



UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria Civile, Informatica e delle  
Tecnologie Aeronautiche  
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

**Image e Sketch Captioning**  
Sviluppo e Confronto di Modelli Encoder-Decoder per la  
Generazione di Didascalie per Immagini e Disegni

Laureando

**Tommaso Senatori**

Matricola 576833

Relatore

**Giuseppe Sansonetti**

Anno Accademico 2023/2024

# Introduzione

Negli ultimi decenni, l'intelligenza artificiale (**IA**) ha compiuto progressi straordinari, emergendo come un campo di ricerca centrale in numerosi settori. Le sue applicazioni pratiche spaziano dai sistemi di raccomandazione all'automazione industriale, dalla traduzione automatica agli assistenti virtuali, fino alla sicurezza informatica e oltre. Con algoritmi sempre più sofisticati e l'accesso a ingenti quantità di dati, l'IA ha rivoluzionato la nostra concezione di macchine intelligenti, dotandole di capacità analitiche e decisionali una volta considerate esclusivamente umane. Tra le varie sfide affrontate nell'ambito dell'IA, l'Image Captioning e lo Sketch Captioning rappresentano due problemi affascinanti e complessi.

L'**Image Captioning** consiste nell'abilità di generare una descrizione testuale per un'immagine.

**Sketch Captioning**, una variante più recente e complessa dell'Image Captioning, mira a generare descrizioni testuali a partire da schizzi stilizzati. Questi ultimi, contenendo meno dettagli visivi rispetto alle immagini, richiedono una maggiore astrazione da parte del modello per comprendere e descrivere accuratamente il contenuto.

Questi compiti si collocano all'intersezione tra la visione artificiale (Computer Vision) e l'elaborazione del linguaggio naturale (natural language processing, NLP).

La **Computer Vision** è un campo dell'intelligenza artificiale (IA) che permette ai computer e ai sistemi di ricavare informazioni significative da immagini digitali, video e altri input visivi - e intraprendere azioni o formulare delle segnalazioni sulla base di tali informazioni. Se l'IA permette ai computer di pensare, la computer vision permette loro di vedere, osservare e capire. [1]

Per NLP (Natural Language Processing) o elaborazione del linguaggio naturale si

intendono algoritmi di Intelligenza Artificiale in grado di analizzare, rappresentare e quindi comprendere il linguaggio naturale. Questo campo studia e sviluppa metodologie per le interazioni tra computer e esseri umani, occupandosi di dotare i computer della capacità di comprendere il linguaggio umano.

La combinazione di queste due discipline per l'Image Captioning è considerata una delle sfide più difficili nell'ambito del Machine Learning, poiché richiede che entrambe lavorino in coordinazione per trasformare input visivi in output testuali coerenti e significativi.

Il lavoro presentato in questa tesi si concentra sullo sviluppo di un modello Encoder-Decoder per affrontare sia il problema dell'Image Captioning che quello dello Sketch Captioning. L'obiettivo principale è di progettare un sistema capace di estrarre caratteristiche rilevanti dalle immagini e dai disegni stilizzati e di generare descrizioni testuali coerenti ed esplicative. Questo compito è complesso a causa della necessità di comprendere il contesto visivo e di tradurlo in un linguaggio naturale adeguato, mantenendo la coerenza semantica e sintattica.

Per risolvere il problema, è stato adottato un approccio basato su una rete neurale convoluzionale (CNN) per l'estrazione delle caratteristiche visive, combinata con una rete neurale ricorrente (RNN) per la generazione delle descrizioni testuali. Varie tecniche e diverse architetture di reti neurali sono state sperimentate e discusse al fine di ottimizzare le prestazioni del sistema. Questo modello Encoder-Decoder è stato quindi addestrato e valutato su diversi dataset, tra cui il noto “Flickr8k” per le immagini e “sketch-scene” per i disegni stilizzati.

Inoltre, è stato creato un dataset appositamente trasformando le immagini di Flickr8k in disegni stilizzati (sketch), permettendo di addestrare e valutare il modello su input visivi di diversa complessità e dettaglio. Questa varietà di dati ha permesso di testare il modello in diverse condizioni, garantendo una valutazione completa delle sue capacità e performance.

La Tesi è strutturata come segue:

Il Capitolo 1 svolge un’analisi approfondita dei fondamenti teorici, esaminando il modello Encoder-Decoder, l’architettura CNN e il ruolo delle reti neurali ricorrenti (RNN) nella generazione di descrizioni testuali.

Il Capitolo 2 presenta la metodologia di sviluppo, focalizzandosi sui diversi dataset utilizzati e le librerie impiegate per l’implementazione.

Il Capitolo 3 delinea l’approccio tecnico adottato, dall’estrazione delle caratteristiche alla creazione del modello Encoder-Decoder, con particolare attenzione alla fase di training e alle metriche di valutazione.

Infine, il Capitolo 4 fornisce i risultati e la valutazione sperimentale del sistema, presentando grafici di loss, metriche di valutazione e esempi di immagini generate.

# Indice

<b>Introduzione</b>	<b>ii</b>
<b>Indice</b>	<b>v</b>
<b>Elenco delle figure</b>	<b>viii</b>
<b>1 Fondamenti Teorici</b>	<b>1</b>
1.1 Modello Encoder-Decoder . . . . .	1
1.2 CNN . . . . .	3
1.2.1 Elementi CNN . . . . .	3
1.2.2 CNN e Transfer Learning nell'Image Recognition . . . . .	7
1.2.3 CNN Utilizzate . . . . .	8
1.3 RNN . . . . .	14
1.3.1 Vanishing/Exploding Gradient Problem . . . . .	16
1.3.2 LSTM . . . . .	18
<b>2 Metodologia di Sviluppo</b>	<b>21</b>
2.1 Datasets . . . . .	21
2.1.1 Flickr8k . . . . .	21
2.1.2 sketch-scene . . . . .	23
2.1.3 Flickr8k Sketchified . . . . .	24
2.2 Librerie . . . . .	26
2.2.1 Keras/Tensorflow . . . . .	26
2.2.2 Altre Librerie . . . . .	27

<b>3 Approccio Tecnico</b>	<b>29</b>
3.1 Feature Extraction . . . . .	29
3.2 Mapping Image_ID - Captions . . . . .	31
3.3 Text Preprocessing . . . . .	32
3.3.1 Cleaning . . . . .	32
3.3.2 Lemmatization . . . . .	32
3.3.3 Ulteriori Azioni di Preprocessing del Testo . . . . .	33
3.3.4 Tokenization . . . . .	35
3.3.5 max_length . . . . .	36
3.3.6 Istogramma della Lunghezza delle Didascalie . . . . .	37
3.4 Training-Validation-Test Split . . . . .	38
3.5 Creazione del Modello Encoder-Decoder . . . . .	39
3.6 Training e Data Generator . . . . .	42
3.6.1 Data Generator . . . . .	42
3.6.2 Training . . . . .	44
3.7 Metodi di Generazione delle Didascalie . . . . .	45
3.7.1 Greedy . . . . .	46
3.7.2 Beam Search . . . . .	48
3.8 Metriche . . . . .	49
3.8.1 Metrica BLEU . . . . .	50
3.8.2 Metrica CIDEr . . . . .	52
3.8.3 Metrica ROUGE-L . . . . .	53
<b>4 Risultati e Valutazione Sperimentale</b>	<b>55</b>
4.1 Grafici di Loss . . . . .	57
4.2 Metriche . . . . .	60
4.2.1 BLEU-1 e BLEU-2 . . . . .	61
4.2.2 CIDEr . . . . .	63
4.2.3 ROUGE-L . . . . .	63
4.3 Esempi di Immagini . . . . .	64
4.3.1 Immagini del dataset sketch-scene . . . . .	65
4.3.2 Immagini dei dataset Flickr8k e Flickr8k Sketchified . . . . .	69

4.3.3    Confronto tra i Dataset . . . . .	74
<b>Conclusioni e sviluppi futuri</b>	<b>75</b>
Conclusioni . . . . .	75
Sviluppi futuri . . . . .	76
<b>Bibliografia</b>	<b>77</b>

