [13], realizzando una soluzione basata su apprendimento rinforzato e *Chain of Thought* (CoT). In campo scientifico possiamo vederne applicazioni sul già menzionato *ScienceQA* in cui vediamo uno schema più ampio, in cui vengono coinvolti altri elementi aggiuntivi oltre all'immagine e al testo di domanda e risposta:

- *Question*: la domanda che viene posta al modello

- *Answer*: la risposta finale del modello
- *Lecture*: una breve spiegazione di carattere scientifico del contesto
- *Explanation*: una spiegazione che applica la *lecture* a quello che è il contesto della domanda

Altri studi hanno invece dimostrato l'efficacia di utilizzare il prompting in diversi modi per affrontare dei task di VQA, come possiamo vedere in [6]. Il vantaggio fondamentale di queste tecniche risiede nel poter adattare un modello seguendo diverse implementazioni particolari della tecnica generale, rendendo il *fine tuning* di un modello non necessario, risparmiando così le ingenti risorse che sarebbero necessarie altrimenti. Alcune di queste tecniche di prompting verranno approfondite ulteriormente nel prossimo capitolo, dal momento che saranno oggetto di sperimentazioni sul dataset scelto.

3.2.1 Prompting

Verranno adesso analizzate brevemente alcune tecniche di prompting che hanno dimostrato poter portare a dei miglioramenti nelle performance dei VLM, prendendo in particolare spunto da quelle presentate in [6].

Templating

Il *templating* è una tecnica di prompting che prevede l'uso di strutture predefinite e standardizzate per formulare input testuali. Questa metodologia aiuta a migliorare la comprensione del task, garantendo coerenza e chiarezza nell'interazione tra input e modello.

La tecnica consiste nel creare schemi preformattati, chiamati appunto *template*, che definiscono il formato e la struttura dell'input. Ogni template contiene elementi statici e dinamici:

- **Elementi statici:** Parti fisse del prompt che forniscono il contesto, istruzioni generali o un'impostazione della risposta.
- **Elementi dinamici:** Variabili che vengono riempite con informazioni specifiche del task, come una domanda, una descrizione testuale o una lista di opzioni possibili per la risposta.

Un template potrebbe avere una forma simile alla seguente:

> *"Dati l'immagine e la seguente domanda: '[Domanda]'.*
>
> *Fornisci una risposta completa basata sui contenuti visivi e testuali.*
>
> Opzioni: ['Opzioni']."

Dove *[Domanda]* e *[Opzioni]* rappresentano gli elementi dinamici che cambiano per ogni esempio.

Il templating è una tecnica che può portare diversi vantaggi, come una migliore interpretazione del task da parte del modello, una standardadizzazione di input e output per il modello e una certa flessibilità per l'adattamento a specifici task.

In un contesto multimodale come quello del VQA dove è necessario integrare informazioni visive e testuali, può essere particolarmente utile. In questi casi, il template guida il modello nella fusione delle modalità, enfatizzando la rilevanza delle informazioni visive in relazione alla domanda.

Nonostante i vantaggi, l'implementazione di questa tecnica presenta alcune sfide:

- **Dipendenza dalla Formulazione:** Piccole variazioni nella struttura del template possono influire sulle prestazioni del modello.
- **Bias:** Template mal progettati possono introdurre bias o guidare il modello verso risposte errate.

Il templating rappresenta una strategia potente e flessibile per migliorare le capacità di interpretazione e risposta dei VLM. La sua combinazione con altre tecniche di prompting, come il CoT, può ulteriormente potenziare l'efficacia del modello nei task complessi.

Zero-shot e few-shot prompting

Le tecniche di prompting *few-shot* e *zero-shot* rappresentano approcci fondamentali per sfruttare modelli in scenari con poca o nessuna supervisione esplicita. Entrambi i metodi sfruttano la capacità di generalizzare a compiti nuovi senza richiedere un addestramento specifico.

Zero-shot Prompting Il *zero-shot prompting* consiste nel fornire al modello solo una descrizione testuale del task, senza esempi espliciti di input-output, facendo quindi uso della sua conoscenza pre-addestrata per inferire la risposta.

Per un task di classificazione di sentimenti, ad esempio, un prompt zero-shot potrebbe essere:

> *"Determina se il sentimento espresso nella seguente frase è positivo o negativo:*
>
> *'Questo prodotto è fantastico'."*

Il modello genera direttamente la risposta basandosi esclusivamente sulle informazioni fornite nel prompt.

I vantaggi di questa tecnica sono evidenti, considerando soprattutto l'onere computazionale che rappresenta l'addestramento di specializzazione di modelli generici, andando quindi a sfruttare la rapidità e la versatilità dell'implementazione.

I modelli che ne fanno uso potrebbero tuttavia presentare prestazioni inferiori rispetto ad altri che utilizzano, per esempio, il few-shot in task più complessi, oltre a dipendere fortemente dalla chiarezza e dalla precisione del prompt.

Few-shot Prompting Il *few-shot prompting* include alcuni esempi di input-output nel prompt per aiutare il modello a comprendere meglio il task. Questo approccio sfrutta la capacità del modello di generalizzare da pochi esempi forniti "on the fly".

Per il medesimo task di classificazione dei sentimenti, un prompt few-shot potrebbe essere:

> *"Classifica il sentimento come positivo o negativo.*
>
> *Esempi:*
>
> *1. 'Il cibo era delizioso.' →Positivo*
>
> *2. 'Il servizio era pessimo.' →Negativo*
>
> *Ora analizza: 'Questo prodotto è fantastico'."*

Il modello utilizza quindi gli esempi forniti per formulare la risposta.

Oltre a godere degli stessi vantaggi dello zero-shot, può essere sicuramente più performante su task che vanno ad essere più complessi, così come soffrire degli stessi problemi, come la dipendenza dal prompt utilizzato che può anche appesantire, in caso di prompt più lunghi e specializzanti, il costo computazionale.

Confronto tra Zero-shot e Few-shot Prompting

- **Contesto d'uso:** Lo zero-shot è preferibile per task generici o ben definiti, mentre il few-shot è più adatto a scenari complessi o ambigui.
- **Dipendenza dagli esempi:** Lo zero-shot non utilizza esempi, mentre il few-shot richiede una selezione accurata degli esempi.
- **Efficienza:** Lo zero-shot è più rapido e meno costoso in termini computazionali, mentre il few-shot può essere più efficace ma richiede maggiore memoria e attenzione alla progettazione del prompt.

Chain-of-Thought

Il *Chain of Thought* (CoT) è una tecnica di prompting che guida i modelli nel generare risposte complesse attraverso un ragionamento esplicito e passo-passo. L'approccio si basa sull'idea che l'esplicitazione del processo logico possa migliorare l'accuratezza e la comprensione del task, specialmente nei problemi che richiedono ragionamenti complessi.

Il CoT incoraggia il modello a seguire una sequenza logica di passaggi per arrivare alla risposta finale. Invece di generare direttamente la risposta, il modello elabora ogni

fase del ragionamento, migliorando la trasparenza e riducendo gli errori nei calcoli o nei passaggi intermedi.

Per un problema matematico, un esempio di CoT potrebbe essere:

> *"Ci sono tre mele in ogni cestino e quattro cestini. Quante mele ci sono in totale? Ragiona passo per passo: 1. Ogni cestino contiene tre mele. 2. Ci sono quattro cestini in totale. 3. Moltiplico tre per quattro. 4. Risultato: ci sono dodici mele in totale."*

Il CoT consente sicuramente ai modelli di affrontare problemi complessi suddividendoli in parti più gestibili, utilizzando un approccio di risoluzione del problema di tipo *divide et impera* e portando sì a una generalizzazione per l'adattamento a un task specifico, ma anche a una maggiore trasparenza grazie ai passaggi logici che vengono esplicitati. Può inoltre essere pensata come una tecnica che permette il debug del modello.

Il CoT è particolarmente efficace nella risoluzione di problemi in cui la logica è di primaria importanza, come problemi matematici e scientifici [24] e domande aperte nel VQA per cui il modello fornisce spiegazioni sulle informazioni visive.

Il CoT può essere implementato in combinazione con tecniche di prompting come il *few-shot* e il *templating*. Ad esempio:

> *"Esempi: 1. Qual è la somma di 2 e 3? Ragiona passo per passo: - Primo passo: Determina il valore di 2. - Secondo passo: Determina il valore di 3. - Somma: 2 + 3 = 5. Ora risolvi: Qual è il doppio di 6?"*

Nonostante i vantaggi, il CoT presenta alcune limitazioni condivise con altri approcci di prompting già visti, ovvero la dipendenza dal prompt, fattore che può portare a sviluppare un bias nelle risposte, e un costo computazionale che cresce insieme alla complessità del ragionamento necessario per la formulazione della risposta.

Il Chain of Thought rappresenta una potente tecnica di prompting per migliorare il ragionamento e l'accuratezza nei modelli di grandi dimensioni. La sua capacità di scomporre problemi complessi in passaggi logici lo rende un elemento chiave nei task che richiedono comprensione profonda e ragionamenti articolati.

3.3 Modelli

Dopo aver analizzato alcune tecniche core dei VLM moderni, passiamo adesso ad un'analisi dei principali modelli multimodali presenti.

3.3.1 Vision Transformer (ViT)

Il *Vision Transformer (ViT)* [8] è una delle prime architetture basate su *Transformer* applicate al dominio visivo. A differenza delle tradizionali CNN, che utilizzano convoluzioni per estrarre caratteristiche dalle immagini, ViT adotta un approccio ispirato

ai Transformer originariamente introdotti per l'NLP. ViT divide l'immagine di input in una griglia di patch quadrate di dimensione fissa, trattando ciascun patch come un "token" analogo alle parole in un modello NLP. Ogni patch viene quindi trasformato in un embedding vettoriale tramite una proiezione lineare. Questi embedding, insieme a un embedding di posizione, vengono forniti al Transformer, che applica meccanismi di attenzione per elaborare le relazioni tra i patch. Grazie alla capacità di catturare relazioni globali tra i patch, ViT ha mostrato una maggiore efficienza rispetto alle CNN tradizionali, specialmente su immagini ad alta risoluzione.

3.3.2 Vision-and-Language Transformer (ViLT)

Il *Vision-and-Language Transformer (ViLT)* [14] è un'estensione del concetto di Transformer al dominio multimodale, progettato per combinare informazioni visive e testuali. ViLT affronta task come il *Visual Question Answering (VQA)* integrando immagini e testo in un unico spazio rappresentativo.

A differenza di molti modelli multimodali che utilizzano un backbone visivo preaddestrato separato, tecnica comunemente nota come *Vision-Language Pretraining*, ViLT elimina la necessità di un estrattore di caratteristiche visive dedicato. Le immagini vengono direttamente convertite in patch e fornite al Transformer insieme ai token testuali. Il modello utilizza un unico Transformer per elaborare entrambe le modalità contemporaneamente ed è progettato specificatamente per task multimodali.

ViT e ViLT condividono l'idea di utilizzare i Transformer per elaborare le immagini, ma differiscono per finalità e architettura:

- **Dominio:** ViT è progettato esclusivamente per compiti visivi, mentre ViLT è orientato a task multimodali che combinano immagini e testo.
- **Backbone visivo:** ViT utilizza esclusivamente immagini come input, mentre ViLT incorpora anche input testuali, eliminando la necessità di un backbone visivo separato.
- **Applicazioni:** ViT è focalizzato su task come classificazione e segmentazione di immagini, mentre ViLT affronta task come VQA.
- **Efficienza:** ViLT è più leggero dal punto di vista computazionale grazie alla sua architettura unificata, ma può perdere in precisione visiva rispetto a modelli con backbone separati.