# Elenco delle figure

1.1	Architettura di un modello Encoder-Decoder per Image Captioning. . . . .	2
1.2	Esempi di filtri basilari in una Rete Neurale Convoluzionale. . . . .	4
1.3	Dimostrazione pratica del funzionamento del Max Pooling. . . . .	5
1.4	Applicazione di Max Pooling con pool size 2x2 e stride 2 su un'immagine del dataset MNIST, riducendo la dimensione dell'immagine e preservandone le caratteristiche rilevanti. . . . .	6
1.5	Prestazioni di diversi modelli di Deep Learning nella competizione ILSVRC in termini di accuratezza. . . . .	8
1.6	Architettura della rete neurale VGG16. . . . .	9
1.7	Altra rappresentazione dell'architettura della rete VGG16 con dimensioni specifiche per ogni strato. . . . .	9
1.8	Struttura di un modulo Inception "naive" (ingenuo). . . . .	12
1.9	Struttura di un modulo Inception che utilizza convoluzioni 1x1 per la riduzione della dimensionalità. . . . .	12
1.10	Architettura della rete neurale Xception. . . . .	13
1.11	Confronto delle validation accuracy tra le architetture Xception e InceptionV3 usando il dataset ImageNet. . . . .	14
1.12	Schema della cellula unitaria di una rete neurale ricorrente (RNN) con 2 input e con anello di feedback. . . . .	15
1.13	Schema della cellula unitaria di una rete neurale ricorrente (RNN) con 2 input e con anello di feedback "srotolato". . . . .	15
1.14	Rappresentazione semplificata dell'Exploding Gradient Problem, con n=2. .	17

1.15 Architettura di una unit cell LSTM. I valori dei pesi, bias, input e output illustrati sono esempi per agevolare la comprensione. . . . .	18
2.1 Alcune delle immagini presenti nel dataset Flickr8k. . . . .	22
2.2 Alcune delle descrizioni presenti nel dataset Flickr8k. . . . .	22
2.3 Alcune delle immagini presenti nel dataset sketch-scene. . . . .	23
2.4 Alcune delle descrizioni presenti nel dataset sketch-scene. . . . .	24
2.5 Illustrazione dell'intero processo di conversione di un'immagine in sketch. .	25
2.6 Alcune delle immagini presenti nel dataset Flickr8k Sketchified. . . . .	26
3.1 Feature Vector estratto da un'immagine di un cane. . . . .	31
3.2 Esempio di distribuzione 2D dei Feature Vector per 2 classi: gatti e cani. .	31
3.3 Captions di un'immagine di Flickr8k prima e dopo il preprocessing. . . . .	34
3.4 Captions di un'immagine di sketch-scene prima e dopo il preprocessing. .	34
3.5 Confronto tra i vocabolari dei dataset. . . . .	36
3.6 Istogramma della lunghezza delle captions, per il dataset Flickr8k. . . . .	37
3.7 Istogramma della lunghezza delle captions, per il dataset sketch-scene. .	38
3.8 Modello Encoder-Decoder per Image Captioning, usando una rete Xception come CNN e una LSTM come RNN. . . . .	41
3.9 Modello Encoder-Decoder per Image Captioning, usando una rete VGG16 come CNN e una SimpleRNN come RNN. . . . .	42
3.10 Meccanismo di generazione di una caption, usando la tecnica Greedy (rosso). .	47
3.11 Meccanismo di generazione di una caption, usando la tecnica Beam Search (rosso). . . . .	48
3.12 <b>Esempio 1</b> BLEU: Precisione Unigramma. . . . .	51
3.13 <b>Esempio 2</b> BLEU: Confronto tra precisione Unigramma e Bigramma. . . . .	51
4.1 Confronto delle funzioni di loss tra i due programmi basati sul dataset SKETCH-SCENE. . . . .	58
4.2 Confronto delle funzioni di loss tra i due programmi basati sul datset FLICKR8K: "FLICKR8K standard" (a sinistra) e "FLICKR8K advanced" (a destra). . . . .	59

4.3	Confronto delle funzioni di loss tra il programma basato sul dataset FLICKR8K SKETCHIFIED (a sinistra) e il programma basato su FLICKR8K "FLICKR8K advanced" (a destra). . . . .	59
4.4	. . . . .	65
4.5	. . . . .	66
4.6	. . . . .	67
4.7	. . . . .	68
4.8	. . . . .	70
4.9	. . . . .	71
4.10	. . . . .	71
4.11	. . . . .	72
4.12	. . . . .	73
4.13	. . . . .	74

# Capitolo 1

## Fondamenti Teorici

Questo capitolo esplora la struttura, gli elementi costitutivi e il funzionamento generale del modello encoder-decoder, delle reti neurali ricorrenti (RNN) e delle reti neurali convoluzionali (CNN), includendo un'analisi dettagliata delle specifiche architetture di CNN e RNN utilizzate nel progetto.

### 1.1 Modello Encoder-Decoder

Un modello Encoder-Decoder è un'architettura neurale impiegata per trasformare sequenze di input di lunghezza variabile in sequenze di output di lunghezza variabile. È comunemente utilizzata per affrontare problemi di generazione di sequenze, come la traduzione automatica, e nel contesto di questo lavoro, per generare descrizioni di immagini. Questa architettura è composta da due componenti principali: l'**Encoder** e il **Decoder**.

Nel contesto dell'Image Captioning, il modello Encoder-Decoder sfrutta una combinazione di reti neurali convoluzionali (CNN) e reti neurali ricorrenti (RNN) per generare didascalie descrittive per le immagini. Questo approccio sfrutta le capacità delle CNN nell'estrazione delle caratteristiche salienti dall'immagine e delle RNN nella generazione del testo.

L'encoder è costituito da una CNN pre-addestrata che converte l'immagine in una rappresentazione vettoriale compatta che cattura le informazioni visuali rilevanti. Questa rappresentazione vettoriale, spesso denominata vettore di caratteristiche o **feature vector**, contiene informazioni astratte sull'aspetto dell'immagine.

Successivamente, il vettore di caratteristiche ottenuto dall'encoder viene fornito al decoder, implementato tipicamente come una rete neurale ricorrente. Il decoder riceve in input anche un token di inizio sequenza e genera una parola come output. Questa parola predetta viene quindi aggiunta all'input che viene usato per predire la parola successiva.

Ad ogni passo temporale, la RNN riceve in input il vettore di caratteristiche e la didascalia generata fino alla parola corrente, e genera la parola successiva nella sequenza.

Viene quindi predetta l'intera sequenza una parola per volta fino alla predizione del token di fine.

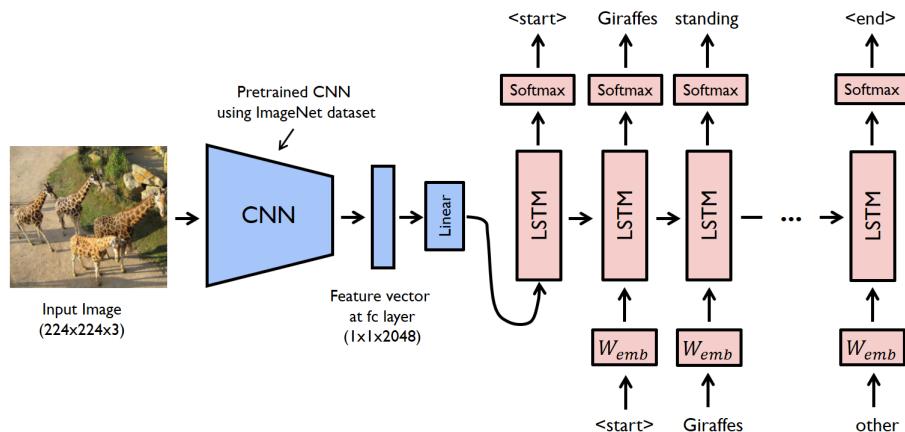


Figura 1.1: Architettura di un modello Encoder-Decoder per Image Captioning.

L'obiettivo del modello Encoder-Decoder per Image Captioning è quello di imparare a generare captions significative e coerenti che descrivano accuratamente il contenuto dell'immagine di input. Questo viene fatto minimizzando una funzione di loss che valuta la discrepanza tra le didascalie generate e le didascalie di riferimento nel set di dati di addestramento.

In altre parole, maggiore è la somiglianza tra la caption generata dal modello e quella reale, minore sarà la loss, e viceversa.

In sintesi, il modello Encoder-Decoder nel presente contesto sfrutta le CNN per l'estrazione delle caratteristiche visive e le RNN per la generazione del testo, consentendo di creare descrizioni testuali delle immagini.

Nelle sezioni successive di questo capitolo sono analizzate l'architettura e il funzionamento delle reti neurali convoluzionali (CNN) e ricorrenti (RNN), ponendo particolare attenzione alle implementazioni utilizzate nel contesto del progetto.

## 1.2 CNN

Le **Convolutional Neural Networks (CNN)** [2] rappresentano una tipologia di reti neurali ampiamente impiegata nel campo del Deep Learning per l'analisi delle immagini. Queste reti sono progettate per catturare e apprendere automaticamente le caratteristiche spaziali delle immagini attraverso l'utilizzo di convoluzioni.

### 1.2.1 Elementi CNN

Le CNN sono composte principalmente da tre tipi di strati:

1. Strati convoluzionali (convolutional layers)
2. Strati di pooling (pooling layers)
3. Strati completamente connessi (fully connected layers)

#### 1.2.1.1 Strati Convoluzionali

Gli strati convoluzionali (**Convolutional Layers**) costituiscono il fondamento delle CNN. Essi eseguono operazioni di convoluzione sui dati in ingresso, applicando filtri su regioni sovrapposte dei dati di input. Questi filtri, o **kernel**, presenti negli strati convoluzionali delle CNN, agiscono come detector di pattern, identificando caratteristiche specifiche nelle immagini.

Nella fase iniziale della rete, i filtri si concentrano principalmente sull'individuare pattern geometrici di base, come bordi e angoli, mentre negli strati più profondi della

rete, si specializzano sempre di più nel rilevare pattern più complessi, come oggetti, animali, volti o parti del corpo.

Un filtro è essenzialmente una matrice relativamente piccola, i cui valori sono inizializzati con numeri casuali. Durante il processo di backpropagation, i valori nella matrice del filtro vengono regolarmente aggiornati per ottimizzare le prestazioni della rete neurale.

Durante l'applicazione della convoluzione, tale matrice scorre sull'immagine di input, eseguendo un prodotto scalare con ciascuna regione sovrapposta dell'immagine. Il risultato di questa operazione di convoluzione genera una nuova rappresentazione dell'input, nota come **feature map**, in cui i valori riflettono l'attivazione del filtro nelle varie regioni dell'immagine.

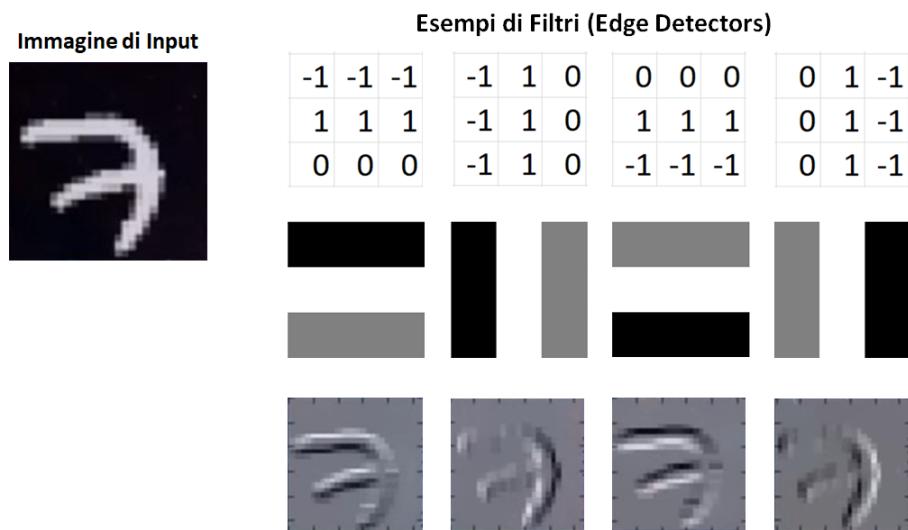


Figura 1.2: Esempi di filtri basilari in una Rete Neurale Convoluzionale.

Nell'esempio in figura 1.2, si presentano quattro tipologie di filtri basilari. Questi filtri sono rappresentati visivamente, dove il valore 1 indica il colore bianco, -1 il colore nero e 0 il colore grigio. L'immagine di input utilizzata è estratta dal dataset MNIST.

Effettuando la convoluzione tra l'immagine di input e tali filtri, si ottengono differenti output. I pixel più chiari indicano i pattern che il filtro ha riconosciuto.

Ad esempio, il primo output evidenzia la rilevazione di bordi orizzontali superiori da parte del primo filtro, mentre il secondo rileva bordi verticali sinistri, e così via.

Questi filtri, poiché identificano pattern basilari, sono comunemente posizionati nei primi strati di una CNN.

### 1.2.1.2 Strati di Pooling

I **pooling layers** nelle reti neurali convoluzionali (CNN) sono strati utilizzati per ridurre la dimensionalità dell'immagine di input, raggruppando i pixel in regioni e selezionando il valore più rappresentativo per ciascuna regione. Questa tecnica aiuta a ridurre la complessità della rete e a estrarre le caratteristiche più rilevanti dell'immagine.

Il **max pooling** è il metodo più diffuso tra le tecniche di pooling.

Questa tecnica, applicata dopo i livelli convoluzionali, raggruppa i pixel in regioni (pool) e per ciascuna regione seleziona il valore massimo, riducendo così l'immagine con le caratteristiche rilevanti conservate.

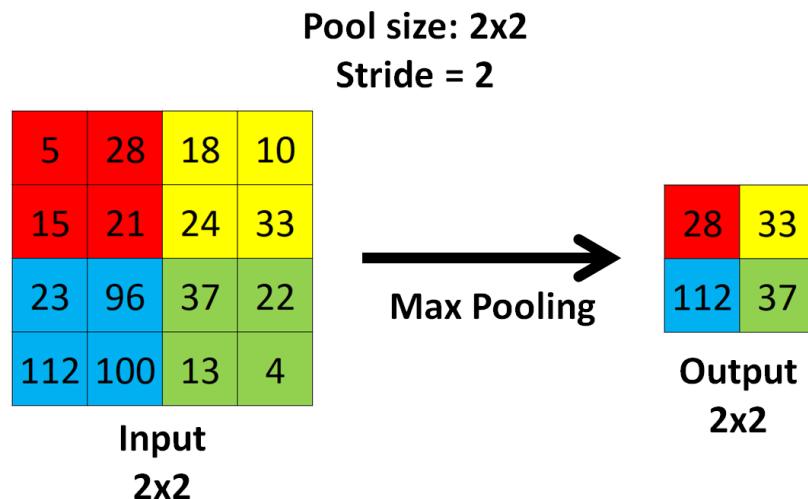


Figura 1.3: Dimostrazione pratica del funzionamento del Max Pooling.

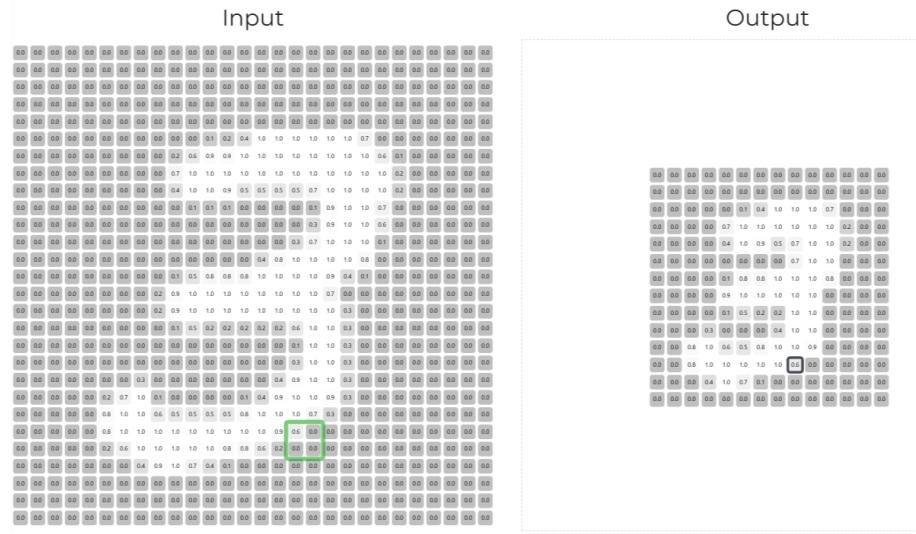


Figura 1.4: Applicazione di Max Pooling con pool size 2x2 e stride 2 su un’immagine del dataset MNIST, riducendo la dimensione dell’immagine e preservandone le caratteristiche rilevanti.

Il processo di max pooling coinvolge la definizione di una dimensione della regione (**pool\_size**) e uno **stride**, che indica di quanti pixel la regione si sposta lungo l’immagine durante il pooling. È inoltre possibile scegliere se preservare o meno la dimensione dell’input. In caso affermativo, è necessario utilizzare lo **zero padding**. Questo significa che vengono aggiunti zeri intorno ai bordi dell’immagine di input prima dell’applicazione del max pooling, garantendo che la dimensione dell’output sia uguale a quella dell’input.

Durante l’esecuzione del max pooling, la regione si sposta sull’immagine e per ciascuna regione viene selezionato il valore massimo, che viene quindi conservato nell’output.