3.3.3 Contrastive Language–Image Pretraining(CLIP)

CLIP (**Contrastive Language–Image Pretraining**) [26] è un modello multimodale sviluppato da OpenAI che apprende a collegare immagini e descrizioni testuali in un unico spazio rappresentativo. Questo approccio innovativo si basa sull'apprendimento

contrastivo, consentendo a CLIP di associare immagini e testi senza la necessità di un addestramento supervisionato specifico per il task.

CLIP utilizza due encoder separati:

- Un **encoder visivo**, che può essere basato su modelli come ResNet o Vision Transformer (ViT), per estrarre caratteristiche dalle immagini.
- Un **encoder testuale**, basato su Transformer, che rappresenta il testo in un embedding denso.

Questi encoder generano rappresentazioni indipendenti di immagini e testo, mappandoli in uno spazio latente condiviso. Il modello è addestrato a massimizzare la similarità tra coppie corrette (immagine-testo) e a minimizzare la similarità tra coppie errate.

CLIP è stato addestrato su un dataset composto da 400 milioni di coppie immagine-testo raccolte da Internet. Questo ampio corpus non supervisionato consente al modello di generalizzare a un'ampia varietà di compiti senza necessità di fine-tuning.

Nonostante i risultati promettenti, CLIP presenta alcune limitazioni:

- **Bias nei dati:** Essendo addestrato su dati raccolti da Internet, CLIP può riflettere i bias presenti nei dati.
- **Prestazioni limitate in contesti specifici:** CLIP può avere difficoltà con compiti che richiedono una comprensione specialistica, come nel dominio medico.
- **Dipendenza dalla qualità dei prompt:** La formulazione del testo gioca un ruolo cruciale nelle performance del modello.

CLIP rappresenta un esempio di successo nell'integrazione di informazioni visive e testuali in un unico spazio rappresentativo. Le sue capacità di generalizzazione e il supporto ai compiti zero-shot lo rendono un modello di riferimento per approcci multimodali. Nel contesto del VQA, CLIP può essere utilizzato come feature extractor o come punto di partenza per approcci basati su prompting, fornendo una base solida per migliorare la comprensione visivo-testuale.

3.3.4 Bootstrapping Language-Image Pre-training(BLIP)

BLIP (Bootstrapping Language-Image Pre-training) [18] è un modello multimodale progettato per affrontare compiti che richiedono l'integrazione di dati visivi e testuali, come il VQA, *Image Captioning* e il *Image-Text Retrieval*. L'architettura di BLIP combina un encoder visivo e un decoder testuale, costruiti per massimizzare l'allineamento tra immagini e descrizioni testuali durante la fase di pretraining.

Il modello utilizza una base ViT come encoder visivo per estrarre rappresentazioni ricche dalle immagini, mentre il decoder testuale si basa su un Transformer standard,

ottimizzato per generare descrizioni coerenti e contestuali. BLIP è addestrato utilizzando un ampio corpus di dati immagine-testo, con tecniche di apprendimento contrastivo e generativo per migliorare la qualità delle rappresentazioni multimodali.

Uno degli aspetti chiave di BLIP è la sua capacità di adattarsi a diversi task multimodali tramite fine-tuning, sfruttando l'allineamento già appreso durante il pretraining. Per il task di VQA, il modello si distingue per la sua capacità di comprendere domande in linguaggio naturale e di rispondere basandosi sui contenuti visivi, grazie a una fusione efficace delle modalità.

Tuttavia, BLIP presenta alcune limitazioni intrinseche, come una complessità computazionale relativamente alta e la dipendenza da un numero significativo di dati annotati per ottenere risultati ottimali nei task specifici. Nonostante ciò, ha rappresentato un importante punto di partenza per successive evoluzioni nei modelli multimodali.

3.3.5 Generative Image-to-Text Transformer (GIT)

Il *GIT* (Generative Image-to-Text Transformer) [30] utilizza un'architettura semplificata basata su Transformer.

Oltre al VQA, GIT è stato applicato con successo a diversi compiti, tra cui:

- **Image Captioning:** Generazione di descrizioni testuali per immagini.
- **Video Captioning:** Creazione di descrizioni per contenuti video.

GIT ha stabilito nuovi standard in vari benchmark, superando le prestazioni umane in alcuni casi. Ad esempio, ha ottenuto un punteggio CIDEr di 138,2 su TextCaps, rispetto al punteggio umano di 125,5.

Grazie alla sua architettura semplice e diretta che combina un encoder di immagini con un decoder di testo, elimina la necessità di moduli esterni complessi, semplificando l'architettura complessiva.

Grazie a questa struttura, GIT si distingue per la sua efficienza computazionale e per le prestazioni all'avanguardia in compiti come la didascalizzazione di immagini e video. La semplicità dell'architettura consente di bilanciare efficacemente capacità di generalizzazione e velocità di addestramento.

3.3.6 Large Language and Vision Assistant (LLaVA) e LLaVA-Med

LLaVA (**Large Language and Vision Assistant**) [23] è un modello multimodale open-source progettato per integrare input visivi e testuali, consentendo la generazione di risposte basate su entrambi i tipi di dati e sfruttando un approccio basato sul *Visual Instruction Tuning*.

LLaVA utilizza un encoder visivo pre-addestrato, come CLIP, e lo collega a un LLM tramite una matrice di proiezione adattabile. Questo setup permette la trasformazione

delle rappresentazioni visive in embedding testuali condivisi, migliorando l'interazione multimodale.

Nonostante l'addestramento su un dataset relativamente ridotto di circa 80.000 immagini, LLaVA ha dimostrato un'elevata efficacia, raggiungendo prestazioni comparabili a quelle di GPT-4 in compiti multimodali specifici.

LLaVA-Med [17] è un adattamento di LLaVA specifico per il dominio medico, progettato per affrontare sfide diagnostiche e interpretative basate su immagini cliniche. Questo modello utilizza un approccio simile a LLaVA ma è stato ottimizzato per elaborare dati medici, come immagini di radiografie o scansioni diagnostiche, insieme a domande in linguaggio naturale.

Entrambe le configurazioni hanno permesso di esplorare il potenziale di modelli multimodali nel dominio medico, valutandone la capacità di gestire task complessi, come la classificazione delle anomalie e la risposta a domande specifiche relative a strumenti o procedure.

3.3.7 Florence-2

Florence-2 [31] è un modello sviluppato da Microsoft che rappresenta uno dei sistemi di fondazione visiva più avanzati. Il modello è stato progettato per gestire molteplici compiti di visione artificiale utilizzando una rappresentazione unificata basata su prompt testuali, rendendolo altamente adattabile a diversi scenari.

Florence-2 adotta un'architettura sequence-to-sequence per elaborare input visivi e generare output testuali corrispondenti al compito specifico. Il modello è stato addestrato su FLD-5B, un dataset non ancora rilasciato di 5,4 miliardi di annotazioni su 126 milioni di immagini, utilizzando un processo iterativo di annotazione e raffinamento.

Grazie al suo addestramento su larga scala e alla struttura innovativa, Florence-2 ha dimostrato eccellenti capacità di zero-shot e prestazioni superiori su compiti visivi standard. Nonostante Florence-2 non supporti originalmente il task di VQA e si concentri maggiormente sul captioning, è comunque possibile fare fine-tuning per adattarlo anche a questo task.

3.3.8 GPT-4V

GPT-4V [1] è un'estensione del modello GPT-4 di OpenAI, progettata per includere capacità visive oltre a quelle linguistiche. Questo modello consente l'elaborazione simultanea di input testuali e visivi, rendendolo particolarmente utile per compiti multimodali avanzati.

Sebbene i dettagli architetturali completi di GPT-4V non siano stati divulgati, è noto che il modello utilizza tecniche di embedding condivisi per unificare la rappresentazione visiva e testuale. Questa architettura permette al modello di interpretare e rispondere a domande complesse che coinvolgono informazioni visive e linguistiche in

modo integrato.

GPT-4V si distingue per la sua capacità di ragionamento multimodale e per le sue applicazioni nei campi dell'assistenza visiva e dell'interazione utente naturale.

3.4 Conclusione

In questo capitolo sono state analizzate le tecniche principali e i modelli che hanno guidato lo sviluppo del VQA. Questa panoramica fornisce il contesto necessario per comprendere le strategie sperimentali adottate nella tesi, che verranno esplorate nei capitoli successivi.

Capitolo 4

Approccio Proposto

4.1 Introduzione

In questo capitolo verranno presentate le proposte sperimentali sviluppate per indagare le capacità dei modelli di VQA nel contesto medico del *KvasirVQA*.

A partire dalle tecniche e dai modelli analizzati nei capitoli precedenti, sono state definite strategie sperimentali volte a testare l'efficacia di diverse architetture e approcci, includendo sia modelli basati su tecniche tradizionali, sia soluzioni più recenti che sfruttano il prompting, il *Chain-of-Thought* e i transformers. A supporto di queste operazioni, è stato effettuato anche un trattamento di pulizia e *data augmentation*, per rendere possibili alcuni esperimenti e tentare di migliorare la qualità del dataset. L'obiettivo principale è quello di esplorare le potenzialità dei modelli multimodali nell'elaborazione di immagini diagnostiche e domande strutturate, valutandone i limiti e le possibili aree di miglioramento.

Le proposte sperimentali includono:

- L'applicazione di tecniche di prompting, con particolare attenzione al *templating* e al *Chain-of-Thought*.
- Adozione di architetture personalizzate per verificare l'efficacia sui dati a disposizione.
- Variazioni procedurali per affrontare il task in forma di classificazione multilabel, oltre che multiclasse.
- Utilizzo di un'architettura moderna e basata su transformers, addestrata per adattarsi al contesto del KvasirVQA.
- Operazioni di data augmentation sulle classi scarsamente bilanciate e correzione di imprecisioni testuali.

L'obiettivo di queste proposte è non solo di testare l'efficacia di tecniche avanzate, ma anche di fornire insight utili per l'ottimizzazione dei modelli di VQA in ambito

medico, contribuendo così a colmare le lacune identificate nella ricerca attuale. I risultati ottenuti verranno discussi nel capitolo successivo, con un'analisi dettagliata delle prestazioni e del confronto tra le diverse configurazioni sperimentali.

4.2 Analisi, limiti e trattamento dei dati

L'analisi approfondita del dataset KvasirVQA ha messo in evidenza alcune criticità rilevanti. Tra queste, è emersa per esempio la presenza di dati nulli all'interno dei metadati, che aggregano domande e risposte associate a ciascuna immagine, e una distribuzione sbilanciata delle risposte uniche, con alcune categorie sovrarappresentate rispetto ad altre. Questi aspetti, pur rappresentando delle limitazioni, offrono spunti di riflessione sulle peculiarità del dataset, che saranno approfonditi nelle sezioni successive.

Imprecisioni testuali

Sono emerse delle incongruenze riguardanti le annotazioni di alcune domande o risposte, probabilmente dovute a errori di battitura o sviste, per esempio:

- Alcune righe riportano la dicitura "instrumnets" piuttosto che "instruments;
- Alcune righe riportano la dicitura ">20" piuttosto che ">20mm";
- Alcune righe riportano la dicitura "rigth" piuttosto che "right".

Si suppone che queste impurità verranno corrette con dei rilasci futuri di versioni più ricche e rifinite del dataset. Per l'attività di ricerca della tesi, queste righe sono state pulite e una copia del file in formato CSV bonificato verrà mantenuta ed utilizzata per addestramenti e test.

Presenza di dati nulli

All'interno del file contenente i metadati sono state ritrovate cinquantuno righe con dati nulli, che reputiamo un refuso del dataset e sono state eliminate prima di ulteriori elaborazioni, modifiche o considerazioni sul dataset.

La rimozione ha portato il numero effettivo di righe del dataset a 58.798, permettendoci di proseguire nella nostra ricerca.