Il max pooling nelle CNN offre vantaggi significativi: riduce la dimensionalità dell’immagine di input, contribuendo a una maggiore efficienza computazionale e riducendo il numero di parametri nella rete. Questo porta a una riduzione del carico computazionale durante l’addestramento e l’utilizzo del modello, migliorando allo stesso tempo la sua capacità di generalizzazione e prevenendo l’overfitting.

È importante notare che, oltre al max pooling, esistono altre tecniche di pooling, come l’average pooling, che calcola la media dei valori nella regione anziché il massimo.

### 1.2.1.3 Strati Completamente Connessi

Gli strati **fully connected** o **dense** delle CNN, che sono analoghi agli strati presenti in tutte le reti neurali, svolgono un ruolo fondamentale nell'analisi delle feature estratte.

Dopo l'elaborazione attraverso gli strati di convoluzione e di pooling, le mappe delle caratteristiche di output dell'ultimo strato vengono comunemente appiattite, trasformate in un array monodimensionale di numeri o vettori. Questo processo, noto come **flattening**, consente di connettere le caratteristiche estratte agli strati densi, in cui ogni neurone è collegato a ogni output mediante pesi apprendibili, contribuendo così alla generazione degli output finali della rete. Ad esempio, nei compiti di classificazione, l'ultimo strato completamente connesso avrà tipicamente lo stesso numero di nodi di output del numero di classi, producendo le probabilità per ciascuna classe.

## 1.2.2 CNN e Transfer Learning nell'Image Recognition

Le reti neurali convoluzionali (CNN) hanno dimostrato eccellenti capacità nell'elaborazione delle immagini, ma richiedono enormi quantità di dati per l'addestramento. In questa sezione sono esaminati il concetto di Transfer Learning e il ruolo fondamentale del dataset ImageNet e della competizione ILSVRC nel contesto delle CNN.

### 1.2.2.1 ImageNet Dataset

Il dataset **ImageNet** è una vasta raccolta di oltre 14 milioni di immagini etichettate e divise in più di 20.000 categorie che includono una varietà di soggetti come persone, animali, oggetti domestici, veicoli e altro ancora.

### 1.2.2.2 Competizione ILSVRC

La competizione ILSVRC (**ImageNet Large Scale Visual Recognition Challenge**) è un evento annuale all'interno del contesto di ImageNet, che utilizza un sottogruppo del dataset con oltre 1,2 milioni di immagini etichettate e circa 1.000 categorie. Gli obiettivi principali della competizione includono il riconoscimento di oggetti, la localizzazione degli oggetti e altre sfide relative alla computer vision. Ogni anno, vengono

presentate nuove architetture di reti neurali con risultati variabili, contribuendo così all'avanzamento delle CNN e dell'apprendimento automatico.

Nella tabella seguente sono riportati i risultati di alcune reti neurali partecipanti alla competizione.

	<b>Top-1 accuracy</b>	<b>Top-5 accuracy</b>
<b>VGG-16</b>	0.715	0.901
<b>ResNet-152</b>	0.770	0.933
<b>Inception V3</b>	0.782	0.941
<b>Xception</b>	<b>0.790</b>	<b>0.945</b>

Figura 1.5: Prestazioni di diversi modelli di Deep Learning nella competizione ILSVRC in termini di accuratezza.

### 1.2.2.3 Transfer Learning

Invece che addestrare una rete neurale da zero, ovvero partendo con dei pesi randomici che vengono progressivamente ottimizzati durante l'addestramento, oggi è possibile utilizzare una CNN pre-addestrata. Si può prendere una CNN come una di quelle riportate nella tabella 1.5, che ha acquisito una comprensione degli oggetti del mondo, e si trasferisce tutta la sua conoscenza per svolgere compiti simili, come nell'ambito dell'Image Captioning. Questo concetto è noto come Transfer Learning. Questa tecnica consente di ridurre significativamente il tempo e le risorse necessarie per addestrare nuovi modelli, ottenendo un'elevata accuratezza nei compiti di computer vision.

## 1.2.3 CNN Utilizzate

Per questo lavoro, sono state selezionate due reti neurali convoluzionali pre-inizializzate per ImageNet: VGG16 e Xception, le cui architetture sono descritte di seguito.

### 1.2.3.1 VGG16

L'architettura della VGG-16, sviluppata dal Visual Geometry Group (VGG) presso l'Università di Oxford, si distingue per la sua complessità stratificata, comprendente 16 strati in totale, tra cui 13 strati convoluzionali e 3 strati completamente connessi.

Questa rete neurale convoluzionale (CNN) è rinomata per la sua semplicità e capacità di eccellere in una vasta gamma di compiti di Computer Vision. Nonostante la sua struttura più lineare rispetto ad altre architetture più recenti, la VGG-16 continua a rimanere una scelta ampiamente apprezzata nel campo del Deep Learning, grazie alla sua versatilità e alle prestazioni di alto livello.

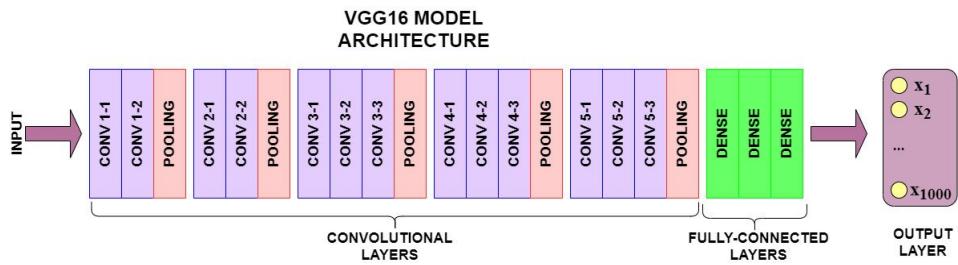


Figura 1.6: Architettura della rete neurale VGG16.

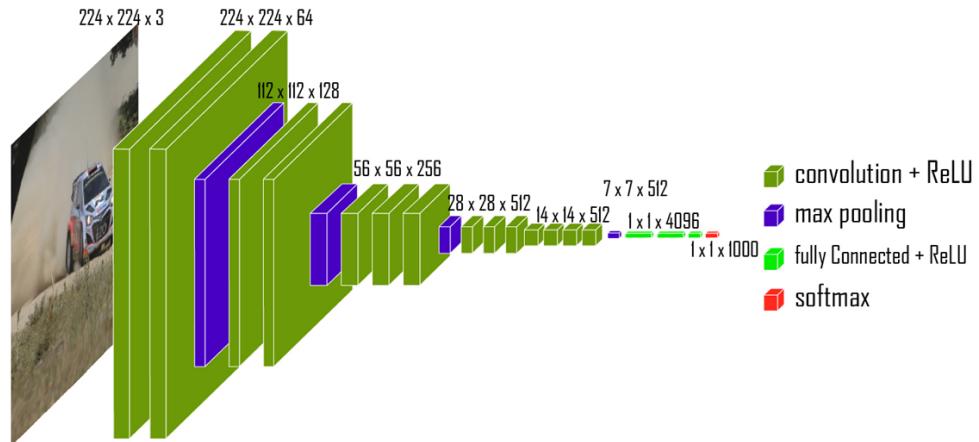


Figura 1.7: Altra rappresentazione dell'architettura della rete VGG16 con dimensioni specifiche per ogni strato.

L'architettura della VGG16 è definita come segue.

Il primo stack si occupa dell'elaborazione dell'immagine iniziale di dimensioni 224x224x3, che rappresenta un'immagine a colori con tre canali RGB. In questa fase, vengono applicati 64 kernel convoluzionali con dimensioni di 3x3, utilizzando zero padding. Successivamente, l'output ottenuto, con le stesse dimensioni di input, viene elaborato ulteriormente da un altro strato convoluzionale identico al precedente, seguito infine da

uno strato di max pooling che produce un volume finale di  $112 \times 112 \times 64$  per il primo strato della VGG16.

Nei successivi 4 blocchi, il procedimento segue lo stesso schema del primo: i primi due blocchi presentano 2 strati convoluzionali ciascuno, mentre gli ultimi tre ne includono 3. In ogni blocco è infine presente uno strato di max pooling.

In generale, si osserva un aumento progressivo del numero di filtri da 64 a 128, 256, e infine 512.

Allo stesso tempo, le dimensioni dell'output vengono ridotte progressivamente tramite max pooling, dimezzandosi di blocco in blocco da 224x224 fino a 7x7. Tutti i layer di max pooling utilizzano un pool\_size di 2x2 e uno stride di 2, garantendo questa riduzione delle dimensioni.

Dopo i primi 5 blocchi, l'uscita della rete ha dimensioni 7x7x512. A questo punto, i blocchi successivi consistono interamente in strati densi. Inizialmente, l'uscita viene appiattita tramite un layer di flatten, ottenendo così un vettore di lunghezza 25088.