Domande

Come possiamo vedere dalla tabella presentata precedentemente, sono presenti in totale diciannove domande distinte all'interno del dataset e con riferimenti ai tre principali punti di interesse che possiamo ritrovare nelle foto, ovvero anomalie, strumenti e *landmark* anatomici. Nel KvasirVQA possiamo trovare dunque domande riguardo:

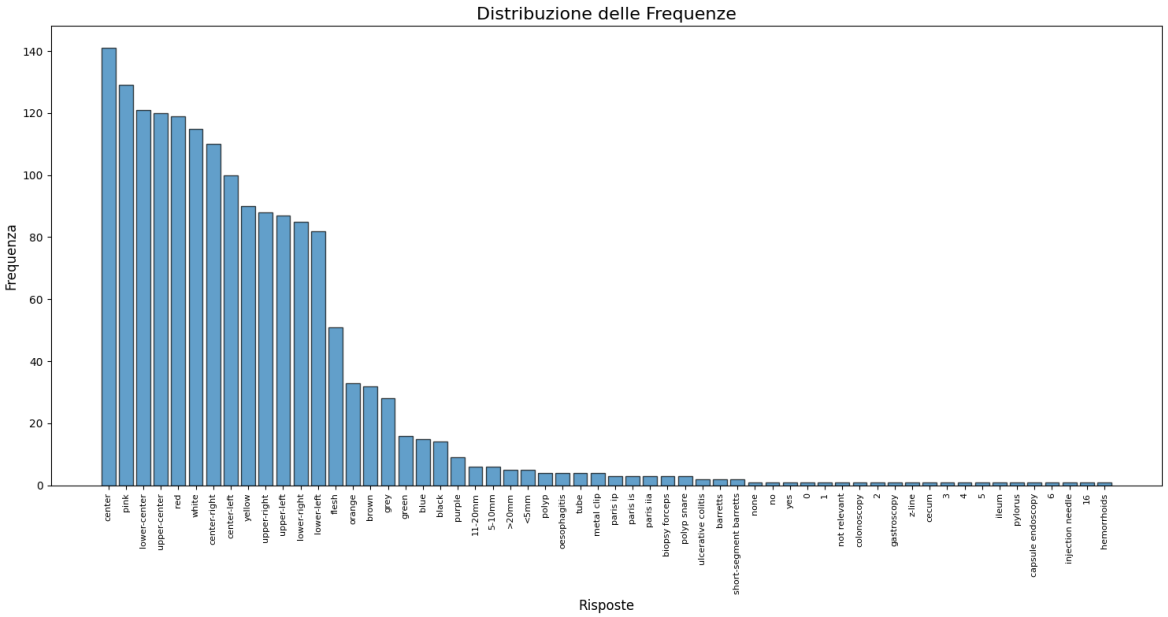
- Domande relative al posizionamento;

- Presenza nell'immagine;
- Conteggio degli oggetti d'interesse;
- Presenza artefatti software;
- Domande relative a caratteristiche dei ritrovamenti.

È stata inoltre verificata una lunghezza massima in parole di tredici per le domande, dato utile per stabilire eventuali parametri per la tokenizzazione della componente testuale.

Risposte e Data Augmentation

Prima della bonifica testuale, sono state rilevate all'interno del dataset 501 risposte distinte alle domande presenti, diventate 469 dopo la correzione delle imprecisioni che sono state notate durante lo studio del KvasirVQA. Parallelamente all'osservazione sulle 469 risposte uniche, è stato rilevato quello che possiamo considerare un vocabolario di, in totale, 58 opzioni uniche combinabili ad esempio nelle risposte a domanda multipla. Dato l'alto numero di risposte diverse, è stata ipotizzata una distribuzione sbilanciata di queste all'interno dei dati. Nonostante non sia totalmente rappresentativo della distribuzione delle risposte uniche, possiamo comprendere meglio lo sbilanciamento con il seguente grafico delle frequenze dei singoli vocaboli nelle risposte:



Questo dato è stato di particolare rilevanza nel trattamento del problema in forma chiusa, dal momento che è stato necessario generare sinteticamente nuovi dati a partire da quelli già presenti, distinguendo però l'operazione in due sotto-operazioni distinte, ovvero: *data augmentation* per righe con domande relative alla posizione, e *data augmentation* per tutti gli altri tipi di domanda. Comunemente a entrambe le procedure

è stato fatto in modo che ogni risposta comparisse comunque in almeno otto righe del dataset, così da fornire un minimo esempio, seppur esiguo, al modello, da cui imparare per una corretta classificazione riguardo le coppie di domande e risposte meno frequenti. Considerando una già menzionata soglia minima di otto righe come frequenza per ogni risposta, sono state generate in totale 2096 nuove righe. Sebbene la soglia sia esigua, per limitazioni computazionali non è stato possibile fornire una significativa rappresentazione a ognuna delle 496 risposte singole. Tuttavia, ipotizziamo che l'augmentation delle righe contenenti le risposte combinate possa essere una base solida. Le osservazioni sulla distribuzione dei dati e sulle frequenze sono state fondamentali, dal momento che da queste è scaturita la proposta sperimentale di porre il problema nella forma di un task di classificazione multilabel, approccio che verrà approfondito nel capitolo successivo.

Domande posizionali

Per quanto riguarda le domande posizionali, sono state applicate due trasformazioni distinte dal *framework PyTorch*:

- *ColorJitter*, modificando i parametri dell'immagine come luminosità, contrasto e saturazione con una variazione massima del dieci per cento, e la gradazione (*hue*) del cinque per cento;
- *GaussianBlur* per introdurre rumore all'interno delle immagini.

Altre domande

Per il resto delle domande, sono state applicate le seguenti trasformazioni:

- *RandomHorizontalFlip*, ovvero un ribaltamento orizzontale dell'immagine con probabilità che può avvenire con una probabilità p=0.5;
- *RandomRotation*, con una rotazione minima tra 0°e 270°.

4.3 Classificazione multilabel

La scelta di trasformare il task di classificazione multiclasse in classificazione multilabel nasce dall'analisi delle peculiarità del dataset KvasirVQA e delle caratteristiche intrinseche del task di VQA. Questa sezione illustra le motivazioni che hanno portato a questa decisione, i vantaggi attesi e le modifiche necessarie per implementare tale approccio.

Nel task di classificazione multiclasse, ogni combinazione di opzioni rappresenta una risposta unica e distinta. Tuttavia, questo approccio presenta delle limitazioni significative nel contesto del KvasirVQA:

- **Elevato numero di classi:** Il dataset presenta 469 risposte uniche generate da combinazioni di opzioni, aumentando significativamente la complessità del task e il rischio di *under-fitting*, specialmente con dataset di dimensioni limitate come quello con cui stiamo lavorando.
- **Relazioni tra le risposte:** Le risposte non sono sempre mutuamente esclusive. Ad esempio, alcune domande ammettono combinazioni di opzioni come risposte valide (ad esempio, presenza simultanea di più anomalie), rendendo più naturale il trattamento del problema come task multilabel.
- **Generalizzazione:** Un task multilabel permette al modello di apprendere in modo più flessibile, associando ogni opzione a un'etichetta binaria indipendente. Questo approccio facilita la generalizzazione a combinazioni non viste durante l'addestramento.

La trasformazione in classificazione multilabel consente di rappresentare ciascuna risposta come un vettore di etichette binarie, dove ogni posizione del vettore indica la presenza (1) o l'assenza (0) di una particolare opzione.

4.3.1 Vantaggi del task multilabel

Il passaggio a un task multilabel offre numerosi vantaggi pratici e metodologici:

- **Riduzione della complessità:** Trattando ogni opzione come etichetta indipendente, il modello si concentra sull'identificazione di caratteristiche specifiche, riducendo la complessità computazionale rispetto alla gestione di un numero elevato di classi distinte.
- **Flessibilità nelle risposte:** Il modello è in grado di prevedere combinazioni di opzioni non presenti nel set di addestramento, migliorando la capacità di adattarsi a nuovi scenari.
- **Mitigazione dello sbilanciamento:** La rappresentazione multilabel consente di bilanciare il peso delle opzioni meno rappresentate senza dover intervenire direttamente sulla distribuzione delle risposte complete.
- **Adattabilità a modelli generativi:** Questo approccio è concettualmente allineato a tecniche basate su modelli generativi e *templating*, che formulano le risposte come insiemi di opzioni selezionate.

4.3.2 Implementazione tecnica

Per implementare il task multilabel, sono state apportate le seguenti modifiche alla pipeline di addestramento:

- **Rappresentazione delle risposte:** Ogni risposta è stata trasformata in un vettore binario di lunghezza pari al numero di opzioni uniche (58). Ad esempio, una risposta contenente due opzioni sarebbe rappresentata come un vettore con due valori pari a 1 e i restanti pari a 0.
- **Funzione di perdita:** La *Binary Cross Entropy Loss* è stata adottata al posto della *Cross Entropy Loss*, per calcolare la perdita indipendentemente per ciascuna etichetta.
- **Modifica dell'output del modello:** Lo strato finale del modello è stato adattato per produrre logits per ciascuna delle 58 etichette, seguite da una funzione *sigmoid* per ottenere probabilità indipendenti.

Nonostante i vantaggi, l'approccio multilabel presenta alcune sfide:

- **Interpretazione delle soglie:** La scelta di una soglia per determinare la presenza di un'etichetta è cruciale e può influire significativamente sulle prestazioni del modello. Nelle implementazioni per questa tesi, la soglia è stata impostata a 0,5.
- **Dipendenza dai dati:** Il task multilabel richiede un dataset ben bilanciato per evitare che il modello ignori le etichette meno rappresentate.
- **Aumento delle risorse computazionali:** Sebbene il numero di classi sia ridotto, la necessità di calcolare probabilità per ciascuna etichetta può aumentare il costo computazionale.

4.4 Fine-tuning del modello ViLT sul dataset KvasirVQA

Il modello ViLT è stato scelto per il fine-tuning grazie alla sua architettura leggera e alla capacità di elaborare simultaneamente immagini e testo utilizzando un unico Transformer. A differenza di altri modelli multimodali che si basano su un backbone visivo separato, ViLT integra direttamente i token visivi e testuali in uno spazio comune, risultando più efficiente in termini di calcolo e memoria. Sono stati quindi eseguiti degli addestramenti e poi dei test per verificare la capacità del modello di performare nel contesto.

4.4.1 Preparazione dei dati

Elaborazione delle immagini

Le immagini del dataset KvasirVQA sono state preprocessate utilizzando il *ViLTProcessor* fornito dalla libreria *HuggingFace*. Questo strumento automatizza operazioni fondamentali come il ridimensionamento delle immagini, il padding e la normalizzazione dei pixel, garantendo una rappresentazione coerente degli input visivi.

In particolare, il metodo *pad* del *ViLTProcessor* è stato utilizzato per creare un padding uniforme, adattandole alle dimensioni della larghezza e altezza maggiori presenti nel batch e preservando le proporzioni originali dell'immagine. Questa operazione ha evitato distorsioni che avrebbero potuto compromettere l'accuratezza delle previsioni del modello.

Figure 4.1: Esempio di immagine con padding aggiunto dal ViLTProcessor

Elaborazione del testo

Le domande e risposte del dataset sono state tokenizzate utilizzando il tokenizer associato al Transformer di ViLT. Analizzando il vocabolario a disposizione del tokenizer del modello ViLT base scelto, sono state rilevate 22 parole non presenti, che sono state aggiunte prima dell'addestramento.

Nel dettaglio, i vocaboli che **non** erano presenti precedentemente sono i seguenti:

> gastroscopy / artefact / snare / barretts / colonoscopy / forceps / endoscopy / 20mm / polyp / ulcerative / pylorus / iia / ileum / 10mm / hemorrhoids / abnormality / biopsy / oesophagitis / colitis / 5mm / polyps / cecum

4.4.2 Configurazione del modello

Per il fine-tuning, è stato utilizzato il checkpoint pre-addestrato *"dandelin/vilt-b32-mlm"* disponibile su *HuggingFace*. L'architettura di ViLT è stata modificata per adattarla al task specifico del KvasirVQA nel seguente modo:

- Lo strato finale *fully connected* è stato rimodulato per produrre output corrispondenti al numero di risposte uniche nel dataset.
- Per il task di classificazione multiclasse, è stata utilizzata una funzione *Softmax* per trasformare i logits in una distribuzione di probabilità.
- Per il task multilabel, è stata adottata una funzione *sigmoid*, permettendo al modello di generare probabilità indipendenti per ciascuna etichetta.

4.4.3 Configurazione dell'addestramento

Iperparametri

Prima di procedere a elencare la configurazione scelta per l'addestramento, precisiamo che per il *learning rate* sono stati scelti valori volutamente molto bassi dal momento che la *backbone* del modello è rimasta intatta e soltanto la testa è stata riaddestrata per specializzarsi, permettendo quindi una maggiore stabilità in fase di addestramento e minori oscillazioni nella minimizzazione della funzione di perdita.

L'addestramento è stato configurato con i seguenti iperparametri, scelti tramite una serie di esperimenti preliminari:

- Learning rate: 4e-5
- Batch size: 16.
- Numero di epoche: 200.
- Ottimizzatore: AdamW.
- Scheduler: ReduceLROnPlateau.
- Pazienza: 5
- Min. Delta: 0.001
- Epoche Minime: 5

Questa scelta di iperparametri è risultata valida per entrambi gli approcci al problema, sia multiclasse che multilabel. Tuttavia, riteniamo che il fattore d'influenza principale dietro i risultati sia da attribuire ai dati in sé e alle criticità già menzionate nel capitolo relativo al KvasirVQA.

Loss function

Per il task multiclasse, è stata utilizzata la *Cross Entropy Loss*, mentre per il task multilabel è stata adottata la *Binary Cross Entropy Loss*.

Early Stop

Per evitare l'overfitting, è stato applicato un meccanismo di *early stop* che prevede una soglia di tolleranza, un delta e un minimo di epoche, in modo da interrompere l'addestramento in caso di mancati miglioramenti ripetuti, ma garantendo comunque un minimo di epoche prima dell'attivazione del meccanismo.

```
class EarlyStopper:
    def __init__(self, patience=1, min_delta=0, min_epochs=0):
        self.min_epochs = min_epochs
        self.patience = patience
        self.min_delta = min_delta
        self.counter = 0
        self.epoch_counter = 0
        self.min_val_loss = float('inf')

    def early_stop(self, val_loss):
        self.epoch_counter += 1

        if self.epoch_counter < self.min_epochs:
            return False

        if val_loss < self.min_val_loss:
            self.min_val_loss = val_loss
            self.counter = 0
        elif val_loss >= (self.min_val_loss + self.min_delta):
            self.counter += 1
            if self.counter >= self.patience:
                return True
        return False
```

Scheduler

Oltre all'utilizzo del meccanismo di early stop per la regolarizzazione, sono stati testati vari *scheduler* per la gestione del learning rate, messi a disposizione da *PyTorch*. Quello che è risultato essere più valido dai test effettuati è stato lo scheduler *ReduceLROnPlateau*, per la riduzione del learning rate nelle fasi di stallo dell'addestramento.

4.5 Architettura custom

L'obiettivo dietro la proposta di un'architettura personalizzata è analizzare come un modello costruito su misura possa affrontare le peculiarità del dataset, in particolare le domande che combinano componenti testuali e visive complesse. In particolare è stata pensata un'architettura ispirata ai metodi tradizionali di fusione multimediale dei *joint-embeddings* della componente testuale e visiva, distinguendo poi il task in classificazione multiclasse della risposta o multilabel, per verificare l'efficienza dei due diversi metodi. Prendendo come punto di partenza l'architettura pensata in [3], ne è stata pensata una

versione che potesse adattarsi e affrontare le peculiarità del KvasirVQA, quali scarsezza di dati e di annotazioni precise per le immagini.

4.5.1 Architettura

L'architettura custom proposta si basa su una combinazione di un estrattore di caratteristiche visive e un modulo di elaborazione testuale, integrati tramite una fase di fusione multimodale. Gli elementi principali dell'architettura sono un estrattore di caratteristiche visive (*feature extractor*) e un modulo per l'elaborazione testuale.

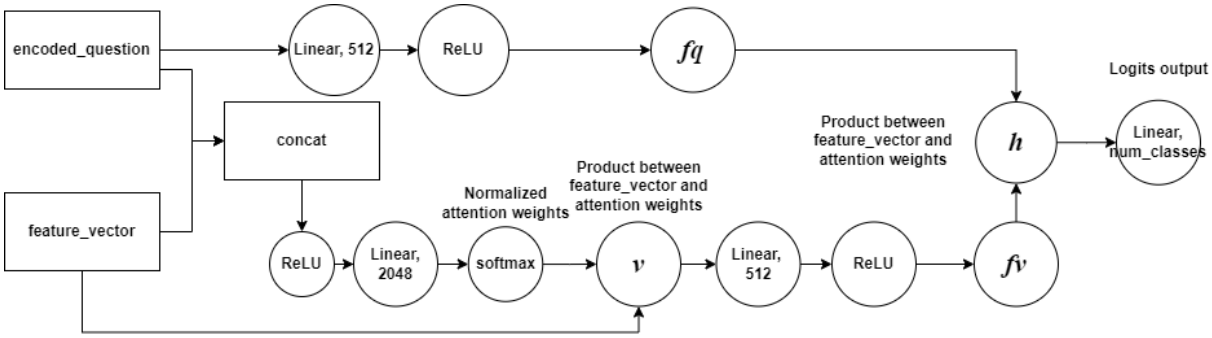


Figure 4.2: Schema riassuntivo dell'architettura completa

Estrattore di Caratteristiche Visive

Per la scelta dell'estrattore visivo, sono stati testati diversi modelli pre-addestrandoli nel riconoscimento delle immagini di dataset originali *HyperKvasir* e *Kvasir Instrument*, scegliendo infine quello con la percentuale di accuracy più alta in base all'F1 macro score. Il migliore tra questi modelli, ovvero una rete *ResNet50*, è stato utilizzato quindi come backbone per l'estrazione delle features visive globali dall'immagine.

Modulo di Elaborazione Testuale

Per la codifica delle domande, è stato utilizzato un modello *BERT-like* [2], specializzato in ambito medico e quindi ritenuto adatto al compito da affrontare. Le sequenze di parole vengono quindi tokenizzate e trasformate in embedding per ottenere una rappresentazione contestualizzata delle domande.

I token non presenti tra quelli conosciuti sono stati aggiunti, e sono i seguenti:

> polyp / barretts / polyps / oesophagitis / 10mm / cecum / ip / colitis / abnormality / abnormalities / forceps / anatomical / gastroscopy / biopsy / colonoscopy / ileum / 5mm / hemorrhoids / endoscopy / 20mm / ulcerative / paris / artefact / pylorus / iia / snare

Attenzione normalizzata

Nel tentativo di replicare un meccanismo di attenzione funzionante e che sfruttasse gli input sia visivi che testuali, è stato integrato l'utilizzo dell'attenzione normalizzata nel seguente modo:

Concatenando il vettore ottenuto dalla ResNet v di dimensione 2048 insieme alla codifica del testo q di dimensioni 768. Il vettore risultante α viene quindi riproiettato per avere una dimensione uguale a V, normalizzato utilizzando una funzione softmax e infine viene calcolato il prodotto tra α e v.

Fusione Multimodale

Dopo aver applicato il meccanismo di attenzione normalizzato su v, è possibile infine procedere alla fusione multimodale di caratteristiche visive e testuali. Facciamo dunque passare q e v entrambi separatamente per layer non lineari e lineari per riproiettarli con dimensioni uguali, moltiplichiamoli e usiamo l'output come input per l'ultimo layer di classificazione.

Strato Finale di Classificazione

L'ultimo layer che si occupa della classificazione è un layer lineare di dimensione variabile in base al task multiclasse o multilabel, con dimensione pari al numero di classi nel dataset. In ogni caso, l'utilizzo è quello di effettuare classificazione sulle logits, variando la funzione di attivazione in uscita:

Per il task multilabel, è stata utilizzata una funzione di attivazione *sigmoid*, mentre per il task multiclasse si è utilizzata una funzione *softmax*.

4.5.2 Configurazione dell'Addestramento

Il modello è stato addestrato utilizzando **PyTorch**, con i seguenti parametri principali:

- Learning rate: 4e-4
- Batch size: 32
- Numero di epoche: 200.
- Ottimizzatore: AdamW.
- Scheduler: ReduceLROnPlateau.
- Pazienza: 5
- Min. Delta: 0.001
- Epoche Minime: 5

4.5.3 Visualizzazione componente visiva

Per una comprensione migliore del funzionamento delle architetture custom, è stato utilizzato l'algoritmo Grad-CAM per generare le heatmap e visualizzare le attivazioni del modello. In particolare, sono state generate le immagini ottenute dal feature extractor già menzionato. Alcuni esempi saranno presentati insieme agli esperimenti nel prossimo capitolo, generati grazie alla libreria PyTorch dedicata [10].

4.6 Prompting applicato al KvasirVQA

In questa sezione verranno descritti gli approcci basati sul prompting utilizzati per esplorare le prestazioni di diversi modelli multimodali sul dataset KvasirVQA. Sono state applicate due tecniche principali di prompting, *templating* e CoT, al fine di migliorare la comprensione testuale e il ragionamento visivo-linguistico dei modelli. Le tecniche di prompting applicate sono state studiate per tentare innanzitutto di guidare i modelli nella generazione di una risposta coerente con il contesto, cercando di instradarli tramite presentazione di opzioni per la risposta o tramite l'esplicitazione del ragionamento fatto per arrivare a una soluzione.

4.6.1 Modelli Utilizzati

Le tecniche di prompting sono state testate su una selezione di modelli multimodali descritti nel capitolo sullo stato dell'arte:

- BLIP
- GIT
- LLaVA

4.6.2 Templating

Il *templating* è stato applicato utilizzando una serie di schemi predefiniti per standardizzare l'interazione tra input testuali e modelli. Questa tecnica mira a fornire al modello una struttura coerente per comprendere il task e generare risposte.

Template Utilizzati

Di seguito alcuni esempi di template adottati:

```
Template-1:

You are given the following question:
Question: <q>
```

```
Below are the possible options:
Options: <o>
Respond using only the options provided. If multiple options
    apply, separate them with a semicolon (';').

Template-2:

Based on the following question:
Question: <q>
Choose the most appropriate answer(s) from the provided
    options:
Options: <o>
Please respond using one or more of the options, separated by
    a semicolon (';'), if applicable.
Answer:
```

4.6.3 Chain-of-Thought (CoT)

La tecnica *Chain-of-Thought* è stata impiegata per stimolare il ragionamento passo-passo dei modelli. Questa tecnica guida il modello a suddividere il problema in passaggi logici prima di fornire una risposta.

Prompt per il Chain-of-Thought

I prompt per il CoT sono stati progettati per incoraggiare i modelli a esplicitare il ragionamento. Alcuni esempi includono:

```
CoT-1:

Q: <q>
Options: <o>
A: Let's think step-by-step.

CoT-2:

Q: <q>
Options: <o>
A: Let's think step-by-step.
1. First, identify the key information in the question.
2. Next, relate this information to the available options.
3. Eliminate any options that do not align with the question.
```

```
4. Finally, based on the analysis, respond with the most
   appropriate option(s), separated by a semicolon (';') if
   multiple answers apply.
Answer:
```

I risultati dettagliati delle tecniche di prompting applicate ai modelli elencati verranno discussi nel capitolo dedicato agli esperimenti, con analisi comparative e approfondimenti sull'efficacia delle diverse configurazioni.