Successivamente, si passa attraverso una serie di strati densi e dropout. Il vettore di 25088 neuroni viene ridotto a 4096 neuroni e infine a 1000 neuroni nell'output finale.

All'interno della rete VGG16, la funzione di attivazione utilizzata è sempre la ReLU (Rectified Linear Unit), definita come  $f(x) = \max(x, 0)$ . Tuttavia, nell'ultimo strato della rete, l'attivazione utilizzata è la Softmax , per generare le probabilità per le 1000 classi di output.

Complessivamente, questa rete conta 134,260,544 parametri, che corrispondono a 512.16 megabyte di spazio di memoria.

### 1.2.3.2 Xception

Xception [3] è un'architettura di rete neurale convoluzionale profonda che incorpora un tipo di operazione di convoluzione noto come Convoluzione Separabile in Profondità (Depthwise Separable Convolutions). Questo modello è stato sviluppato da Francois Chollet, noto anche come creatore di Keras, una libreria ampiamente utilizzata in que-

sto progetto.

Le **Depthwise Separable Convolutions** operano su filtri (kernel) che non possono essere suddivisi in due parti più piccole. Invece di dividere il filtro, questa tecnica separa il processo di convoluzione in due parti: una convoluzione Depthwise e una Pointwise. Questo approccio è più comune perché può essere applicato a una più ampia varietà di filtri. La convoluzione Depthwise si concentra sull'applicazione di filtri su ciascun canale di colore dell'immagine separatamente, mantenendo la profondità dell'immagine invariata. La convoluzione Pointwise, invece, utilizza un filtro di dimensioni 1x1 per combinare le informazioni spaziali e di profondità in ciascun punto dell'immagine, modificando il numero di canali.

Xception è anche conosciuta come una versione "estrema" di un modulo Inception. Pertanto, è utile esaminare il modulo Inception prima di approfondire Xception.

L'**Inception Block** [4] rappresenta una pietra miliare nell'evoluzione delle architetture di reti neurali convoluzionali, sfruttando il potere della parallelizzazione per estrarre e combinare informazioni multi-scala in modo efficiente.

La sua intuizione si fonda sulla consapevolezza che l'aggiunta di più strati convoluzionali in profondità può causare problemi di overfitting e richiedere risorse computazionali sempre maggiori per l'addestramento della rete. Invece, espandendo la larghezza della rete e utilizzando diverse pipeline di convoluzione con filtri di dimensioni differenti, l'Inception Block è in grado di catturare una vasta gamma di caratteristiche salienti in modo più efficiente.

All'interno dell'Inception Block, l'input viene simultaneamente elaborato da più flussi di convoluzione, ciascuno caratterizzato da filtri 1x1, 3x3 e 5x5, affiancati da un layer di max pooling 3x3.

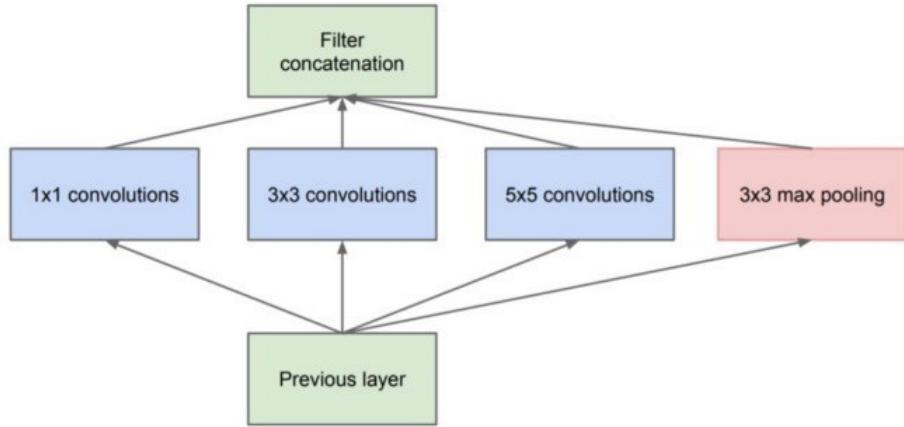


Figura 1.8: Struttura di un modulo Inception "naïve" (ingenuo).

L'uso di convoluzioni 1x1 rappresenta un'innovazione cruciale, in quanto permette di ridurre drasticamente il numero di parametri necessari per le convoluzioni successive, ottimizzando così l'efficienza computazionale del modello.

Ad esempio, utilizzando 30 filtri 1x1, un input di dimensioni 128x128x100 (con 100 feature map) può essere compresso fino a 128x128x30.

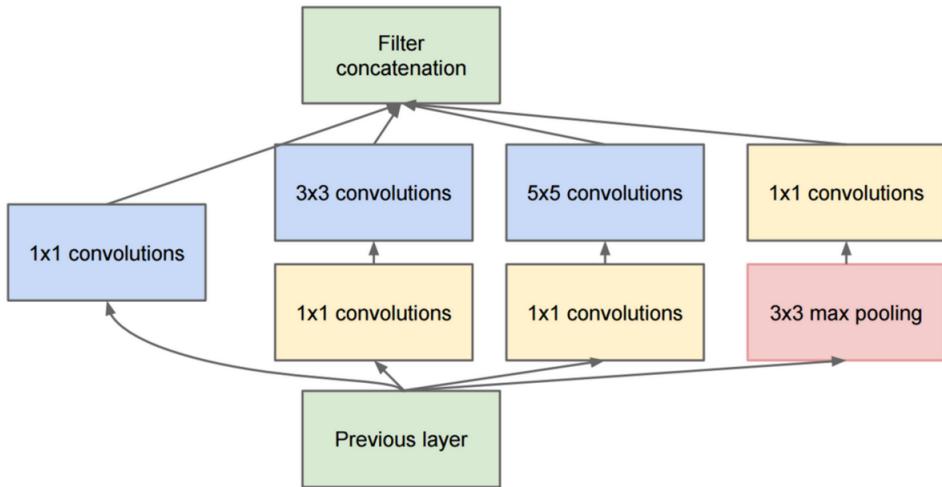


Figura 1.9: Struttura di un modulo Inception che utilizza convoluzioni 1x1 per la riduzione della dimensionalità.

**Xception**, acronimo di "Extreme Inception", spinge al massimo i principi di Inception. Nell'architettura Inception, le convoluzioni 1x1 vengono impiegate per ridurre

la dimensionalità dell'input originale, permettendo successivamente l'applicazione di diversi tipi di filtri su ciascuna dimensione di profondità dei dati risultanti.

Xception inverte questa sequenza operativa. Nell'ambito di Xception, le operazioni di convoluzione vengono dapprima eseguite su ciascuna mappa di profondità, e solo in seguito l'input viene compresso mediante una convoluzione 1x1 applicata in tutta la sua profondità.

Questo approccio innovativo consente a Xception di catturare caratteristiche più ricche e complesse applicando prima i filtri su ciascuna mappa di profondità, prima di eseguire la compressione dello spazio di input tramite la convoluzione 1x1.

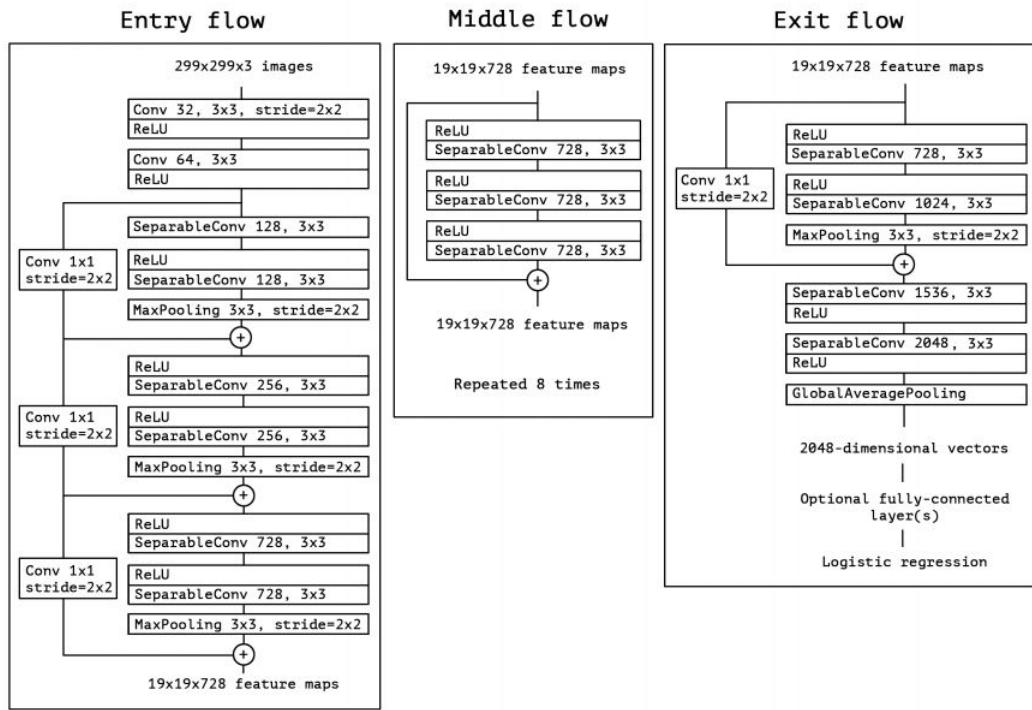


Figura 1.10: Architettura della rete neurale Xception.