Capitolo 5

Esperimenti

5.1 Introduzione

In questo capitolo verranno descritti gli esperimenti effettuati utilizzando modelli e tecniche proposti nel capitolo precedente per valutare le prestazioni sul dataset KvasirVQA, con un approfondimento sulla parte tecnica e sulle implementazioni adottate.

5.2 Setup degli esperimenti

5.2.1 Configurazione hardware

L'addestramento è stato eseguito su un computer con una configurazione non professionale né ottimale per il compito:

- **GPU**: NVIDIA RTX3060Ti con 8GB di memoria, sfruttando il supporto per il calcolo parallelo che ha permesso di ridurre i tempi di calcolo sfruttando i CUDA cores.
- **CPU**: Intel i7 12700KF.
- **RAM**: 32GB di RAM DDR4.

5.2.2 Configurazione software

Per quanto riguarda la configurazione software, sono state utilizzate in primis le librerie solitamente più gettonate in ambito *Machine Learning* e *AI*, ovvero:

- *Scikit-learn* per la possibilità di richiamare le API funzionali per il calcolo delle metriche, creazione di *classification report* e divisione dei dati in split per gli addestramenti.
- *Seaborn e Matplotlib* per la visualizzazione di dati e grafici, in particolare dei report creati dalle librerie di *Scikit-learn*.

- *PyTorch* per la scrittura di modelli personalizzati, manipolazione dei tensori, implementazione di metodi per inferenza e creazione dei loop di addestramento.
- *HuggingFace* per la possibilità di avvalersi dei modelli caricati sulla piattaforma, ma anche delle API per il recupero dei dataset e utilizzo dei modelli transformers e i loro tokenizzatori.
- *NumPy e Pandas* per la manipolazione delle strutture dati, in particolare dei *DataFrame* Pandas.
- *NLTK e pycocoevalcap* per operazioni sul testo come tokenizzazione e calcolo del CIDEr score.

Infine, per sfruttare al meglio la configurazione hardware e in particolare la GPU per l'accelerazione hardware, è stato configurato un ambiente software con CUDA versione 12.6. Per un fattore di riproducibilità, ove possibile, è stato impostato manualmente il valore di *seed* uguale a 42.

5.3 Preprocessing dei dati

In base alla dimensionalità degli input delle varie architetture, sono state necessarie delle operazioni di preprocessing, più o meno simili tra loro. Inoltre, grazie alle operazioni di data augmentation descritte, è stato possibile effettuare addestramenti e poi esperimenti stratificando gli split di addestramento, test e validazione in base alle domande. In particolare, il 70% dei dati del KvasirVQA è stato utilizzato per l'addestramento, il restante 30% è stato ripartito in parti uguali per la validazione dei modelli e per i test.

5.3.1 ViLT

Come accennato nel capitolo precedente, l'operazione di preprocessing in questo caso viene svolta avvalendosi del processor di ViLT, utilizzabile grazie alle API di *HuggingFace*, che va quindi a effettuare una normalizzazione dell'immagine e ad aggiungere una maschera per il padding in base alle dimensioni del resto delle immagini nella batch.

5.3.2 Modelli custom

Per le architetture custom che sono state descritte nello scorso capitolo, si è scelto di adottare un approccio di preprocessing comune, ovvero una pipeline con le seguenti trasformazioni provenienti dalla libreria *v2* di *torchvision.transforms*:

- **v2.Resize**: ridimensionamento delle immagini a 224x224, per l'adattamento alle dimensioni di input delle reti.
- **v2.toDtype** e **v2.Normalize**: per la trasformazione dei valori dei pixel.

5.3.3 Altre architetture

Per il preprocessing dei dati da fornire in input agli altri modelli, sono stati utilizzati i processor associati e disponibili su HuggingFace, che coprono autonomamente la parte relativa al resize dell'immagine, aggiunta eventuale di maschere per il padding, normalizzazione del valore dei pixel ed eliminazione di bordi indesiderati.

5.4 Metriche di valutazione

Per la valutazione della bontà dei modelli scelti, sono stati utilizzati due insiemi di metriche, distinguendo per i task di generazione del testo assistito da prompting e classificazione.

5.4.1 Task di classificazione

Per la valutazione dei task di classificazione è stata scelta la metrica dell'*F1 Score*, una metrica di valutazione utilizzata per misurare l'equilibrio tra *Precision* e *Recall* in un modello di classificazione. È particolarmente utile quando, come nel nostro caso, le classi sono sbilanciate, poiché fornisce una media armonica di Precision e Recall.

La formula dell'F1 Score è la seguente:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Dove:

- **Precision:** La frazione di esempi predetti positivi che sono effettivamente positivi.
- **Recall:** La frazione di esempi positivi correttamente identificati come tali.

L'F1 Score assume valori compresi tra 0 e 1, per cui valori più vicini a 1 indicano un buon equilibrio tra Precision e Recall, mentre valori più bassi suggeriscono che il modello è meno efficace nel bilanciare i falsi positivi e i falsi negativi.

Nel contesto della classificazione multilabel, l'F1 Score può essere calcolato per ogni classe e successivamente aggregato come media macro o pesata in base alla frequenza delle classi. I risultati mostrati saranno il risultato dell'F1 Score con media macro.

5.4.2 Task generativi

Per i task di generazione del testo, sono state scelte in particolare le stesse metriche presenti nel paper originale del KvasirVQA, in modo da fornire un confronto coerente con le prestazioni dichiarate dagli autori che si sono avvalsi di un modello *Florence-2* riaddestrato sui propri dati.

BLEU

BLEU [25] è una metrica automatica utilizzata per valutare la qualità dei testi generati da modelli di linguaggio o traduzione rispetto a un riferimento umano. Misura la somiglianza tra il testo generato e uno o più testi di riferimento, calcolando il numero di n-grammi condivisi. Sfruttando gli n-grammi, ovvero sequenze di n elementi uguali da un testo, è indipendente dalla lingua scelta e favorisce la corrispondenza tra quegli n-grammi di lunghezza maggiore, come i bi e trigrammi. Inoltre, è progettato in modo da essere uno score normalizzato da 0 a 1 o da 0 a 100 in caso di scala percentuale, per una maggiore interpretabilità del punteggio. Non avendo tuttavia comprensione della semantica, non gestisce bene risposte semanticamente simili o che rappresentano una parafrasi della risposta reale.

Formula:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log P_n\right)$$

Dove:

- P_n: precisione degli n-grammi condivisi.
- w_n: peso associato a ogni n-gramma (spesso uguale per tutti).
- BP: brevity penalty, penalizza risposte troppo brevi rispetto al riferimento.

METEOR

METEOR [16] è una metrica progettata per valutare la qualità delle traduzioni automatiche confrontando il testo generato con una traduzione di riferimento.

Grazie alla sua capacità di riconoscere variazioni lessicali e semantiche, METEOR è ampiamente utilizzata in campi come la traduzione automatica e il riepilogo testuale. Rispetto ad altre metriche, METEOR è particolarmente sensibile alle differenze linguistiche e alle variazioni di formulazione, risultando più rappresentativa della qualità percepita dall'utente finale e dimostra una maggiore potenzialità nel riconoscere variazioni lessicali e semantiche.

Formula La metrica combina precisione (P) e richiamo (R) utilizzando una media armonica pesata, calcolata come:

$$F_{\text{mean}} = \frac{10 \cdot P \cdot R}{9 \cdot P + R}.$$

Successivamente, viene applicata una penalità per penalizzare disordini strutturali nella traduzione:

$$\text{METEOR} = F_{\text{mean}} \cdot (1 - P_{\text{penalty}}).$$

ROUGE

ROUGE [20] è una metrica comunemente utilizzata per valutare la qualità dei riassunti automatici e di altri compiti di generazione testuale. Si basa sul confronto tra le unità linguistiche (ad esempio, parole o frasi) generate automaticamente e quelle presenti in un testo di riferimento, privilegiando il richiamo (*recall*) come indicatore principale.

La famiglia di metriche ROUGE comprende diverse varianti, quelle che sono state utilizzate all'interno della tesi, in particolare, sono:

- **ROUGE-1**, che calcola il richiamo di unigrams (parole singole) condivisi tra il testo generato e il riferimento. È particolarmente utile per valutare la copertura lessicale di base.
- **ROUGE-2**, che misura il richiamo di bigrammi (sequenze di due parole consecutive), offrendo un'analisi più dettagliata della fluidità e della continuità testuale.
- **ROUGE-L**, che utilizza la lunghezza della sottosequenza comune più lunga (*Longest Common Subsequence*, LCS) per valutare la similarità strutturale tra i testi. Questa variante tiene conto della coerenza globale.
- **ROUGE-L-SUM**, una variante di ROUGE-L che è specificamente progettata per valutare riassunti generati automaticamente, considerando simultaneamente l'allineamento strutturale e la copertura delle informazioni chiave.

Ad esempio, ROUGE-1 e ROUGE-2 calcolano il richiamo rispettivamente di unigrams e bigrammi:

$$\text{ROUGE-N} = \frac{\text{n-grammi comuni}}{\text{n-grammi nel riferimento}}.$$

ROUGE-L, invece, si concentra sulla lunghezza della sottosequenza comune più lunga, considerando l'ordine delle parole e non solo la loro presenza. Questa metrica è particolarmente utile nei casi in cui si voglia valutare la capacità del modello di mantenere una struttura coerente con il riferimento.

ROUGE-L-SUM rappresenta un'estensione specifica per i riassunti e bilancia l'importanza delle informazioni chiave e della struttura.

Nel contesto della tesi, queste varianti sono state utilizzate per analizzare diverse sfaccettature delle risposte generate dai modelli. ROUGE-1 e ROUGE-2 valutano la corrispondenza lessicale e la fluidità locale, mentre ROUGE-L e ROUGE-L-SUM analizzano la coerenza strutturale e l'adeguatezza delle informazioni. I dati riportati nelle tabelle per la metrica ROUGE riguarderanno in particolare ROUGE-1.

CIDEr

CIDEr [29] è una metrica progettata per valutare la qualità delle descrizioni generate automaticamente per le immagini. Si basa sul confronto tra una descrizione generata

e un insieme di descrizioni di riferimento, ponendo un'enfasi particolare sul consenso tra le descrizioni umane.

La metrica calcola un punteggio basato sulla similarità tra n-grammi della descrizione generata e delle descrizioni di riferimento, ponderando ciascun n-gramma in base alla sua frequenza nei dati di riferimento. Nello specifico, CIDEr assegna maggior peso agli n-grammi che appaiono frequentemente nelle descrizioni di riferimento, ma che non sono troppo comuni nei testi generali, utilizzando il *Term Frequency-Inverse Document Frequency* (TF-IDF) come schema di ponderazione. TF-IDF misura l'importanza di un termine nel contesto di un documento specifico, penalizzando i termini troppo frequenti nel corpus (che portano poca informazione discriminante) e premiando quelli rari, ma significativi.

Il calcolo del punteggio CIDEr può essere riassunto nei seguenti passaggi:

- Ogni descrizione viene rappresentata come un insieme di n-grammi, con n che varia tra 1 e 4.
- Per ciascun n-gramma, viene calcolata una ponderazione basata sul TF-IDF, che misura la sua rilevanza nel contesto delle descrizioni di riferimento.
- Si calcola la similarità tra le descrizioni generate e quelle di riferimento utilizzando la somma ponderata delle similarità degli n-grammi condivisi.
- Infine, il punteggio viene mediato su tutte le descrizioni di riferimento per ottenere una misura complessiva di consenso.

CIDEr è particolarmente adatto per task di descrizione delle immagini, poiché tiene conto sia della specificità del linguaggio utilizzato sia della coerenza semantica rispetto ai riferimenti umani.

Formula La formula del CIDEr per un singolo n-gramma è data da:

$$\text{CIDEr}(c, S) = \frac{1}{|S|} \sum_{s \in S} \frac{\text{TF-IDF}(c) \cdot \text{TF-IDF}(s)}{\|\text{TF-IDF}(c)\| \|\text{TF-IDF}(s)\|},$$

dove c rappresenta la descrizione generata, S l'insieme delle descrizioni di riferimento e TF-IDF è il peso assegnato a ciascun n-gramma.

[29]

5.5 Risultati

A seguire, mostreremo i risultati sperimentali ottenuti testando le varie configurazioni. Distingueremo le configurazioni che hanno previsto un addestramento, come nel caso di ViLT e dell'architettura personalizzata, dalle configurazioni che hanno sfruttato tecniche di prompting: verranno chiaramente fatte valutazioni distinte in base alle

metriche utilizzate e alle differenze riscontrate tra gli addestramenti e la fase di inferenza, dal momento che queste differenze sono ritenute un riflesso della distribuzione sbilanciata dei dati all'interno del dataset.

5.5.1 ViLT - Multiclasse

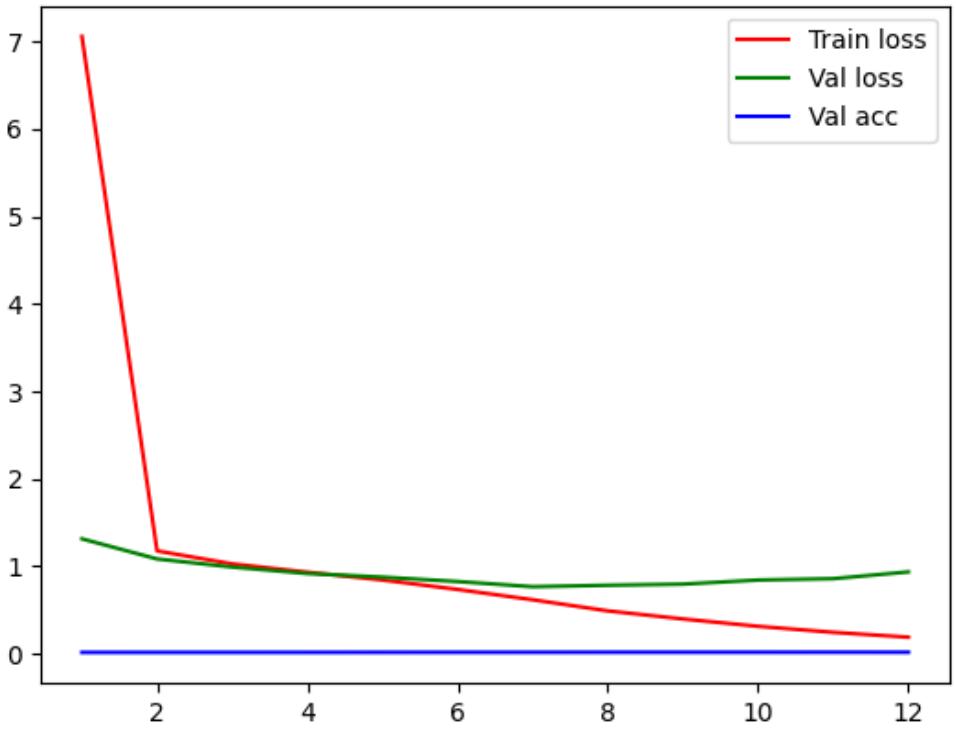


Figure 5.1: Grafico dell'andamento dell'addestramento multiclasse per il modello ViLT based

F1-Score: 0.5955

5.5.2 ViLT - Multilabel

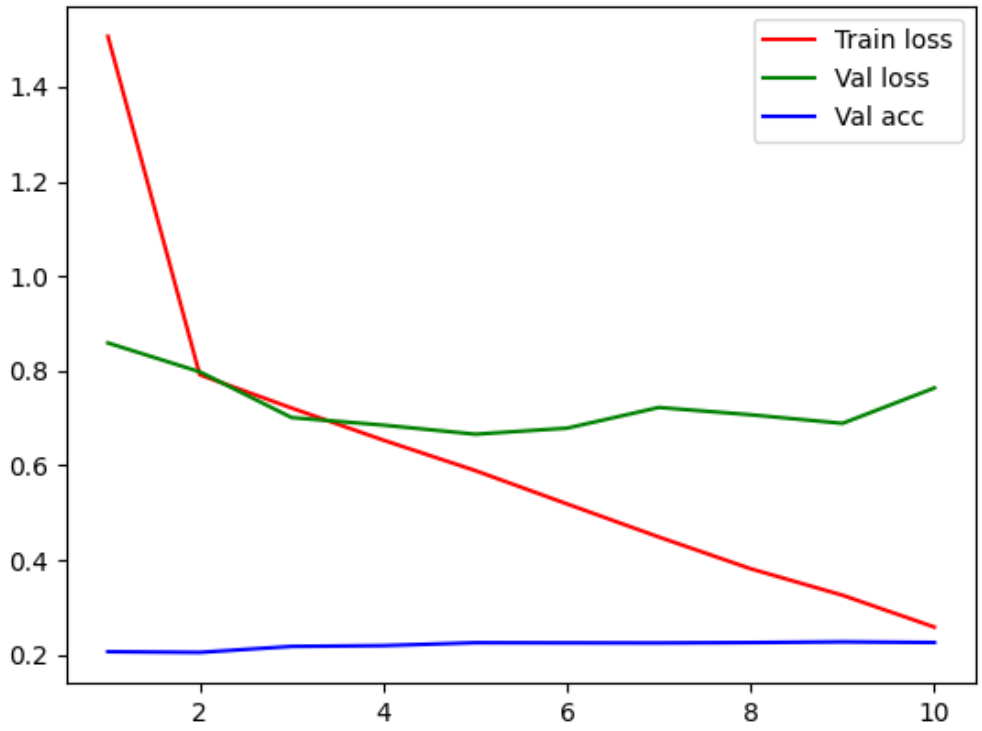


Figure 5.2: Grafico dell'andamento dell'addestramento multilabel per il modello ViLT based

F1-Score: 0.5369

5.5.3 Custom - Multiclasse

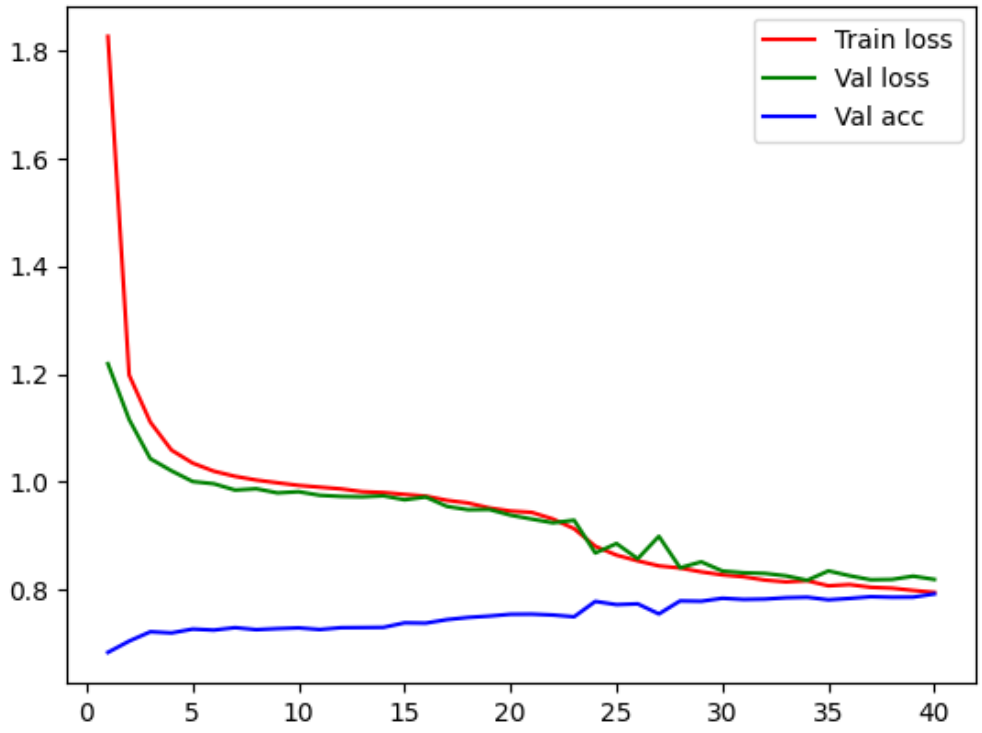


Figure 5.3: Grafico dell'andamento dell'addestramento dell'architettura custom per task di classificazione multiclasse

F1-Score: 0.0369

5.5.4 Custom - Multilabel

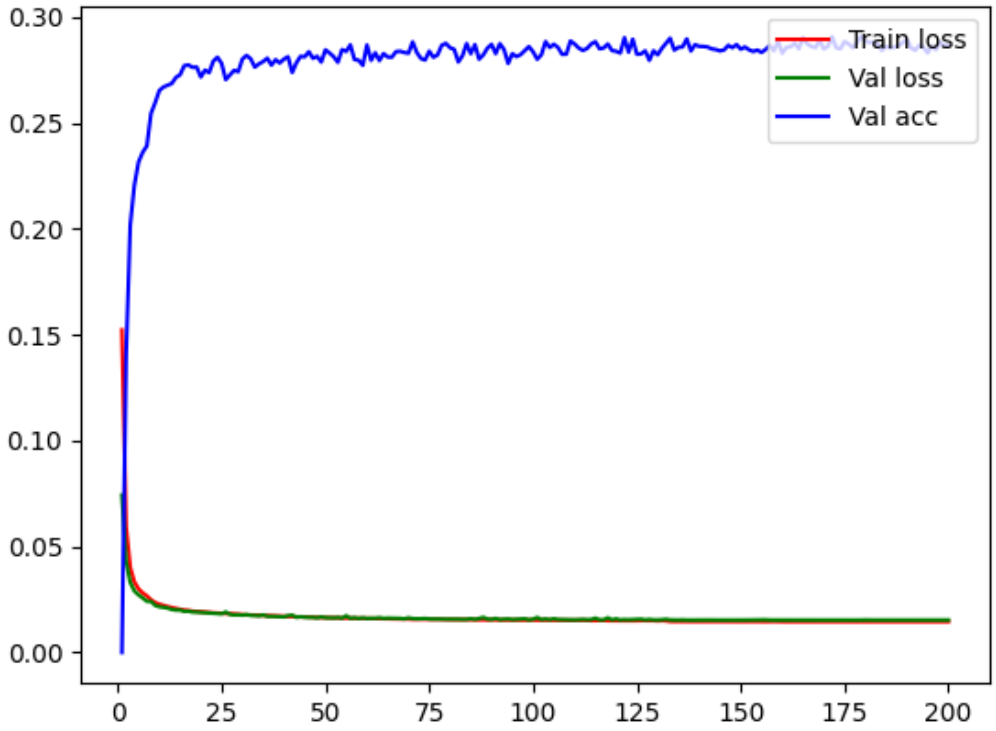


Figure 5.4: Grafico dell'andamento dell'addestramento dell'architettura custom per task di classificazione multilabel

F1-Score: 0.3944.

5.5.5 Classificazione

Modello	Task	F1-Score
Custom	Multiclasse	0.0369
Custom	Multilabel	0.3944
ViLT	Multiclasse	0.5369
ViLT	Multilabel	0.5955

Table 5.1: Tabella riassuntiva delle prestazioni dei vari modelli usati per task di classificazione

La Tabella 5.1 presenta un riepilogo delle prestazioni ottenute dai modelli testati nei task di classificazione multiclasse e multilabel, valutati tramite l'F1-Score. I risultati mettono in evidenza significative differenze tra le due configurazioni di task e tra le architetture considerate.