I dati seguono prima il percorso del flusso di ingresso, procedono poi attraverso il flusso centrale, che si ripete 8 volte, e infine attraversano il flusso di uscita.

I risultati del modello Xception, come evidenziato in figura 1.5, precedentemente mostrata durante la discussione sulla competizione ILSVRC, dimostrano un'eccellente performance nell'ambito del dataset ImageNet.

In particolare, il grafico che segue mostra chiaramente che Xception supera significativamente InceptionV3 anche in termini di validation accuracy.

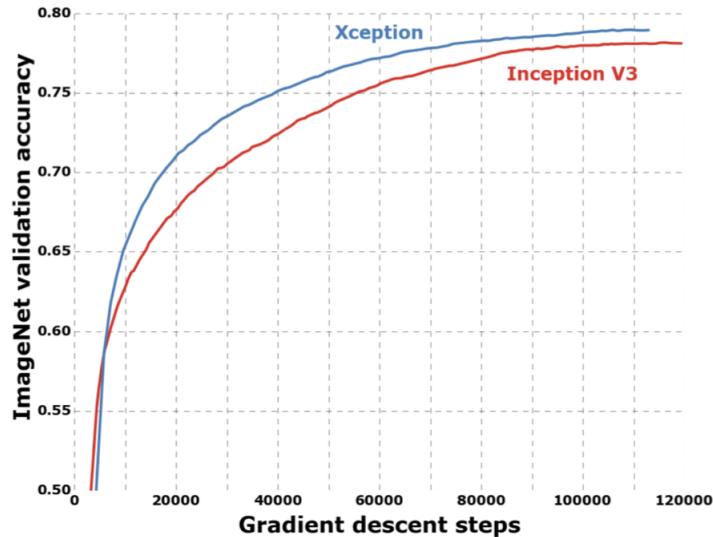


Figura 1.11: Confronto delle validation accuracy tra le architetture Xception e InceptionV3 usando il dataset ImageNet.

### 1.3 RNN

Una **rete neurale ricorrente (RNN)** [5] è un tipo di architettura progettata per gestire dati sequenziali, in cui l'ordine dei dati è significativo, come serie temporali, testi o sequenze di dati.

Rispetto ad altri modelli neurali, le RNN presentano la capacità distintiva di mantenere informazioni sui dati precedenti e di utilizzare queste informazioni per influenzare la previsione di eventi futuri. Questa capacità è resa possibile grazie all'aggiunta di cicli di feedback nella struttura della rete, consentendo alle informazioni di essere propagate attraverso le fasi successive della computazione.

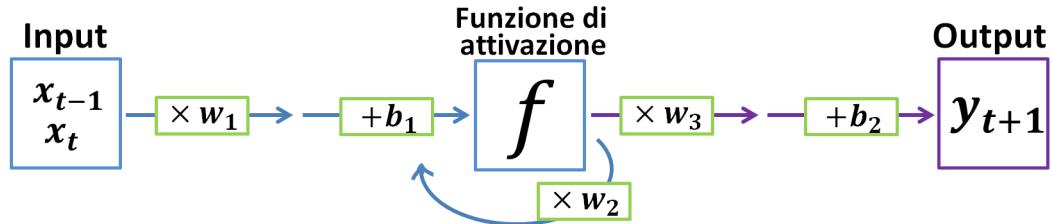
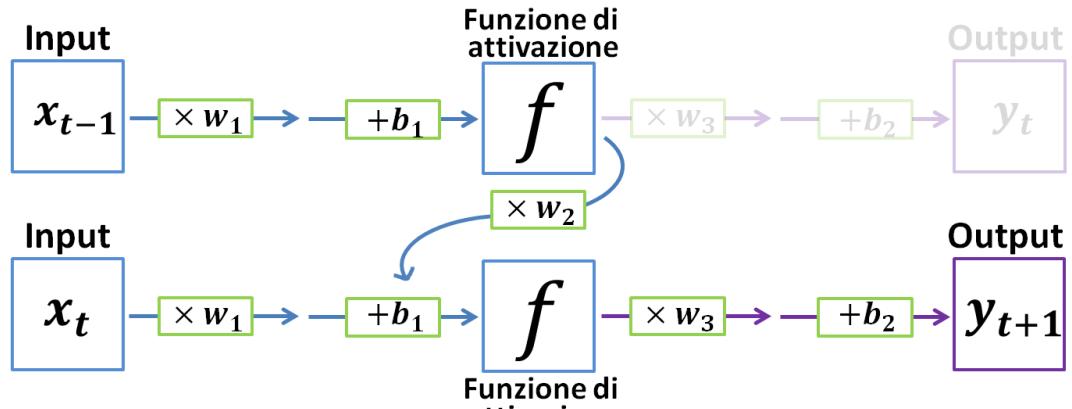


Figura 1.12: Schema della cellula unitaria di una rete neurale ricorrente (RNN) con 2 input e con anello di feedback.

Ogni unità ricorrente riceve in input un dato dalla sequenza in analisi e, insieme alla memoria interna, genera un'uscita che viene riusata come input per la successiva unità ricorrente nella sequenza.

Questa struttura ricorsiva delle unità ricorrenti consente alla RNN di mantenere una memoria a lungo termine degli eventi passati, permettendo di catturare pattern temporali complessi e di utilizzare queste informazioni per formulare previsioni o generare output nel contesto temporale attuale.

Per visualizzare questo processo in modo più intuitivo, è possibile "srotolare" la rete neurale ricorrente, creando una copia della rete per ciascun input sequenziale.



$$y_{t+1} = f(x_t w_1 + b_1 + f(x_{t-1} w_1 + b_1) w_2) w_3 + b_2$$

Figura 1.13: Schema della cellula unitaria di una rete neurale ricorrente (RNN) con 2 input e con anello di feedback "srotolato".

In questo modo, si ottengono una serie di reti neurali separate, ciascuna dotata dei propri input e output. Tuttavia, i pesi e i bias sono condivisi tra tutte le copie della

rete. Questo significa che, indipendentemente da quanto la rete viene "srotolata" per includere più input, il numero di parametri da addestrare rimane costante.

Inoltre, un aspetto determinante delle RNN è la loro flessibilità nel gestire sequenze di lunghezza variabile. Poiché le unità ricorrenti possono essere collegate in una catena senza fine, una RNN può, teoricamente, elaborare sequenze di qualsiasi lunghezza.

Tuttavia, nonostante le loro capacità, le RNN non sono sempre la scelta ideale a causa di problemi come il **Vanishing/Exploding Gradient Problem** durante il processo di addestramento.

### 1.3.1 Vanishing/Exploding Gradient Problem

Dopo aver esaminato i vantaggi delle RNN, è ora opportuno analizzare la ragione per cui non vengono utilizzate frequentemente.

Un grande ostacolo è rappresentato dal fatto che più la rete neurale viene "srotolata", più diventa difficile il processo di addestramento. Questo fenomeno è noto come Vanishing/Exploding Gradient Problem (problema del gradiente che svanisce o esplode).

Nella fase di ottimizzazione della rete neurale attraverso la **backpropagation**, vengono calcolate le derivate, o gradienti, per ogni parametro. Questi gradienti vengono poi utilizzati nell'**algoritmo di discesa del gradiente** per trovare i valori dei parametri che minimizzano una funzione di loss.

Per rendere più comprensibile il concetto di Vanishing/Exploding Gradient Problem, si semplifica considerando solo il peso  $w$  del feedback loop, tralasciando gli altri pesi e bias presenti nella rete. Sotto questa ipotesi, possiamo considerare che nell'RNN l'input venga amplificato un numero di volte pari a  $w^n$  prima di raggiungere l'output finale della RNN, dove  $n$  rappresenta il numero di input.