Il modello *Custom*, progettato per essere specifico al dataset KvasirVQA, ha mostrato un F1-Score particolarmente basso nel task di classificazione multiclasse (0.0369), evidenziando difficoltà nel catturare le relazioni tra le immagini e le etichette di risposta in un contesto altamente discreto come quello multiclasse. Questo risultato può essere attribuito alla natura complessa del dataset, caratterizzata da una distribuzione di classi estremamente sbilanciata e da un numero elevato di risposte possibili, che ricordiamo essere 469. Inoltre, si osserva una discrepanza notevole tra l'accuracy durante la fase di addestramento e quella nella fase di test, suggerendo che il modello fatichi a generalizzare in questa configurazione.

Nel task di classificazione multilabel, il modello *Custom* ha ottenuto un F1-Score pari a 0.3944, dimostrando un netto miglioramento rispetto alla configurazione multiclasse. Questo punteggio riflette il vantaggio dell'approccio multilabel nel trattare risposte che coinvolgono combinazioni multiple di classi, più adatte alla struttura del dataset in analisi. Il miglioramento ottenuto conferma che il task multilabel consente di rappresentare in modo più accurato le complessità intrinseche del problema, riducendo al contempo le difficoltà legate alla rigidità delle classificazioni multiclasse.

Il modello *ViLT* ha ottenuto risultati nettamente migliori rispetto al *Custom*, sia nel task multiclasse (F1-Score di 0.5369) sia in quello multilabel (F1-Score di 0.5955). In particolare, il passaggio al task multilabel ha prodotto un miglioramento significativo, dimostrando che questa configurazione consente al modello di esprimere in modo migliore il suo potenziale. Tuttavia, anche nel caso di ViLT, si osserva una certa discrepanza tra l'accuracy in fase di addestramento e quella nella fase di test. Questo fenomeno, condiviso con il modello *Custom*, è attribuibile alle caratteristiche del dataset.

Questi risultati sottolineano come la scelta del task e dell'architettura influenzi in modo significativo le performance. L'approccio multilabel si è dimostrato più adatto per il dataset KvasirVQA, grazie alla maggiore flessibilità nel rappresentare risposte complesse e composizioni multiclasse. Sebbene il modello *Custom* abbia mostrato un miglioramento apprezzabile nel task multilabel, il divario rispetto a *ViLT* evidenzia la necessità di ulteriori ottimizzazioni, sia a livello architetturale che di pre-elaborazione dei dati. Inoltre, il confronto mette in luce il valore aggiunto offerto dai modelli pre-addestrati su dataset di grandi dimensioni, i quali sono in grado di generalizzare meglio a problemi specifici anche in presenza di dataset complessi e sbilanciati come il KvasirVQA.

5.5.6 Prompting

Modello	Prompting	BLEU	ROUGE	CIDEr	METEOR
LLaVA	template-1	0.0	0.3692	0.0	0.1961
LLaVA	template-2	0.0	0.3797	0.0	0.2041
LLaVA	cot-1	0.0	0.1364	0.0	0.0617
LLaVA	cot-2	0.0	0.3893	0.0	0.2034
GIT	template-1	0.0	0.059	0.0	0.0288
GIT	template-2	0.0	0.0033	0.0	0.001
GIT	cot-1	0.0	0.0003	0.0	0.0001
GIT	cot-2	0.0	0.0046	0.0	0.0002
BLIP	template-1	0.0	0.1266	0.0	0.1266
BLIP	template-2	0.0	0.158	0.0	0.0798
BLIP	cot-1	0.0	0.1009	0.0	0.0441
BLIP	cot-2	0.0	0.1192	0.0	0.0606

Table 5.2: Tabella riassuntiva dei risultati dei modelli con tecniche di prompting

Analisi delle performance per modello

Il **modello LLaVA** si distingue come il più performante, raggiungendo i migliori punteggi per ROUGE e METEOR in quasi tutte le configurazioni. In particolare, con il prompting basato su template-2, LLaVA ottiene un ROUGE pari a 0.3797 e un METEOR di 0.2041, dimostrando un'elevata coerenza e rilevanza semantica nelle risposte generate. Anche utilizzando CoT-2, il modello mantiene prestazioni competitive, con un ROUGE pari a 0.3893 e un METEOR di 0.2034, mostrando una certa versatilità nell'adattarsi a diversi tipi di prompting.

Il **modello BLIP** presenta risultati più modesti rispetto a LLaVA, ma comunque significativi. Con il prompting template-2, raggiunge un ROUGE di 0.158 e un METEOR di 0.0798, dimostrando una discreta capacità di generare risposte contestualmente rilevanti. Tuttavia, i risultati con CoT sono inferiori, con un massimo di 0.1192 per ROUGE e 0.0606 per METEOR, suggerendo che il modello non beneficia pienamente di questa tecnica.

Infine, il **modello GIT** registra i risultati più bassi in tutte le metriche. Anche nella configurazione migliore (template-1), ottiene un ROUGE di appena 0.059 e un METEOR di 0.0288. I punteggi estremamente bassi con CoT (ROUGE massimo di 0.0046 e METEOR massimo di 0.0002) indicano difficoltà significative nel comprendere e generare risposte coerenti, indipendentemente dal prompting utilizzato.

Confronto tra template e Chain-of-Thought

Le tecniche di prompting basate su template si dimostrano generalmente più efficaci rispetto a quelle basate su Chain-of-Thought. Per LLaVA, ad esempio, template-2 ottiene punteggi leggermente superiori rispetto a CoT-2 in METEOR (0.2041 contro 0.2034) e ROUGE (0.3797 contro 0.3893), pur mantenendo un margine molto ridotto. BLIP evidenzia una differenza più marcata, con template-2 che supera CoT sia in ROUGE che in METEOR, indicando che il modello risponde meglio a prompt strutturati piuttosto che a strategie di ragionamento esplicito. GIT, al contrario, non beneficia significativamente di nessuna delle tecniche, confermando le sue limitazioni.

Osservazioni generali

L'assenza di punteggi positivi per BLEU e CIDEr solleva interrogativi sulla capacità dei modelli di generare risposte strettamente correlate ai riferimenti testuali. Questo fenomeno potrebbe essere attribuito a un disallineamento tra i prompt utilizzati e la struttura delle risposte attese, o a limiti intrinseci nei modelli testati.

Inoltre, i punteggi relativamente bassi per ROUGE e METEOR, specialmente per BLIP e GIT, indicano che i modelli non riescono a catturare completamente la complessità semantica richiesta dal task. Tuttavia, LLaVA si distingue come il modello più promettente, suggerendo che ulteriori ottimizzazioni, come un miglioramento del design dei template o un fine-tuning specifico, potrebbero ulteriormente migliorare le sue prestazioni.

In conclusione, LLaVA si dimostra il modello più efficace per i task generativi testati, mentre BLIP e, soprattutto, GIT mostrano prestazioni limitate. Le tecniche di templating emergono come la scelta più solida rispetto a Chain-of-Thought, evidenziando la necessità di un prompting ben strutturato per massimizzare le capacità dei modelli in contesti complessi. Tuttavia, l'assenza di risultati significativi per BLEU e CIDEr sottolinea l'importanza di ulteriori analisi e ottimizzazioni.

5.5.7 Criticità

Durante l'utilizzo delle tecniche di prompting, sono emerse diverse criticità che hanno evidenziato le sfide di questo approccio. Una delle principali difficoltà è stata la formulazione dei prompt, sia per quanto riguarda il *templating* che il *Chain-of-Thought* (CoT). Per il *templating*, la sfida era creare una struttura standardizzata e sufficientemente chiara per guidare il modello nella generazione di risposte pertinenti. Piccole variazioni nella formulazione, come l'ordine delle parole o la presentazione delle opzioni, hanno avuto un impatto significativo sulle prestazioni, mostrando una forte dipendenza del modello dalla struttura del prompt. Nel caso del CoT, l'equilibrio tra un livello di dettaglio sufficiente per supportare il ragionamento passo-passo e la sintesi necessaria per evitare ridondanze si è dimostrato complesso da raggiungere. Prompt troppo det-

tagliati hanno portato a risposte prolisse e inefficaci, mentre prompt più sintetici spesso non fornivano abbastanza supporto per una corretta elaborazione del ragionamento. Un ulteriore problema significativo è stato il fenomeno delle *allucinazioni*, ovvero risposte slegate dai dati di input o dalle istruzioni fornite. Questo è risultato particolarmente evidente in task con domande aperte o opzioni multiple, dove il modello, pur essendo guidato da prompt che specificavano di rispondere utilizzando solo le opzioni fornite, talvolta generava risposte che non appartenevano al vocabolario previsto. Infine, un aspetto non trascurabile è stato rappresentato dai costi computazionali. Sebbene il prompting eviti il processo di *fine-tuning*, il tempo necessario per iterare e testare diverse versioni di prompt, combinato con l'elaborazione passo-passo richiesta dal CoT, ha portato a un utilizzo significativo di risorse. Questo sottolinea come, nonostante i vantaggi in termini di flessibilità e scalabilità, l'adozione di tecniche di prompting richieda comunque un impegno computazionale rilevante.

Per quanto riguarda invece il fine-tuning dei modelli, sono emerse criticità altrettanto significative legate principalmente alla dipendenza da un dataset robusto e corposo. L'addestramento dei modelli si è rivelato fortemente influenzato dalla qualità e dalla quantità dei dati disponibili, evidenziando che dataset con distribuzioni sbilanciate o categorie sottorappresentate possono limitare significativamente le capacità generative e di generalizzazione del modello. Questo aspetto si è mostrato particolarmente problematico nel caso di task di classificazione multiclasse, dove l'elevato numero di classi ha richiesto al modello di apprendere un'enorme varietà di combinazioni di risposte, aumentando notevolmente la complessità del processo di addestramento. La presenza di classi con pochissimi esempi ha portato inoltre a una maggiore probabilità di errore, richiedendo strategie di *data augmentation* per migliorare l'equilibrio tra le categorie. Un'altra criticità significativa riguarda la rigidità dei modelli che hanno subito fine-tuning, i quali, a differenza di approcci basati sul prompting, perdono la loro capacità di adattarsi dinamicamente a nuovi task o domini senza ulteriori cicli di addestramento. Questa rigidità rappresenta un limite importante, soprattutto in contesti dove le domande o i dati di input possono variare in modo significativo rispetto al set di addestramento. Inoltre, il fine-tuning comporta costi temporali e computazionali elevati: la necessità di eseguire molteplici iterazioni di addestramento su GPU di fascia alta, combinata con la ricerca di iperparametri ottimali, ha richiesto un impegno notevole in termini di risorse. Anche il monitoraggio continuo di metriche come la *loss* e l'*accuracy* o la sperimentazione di tecniche di regolarizzazione per evitare fenomeni di *overfitting* ha rappresentato una sfida, soprattutto considerando la relativa esiguità del dataset KvasirVQA rispetto ad altri più ampi come MSCOCO. Infine, nonostante il fine-tuning abbia permesso di ottenere modelli più specializzati e in grado di raggiungere prestazioni competitive, il trade-off tra flessibilità e specializzazione rimane un punto cruciale da considerare. Queste criticità evidenziano come, per sfruttare appieno il potenziale del fine-tuning, sia necessaria una pianificazione dettagliata che includa

dataset sufficientemente diversificati e strategie di ottimizzazione mirate, bilanciando al contempo costi e benefici nel contesto applicativo specifico.

Complessivamente, queste criticità offrono importanti spunti di riflessione per migliorare l'efficacia del prompting e ottimizzare i suoi costi, aprendo nuove strade per future ricerche in questo ambito.

5.5.8 Visualizzazione attivazioni GradCAM

Per comprendere meglio come una rete neurale convoluzionale come la *ResNet-50*, preaddestrata su ImageNet e riaddestrata su HyperKvasir e KvasirInstrument, elabora le immagini e quali regioni visive considera rilevanti durante l'estrazione delle caratteristiche, è stato utilizzato l'algoritmo *Grad-CAM*, per evidenziare le aree di maggiore attivazione che influenzano le decisioni del modello.

In questa analisi, la ResNet-50 è stata utilizzata come feature extractor globale, e le sue attivazioni sono state analizzate per due immagini rappresentative provenienti dal dataset KvasirVQA. Le immagini selezionate includono un'immagine con un polipo e una con uno strumento.

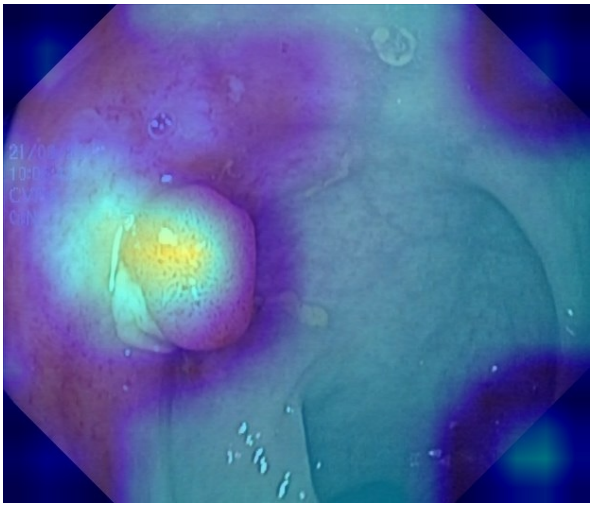


Figure 5.5: Visualizzazione delle attivazioni in immagine con polipo della ResNet-50 tramite Grad-CAM.

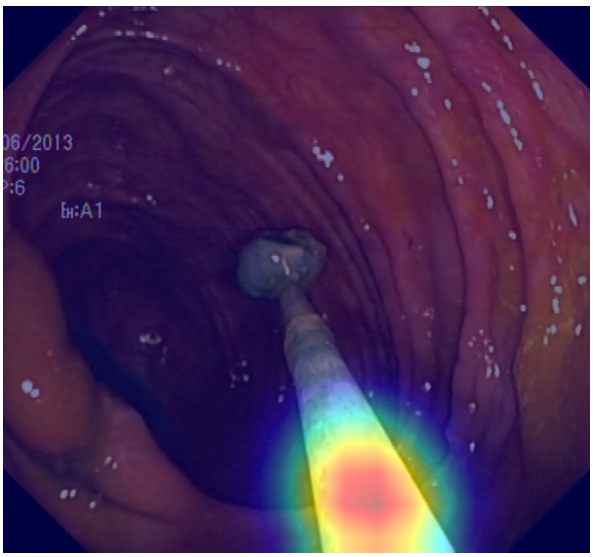


Figure 5.6: Visualizzazione delle attivazioni in immagine con strumento della ResNet-50 tramite Grad-CAM.

Nella Figura 5.5, l'heatmap generata da Grad-CAM evidenzia lievemente la regione contenente l'anomalia, indicando che il modello considera quest'area come più rilevante per l'estrazione delle caratteristiche.

Nella Figura 5.6, l'heatmap mostra una distribuzione più concentrata in una regione dell'immagine contenente sì lo strumento, ma che mantiene un certo grado di imprecisione.

L'utilizzo di Grad-CAM ha permesso di ottenere una maggiore trasparenza nel processo decisionale della ResNet-50, evidenziando le regioni visive più significative.

Capitolo 6

Conclusione

Il lavoro svolto in questa tesi si è concentrato sull'applicazione delle tecniche di VQA al dominio medico, con un focus specifico sul dataset *KvasirVQA*, una risorsa progettata per affrontare domande relative ad immagini gastroenterologiche. Lo studio ha esplorato approcci diversificati, tra cui il fine-tuning di modelli multimodali e l'adozione di tecniche di prompting, al fine di analizzare la capacità dei modelli di combinare informazioni visive e testuali in uno scenario complesso e specialistico.

La prima fase della tesi ha riguardato l'analisi del dataset KvasirVQA, evidenziando le sue caratteristiche principali, come la natura multimodale delle annotazioni e la loro rilevanza clinica. Tuttavia, l'analisi ha anche portato alla luce alcune criticità, come la distribuzione sbilanciata delle classi e la presenza di dati rumorosi, che hanno richiesto interventi mirati di *data augmentation* per ottimizzare le condizioni di addestramento dei modelli. Questi interventi si sono dimostrati cruciali per affrontare la complessità intrinseca del dataset e per garantire una base adeguata per la valutazione delle tecniche sperimentate.

Tra i modelli testati, *ViLT* ha mostrato prestazioni significativamente superiori rispetto all'architettura *Custom*, dimostrando l'importanza del pre-addestramento su dataset di grandi dimensioni e l'efficacia di un'architettura unificata per integrare immagini e testo. Tuttavia, l'approccio di fine-tuning ha evidenziato alcune limitazioni, tra cui la dipendenza dalla qualità e quantità dei dati e l'elevato costo computazionale associato all'addestramento su dataset complessi. Questi aspetti hanno sottolineato la necessità di ottimizzare sia i modelli che i dataset per migliorare l'accessibilità e la scalabilità delle soluzioni basate su VQA.

Parallelamente, sono state esplorate tecniche di prompting, come il *templating* e il *Chain-of-Thought*, che offrono un'alternativa al fine-tuning, permettendo di guidare i modelli pre-addestrati attraverso istruzioni esplicite. Queste tecniche si sono dimostrate vantaggiose in termini di flessibilità e riduzione dei costi computazionali, ma hanno presentato criticità legate alla sensibilità alla progettazione dei prompt. In particolare, sono state osservate risposte incoerenti o allucinazioni in presenza di prompt mal strutturati, evidenziando la necessità di una maggiore standardizzazione e controllo

nella loro implementazione.

Un altro aspetto significativo affrontato in questa tesi è stata l'introduzione del task di classificazione multilabel come alternativa al tradizionale approccio multiclasse. Questo ha permesso di rappresentare meglio la complessità delle risposte nel dataset KvasirVQA, riducendo le rigidità tipiche della classificazione multiclasse e migliorando le prestazioni dei modelli, in particolare nel caso del *Custom*. L'adozione di questo approccio ha inoltre evidenziato il potenziale per ulteriori innovazioni nell'adattamento dei task ai requisiti specifici del dataset.

Infine, sono state impiegate tecniche di interpretabilità, come *Grad-CAM*, per analizzare le attivazioni dei modelli durante l'inferenza del modello *Custom*. Queste analisi hanno fornito insight preziosi sul comportamento dei modelli e sulla loro capacità di focalizzarsi sugli elementi visivi rilevanti, contribuendo a una comprensione più approfondita delle loro prestazioni e limitazioni.

Sebbene il lavoro svolto abbia prodotto risultati promettenti, sono emerse diverse sfide che suggeriscono direzioni per la ricerca futura:

- **Espansione del dataset:** Ampliamento del KvasirVQA con un maggior numero di immagini e annotazioni più dettagliate, al fine di migliorare l'addestramento dei modelli e ridurre lo sbilanciamento delle classi.
- **Ottimizzazione dei modelli:** Sviluppo di modelli multimodali più leggeri e meno costosi in termini computazionali ma mantenendo prestazioni competitive.
- **Miglioramento del prompting:** Esplorazione di tecniche di prompting avanzate, come il *dynamic templating*, per adattare dinamicamente i prompt alle caratteristiche delle domande e delle immagini.
- **Maggiore interpretabilità:** Combinazione di tecniche come *Grad-CAM* con approcci innovativi per analizzare le rappresentazioni interne dei modelli, migliorando la trasparenza e la fiducia negli output generati.
- **Generalizzazione ad altri domini:** Applicazione delle metodologie sviluppate ad altri contesti clinici e scientifici, per verificare la generalizzabilità dei risultati ottenuti.

In conclusione, questa tesi rappresenta un contributo significativo nell'applicazione del VQA al dominio medico, dimostrando come tecniche avanzate di machine learning possano essere utilizzate per affrontare task complessi in ambiti specialistici. I risultati ottenuti evidenziano il potenziale delle tecnologie multimodali per migliorare la comprensione e l'analisi di immagini diagnostiche, fornendo al contempo una base per ulteriori sviluppi nel campo del VQA e delle sue applicazioni in ambito medico.

Capitolo 7

Glossario

CNN - Convolutional Neural Network Reti neurali progettate per analizzare dati visivi, utilizzando filtri convoluzionali per estrarre caratteristiche rilevanti.

CoT - Chain of Thought Tecnica di prompting che guida il modello a generare ragionamenti espliciti e passo-passo per risolvere problemi complessi.

CV - Computer Vision Campo dell'intelligenza artificiale che si occupa dell'elaborazione e comprensione automatica di immagini e video.

KB - Knowledge Base Archivio strutturato di conoscenze utilizzato per supportare processi di ragionamento automatico e rispondere a domande basate su fatti.

LLM - Large Language Model Modello linguistico di grandi dimensioni, progettato per elaborare e generare testo con capacità avanzate di comprensione e generazione del linguaggio naturale.

LSTM - Long Short-Term Memory Una variante delle reti neurali ricorrenti progettata per apprendere dipendenze a lungo termine nei dati sequenziali, evitando il problema del gradiente evanescente.

MSCOCO - Microsoft Common Objects in Context Dataset di immagini annotato per compiti come object detection, segmentation e captioning, noto per la sua varietà e complessità.

NLP - Natural Language Processing Campo dell'intelligenza artificiale che si occupa dell'elaborazione e comprensione automatica del linguaggio naturale.

OCR - Optical Character Recognition Tecnologia che consente di estrarre e digitalizzare automaticamente il testo contenuto in immagini o documenti scansionati.

RPN - Region Proposal Network Rete neurale utilizzata per generare proposte di regioni di interesse in un'immagine, comunemente impiegata in compiti di detection degli oggetti.

VLM - Vision-Language Model Modello multimodale progettato per elaborare e comprendere congiuntamente dati visivi e testuali.

VQA - Visual Question Answering Task multimodale che combina l'elaborazione di immagini e testo per rispondere a domande relative al contenuto visivo.

Bibliografia

[1] Gpt-4v(ision) system card. 2023.

[2] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly available clinical bert embeddings, 2019.

[3] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and VQA. *CoRR*, abs/1707.07998, 2017.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[6] Rabiul Awal, Le Zhang, and Aishwarya Agrawal. Investigating prompting techniques for zero- and few-shot visual question answering, 2024.

[7] Hanna Borgli, Vajira Thambawita, Pia H Smedsrud, Steven Hicks, Debesh Jha, Sigrun L Eskeland, Michael A Riegler, Pål Halvorsen, Arash A Dezfouli, Dag Johansen, et al. Hyperkvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy. *Scientific Data*, 7(1):283, 2020.

[8] 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, 2021.

[9] Sushant Gautam, Andrea Storås, Cise Midoglu, Steven A. Hicks, Vajira Thambawita, Pål Halvorsen, and Michael A. Riegler. Kvasir-vqa: A text-image pair

gi tract dataset. In *Proceedings of the First International Workshop on Vision-Language Models for Biomedical Applications (VLM4Bio '24)*, page 10 pages. ACM, 2024.

[10] Jacob Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[11] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. *CoRR*, abs/1612.00837, 2016.

[12] Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. Pathvqa: 30000+ questions for medical visual question answering, 2020.

[13] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[14] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021.

[15] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *CoRR*, abs/1602.07332, 2016.

[16] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. pages 228–231, 07 2007.

[17] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day, 2023.

[18] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[19] Weixin Liang, Yanhao Jiang, and Zixuan Liu. Graghvqa: Language-guided graph neural networks for graph-based visual question answering, 2021.

[20] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[22] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation, 2022.

[23] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[24] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering, 2022.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[27] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[29] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation, 2015.

[30] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language, 2022.

[31] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks, 2023.