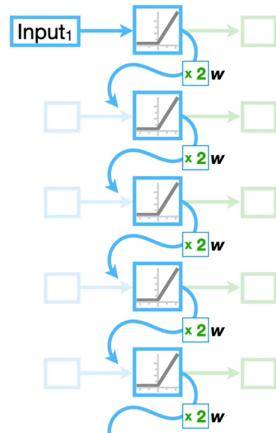


Figura 1.14: Rappresentazione semplificata dell’Exploding Gradient Problem, con  $n=2$ .

Si espone inizialmente l’**Exploding Gradient**. Questo problema si verifica quando il peso  $w$  del ciclo di feedback è impostato su un valore maggiore di 1. Ad esempio, se  $w = 2$  e ci sono 50 input, si ottiene  $2^{50}$ , un numero estremamente grande, che influenzerà uno dei gradienti. Quando il gradiente contiene un numero enorme, i passi compiuti durante la discesa del gradiente saranno relativamente grandi. Invece di convergere verso i parametri ottimali, la rete potrebbe oscillare intorno a questi valori, rendendo difficile il processo di ottimizzazione.

Una soluzione per prevenire l’Exploding Gradient Problem consiste nel vincolare il valore del peso  $w$  a un valore inferiore a 1. Tuttavia, ciò porta al **Vanishing Gradient Problem**. Questo si verifica quando il peso  $w$  del ciclo di feedback è impostato su un valore inferiore a 1. Ad esempio, se  $w = 0.5$  e ci sono 50 input, si ricava  $0.5^{50}$ , un valore molto vicino a zero.

Il problema associato a un gradiente troppo piccolo è che i pesi vengono aggiornati proporzionalmente al loro gradiente. Questi aggiornamenti risulterebbero a loro volta minimi, determinando una variazione insignificante nei valori dei pesi.

Durante la discesa del gradiente, questo comporterebbe passi molto piccoli. Di conseguenza, si potrebbe impiegare un numero elevato di iterazioni prima di raggiungere il punto ottimale, o addirittura rimanere bloccati in un minimo locale sub-ottimale.

### 1.3.2 LSTM

Una Long Short-Term Memory è un tipo di rete neurale ricorrente progettata per mitigare il problema del Vanishing/Exploding Gradient.

A differenza delle reti neurali ricorrenti tradizionali, che utilizzano lo stesso feedback loop per tutti gli eventi passati, la LSTM adotta due percorsi separati per fare predizioni sul futuro: uno per le memorie a breve termine e uno per le memorie a lungo termine.

Di seguito è brevemente illustrato il funzionamento di un'unità LSTM.

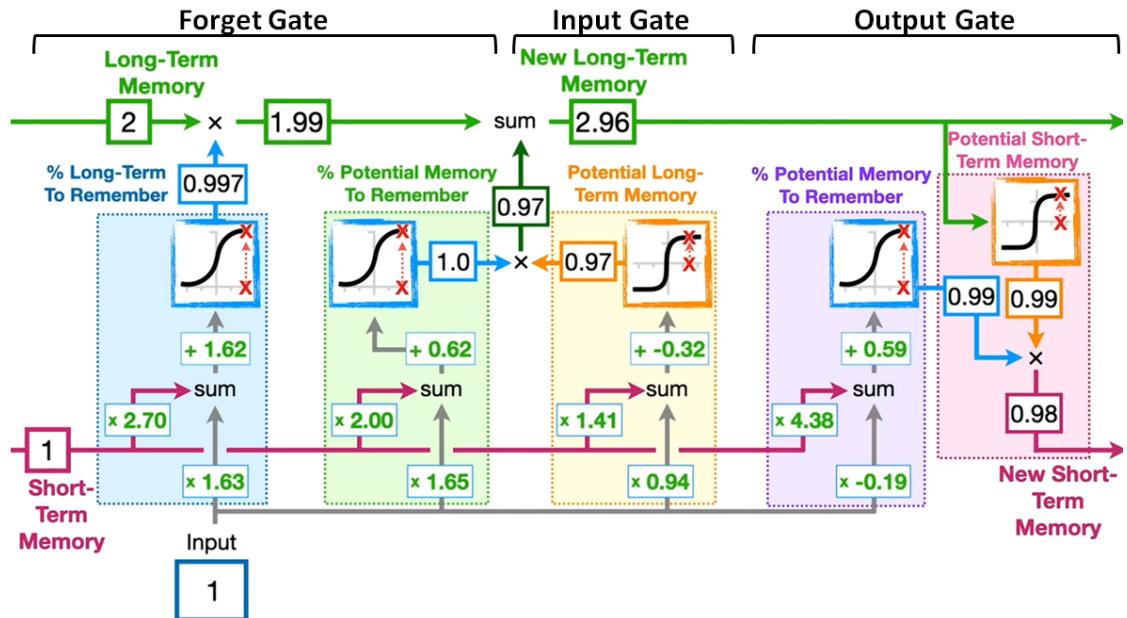


Figura 1.15: Architettura di una unit cell LSTM. I valori dei pesi, bias, input e output illustrati sono esempi per agevolare la comprensione.

Una premessa iniziale è che una LSTM utilizza come funzioni di attivazione Sigmoid e Tanh, definite come segue:

$$\text{Sigmoid}(x) = \frac{e^x}{e^x + 1}$$

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

La funzione **sigmoide** mappa ogni valore  $x$  nell'intervallo compreso tra 0 e 1, mentre la **tangente iperbolica** mappa ogni valore  $x$  nell'intervallo tra -1 e 1. Entrambe le funzioni operano su tutto l'asse reale, con il codominio limitato rispettivamente agli intervalli specificati.

La linea superiore, denominata "**Cell State**", rappresenta la memoria a lungo termine, e si può notare l'assenza di pesi o bias che possano modificarla direttamente. Grazie a questa caratteristica, è possibile gestire una serie di input senza incorrere nel Vanishing/Exploding Gradient Problem.

La linea inferiore, denominata "**Hidden State**", rappresenta le memorie a breve termine ed è direttamente influenzata da pesi che la modificano.

La unit cell di una LSTM è suddivisa in tre fasi o gate.

- La prima fase, nota come "**Forget Gate**", utilizza una funzione sigmoide per calcolare la percentuale della memoria a lungo termine da ricordare.
- La seconda fase è chiamata "**Input Gate**" e stabilisce come aggiornare la memoria a lungo termine. Il blocco a destra combina la memoria a breve termine e l'input per generare una memoria a lungo termine potenziata, mentre il blocco a sinistra determina la percentuale di questa memoria potenziata da aggiungere alla memoria a lungo termine.
- La terza e ultima fase è denominata "**Output Gate**" e ha il compito di aggiornare la memoria a breve termine. In questa fase, la memoria a lungo termine aggiornata funge da input per una funzione tangente iperbolica ( $\tanh$ ), generando così una nuova memoria a breve termine potenziata. Successivamente, la LSTM deve determinare la percentuale di questa memoria potenziata da conservare, utilizzando un'altra funzione sigmoide nel blocco a sinistra, seguendo lo stesso metodo utilizzato precedentemente in due occasioni. Moltiplicando la memoria potenziata per la percentuale da ricordare, si ottiene una nuova memoria a breve termine, che rappresenta l'output complessivo dell'intera cella LSTM.

Si evidenzia che tutte le funzioni di attivazione tanh sono impiegate per calcolare una memoria potenziale, mentre le sigmoidi sono utilizzate per determinare la percentuale di memoria potenziale che deve essere ricordata.

# Capitolo 2

## Metodologia di Sviluppo

Il presente capitolo si concentra sulle risorse e gli strumenti utilizzati durante il processo di creazione e implementazione del sistema di image captioning. Questi includono i dataset utilizzati per l’addestramento e la valutazione del modello, nonché le librerie software impiegate per la sua realizzazione.

### 2.1 Datasets

Nel corso di questa ricerca, sono stati adoperati tre dataset, di cui due selezionati da risorse preesistenti e uno appositamente creato per l’analisi. Tale approccio ha garantito una vasta gamma di dati, comprendente sia fonti esterne che un insieme personalizzato, fondamentale per lo sviluppo e la valutazione del sistema proposto.

#### 2.1.1 Flickr8k

Il dataset Flickr8k [6] rappresenta una scelta fondamentale per il presente studio, essendo una risorsa ampiamente impiegata nel campo dell’Image Captioning. Composto da un corpus di approssimativamente 8.000 immagini, ogni singola fotografia è accompagnata da cinque annotazioni testuali (caption) che dettagliano il suo contenuto visivo.

Le immagini presenti all’interno del dataset Flickr8k sono caratterizzate da una varietà eclettica, che spazia da ambientazioni urbane a paesaggi naturali, da scene do-

mestiche a ritratti umani. Tale eterogeneità visiva si rivela fondamentale nell'assicurare una rappresentazione ampia e completa del mondo reale, favorendo la generalizzazione dei modelli di Image Captioning su una vasta gamma di contesti visivi.

La qualità delle annotazioni testuali fornite insieme alle immagini rappresenta un ulteriore punto di forza di Flickr8k. Ogni fotografia è accompagnata da diverse descrizioni, consentendo di catturare molteplici sfaccettature dell'oggetto fotografato e offrendo un ampio spettro di informazioni linguistiche per l'addestramento dei modelli. Questa ricchezza di annotazioni non solo migliora la robustezza del sistema di generazione di didascalie, ma contribuisce anche a gestire con maggiore efficacia la varietà e la complessità delle descrizioni da generare.

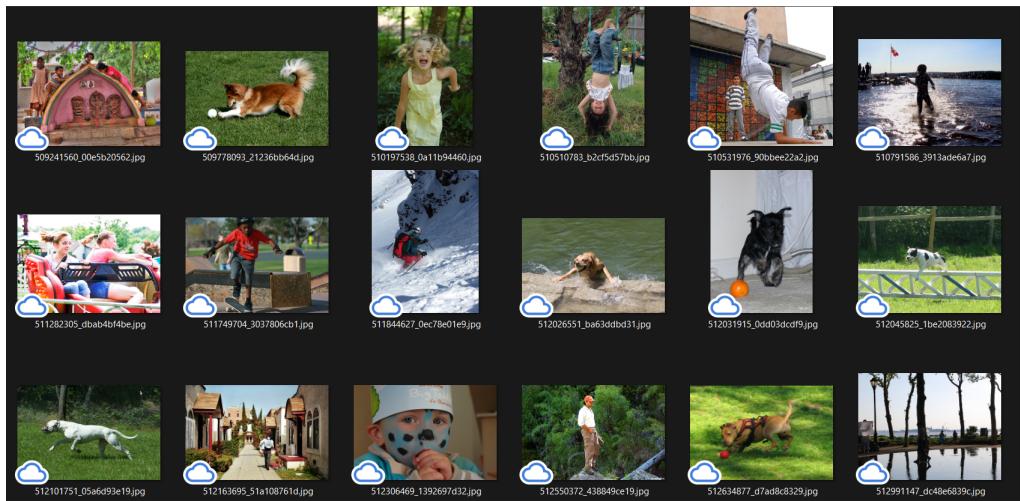


Figura 2.1: Alcune delle immagini presenti nel dataset Flickr8k.

```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg,A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg,A black dog and a white dog with brown spots are staring at each other in the street .
1001773457_577c3a7d70.jpg,Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg,Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg,A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
1002674143_1b742ab4b8.jpg,A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg,A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
1002674143_1b742ab4b8.jpg,There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg,Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg,A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg,A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg,a man sleeping on a bench outside with a white and black dog sitting next to him .
1003163366_44323f5815.jpg,A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg,man laying on bench holding leash of dog sitting on ground
```

Figura 2.2: Alcune delle descrizioni presenti nel dataset Flickr8k.

### 2.1.2 sketch-scene

Il dataset "sketch-scene" [7] [8] emerge come un'opzione significativa per l'indagine attuale nell'ambito dell'Image Captioning. Si compone esattamente di 10.000 immagini, ciascuna corredata da una singola annotazione testuale, distinguendosi come uno dei pochissimi dataset di sketch captioning reperibili online. Tuttavia, rispetto a Flickr8k, gode di una minore notorietà e un utilizzo limitato. Le immagini all'interno di questo dataset presentano una qualità visiva inferiore, caratterizzate da sketch poco chiari e descrizioni con frequenti errori grammaticali.

L'obiettivo primario del dataset "sketch-scene" è quello di raccogliere schizzi che trasmettano efficacemente il contenuto di una scena e che possano essere prodotti in pochi minuti da individui con varie competenze nel disegno. A tal fine, tali schizzi sono stati generati da 100 individui non esperti, assicurando così una vasta gamma di stili e livelli di dettaglio.

Nonostante le limitazioni in termini di qualità visiva e linguistica, il dataset "sketch-scene" si presenta come una risorsa preziosa per l'addestramento dei modelli di Image Captioning, offrendo un'alternativa unica e diversificata rispetto ai dataset più convenzionali.



Figura 2.3: Alcune delle immagini presenti nel dataset sketch-scene.

```

image.caption
img0.jpg,giraffe is eating leaves from the tree
img1.jpg,A zebra is eating grass
img2.jpg,people are riding on the horses
img3.jpg,two girafee's eating the tree leaves
img4.jpg,An areoplane, airfranceis flying
img5.jpg,Some birds are sitting on a branch of a tree
img6.jpg,People gathered over a park
img7.jpg,stop board in the park
img8.jpg,zebras walking in the grass

```

Figura 2.4: Alcune delle descrizioni presenti nel dataset sketch-scene.

### 2.1.3 Flickr8k Sketchified

Al fine di ampliare le risorse disponibili per l'indagine in corso, è stato sviluppato il dataset "Flickr8k Sketchified". Questo dataset rappresenta un'iniziativa originale volta a trasformare le immagini del ben noto dataset Flickr8k in sketch.

Per convertire un'immagine in sketch, è stata adottata una metodologia specifica.

```

1 # Leggo l'immagine
2 image = cv2.imread(image_path)
3 # Converto in grayscale
4 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
5 # Inverte
6 inverted_image = 255 - gray_image
7 # Blurro
8 blurred_image = cv2.GaussianBlur(inverted_image, (21, 21), 0)
9 # Applico Dodge blend
10 sketch = cv2.divide(gray_image, 255 - blurred_image, scale=256)

```

Questa trasformazione è stata realizzata utilizzando il linguaggio di programmazione Python insieme alla libreria OpenCV (cv2). Il codice proposto segue una sequenza di passaggi, ognuno dei quali contribuisce alla creazione dell'effetto finale dello schizzo.

Inizialmente, l'immagine viene letta utilizzando la funzione cv2.imread() di OpenCV. Successivamente, l'immagine viene convertita in scala di grigi utilizzando la funzione cv2.cvtColor().

Successivamente, i valori dei pixel dell'immagine vengono invertiti utilizzando l'operazione di sottrazione da 255, creando così un'immagine in negativo.

L’immagine invertita viene quindi sfocata utilizzando il filtro GaussianBlur per ridurre il rumore e ottenere una transizione più graduale tra le regioni chiare e scure. Questo passaggio aiuta a creare un effetto più “smoothed” nello sketch finale.

Infine, viene applicata un’operazione di miscelazione Dodge Blend utilizzando la funzione cv2.divide(). Questo passaggio combina l’immagine in scala di grigi originale con l’immagine sfocata invertita, producendo un effetto in cui le aree chiare diventano più luminose e gli elementi chiari dell’immagine si accentuano, mentre le zone scure rimangono invariate.

E così, dopo l’applicazione dell’operazione di Dodge Blend, si ottiene il risultato finale dello sketch.

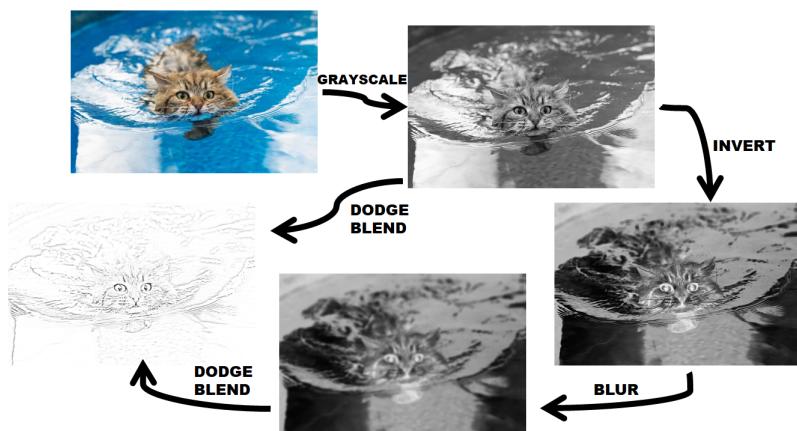


Figura 2.5: Illustrazione dell’intero processo di conversione di un’immagine in sketch.

Questa procedura è stata applicata a tutte le immagini del dataset Flickr8k, generando così un nuovo dataset composto da sketch, che è stato utilizzato come base per l’addestramento dei modelli di sketch captioning.

Nella Figura 3.1 sono illustrati esempi di immagini del dataset creato attraverso la procedura descritta. Le didascalie non sono mostrate poiché corrispondono esattamente a quelle di Flickr8k.

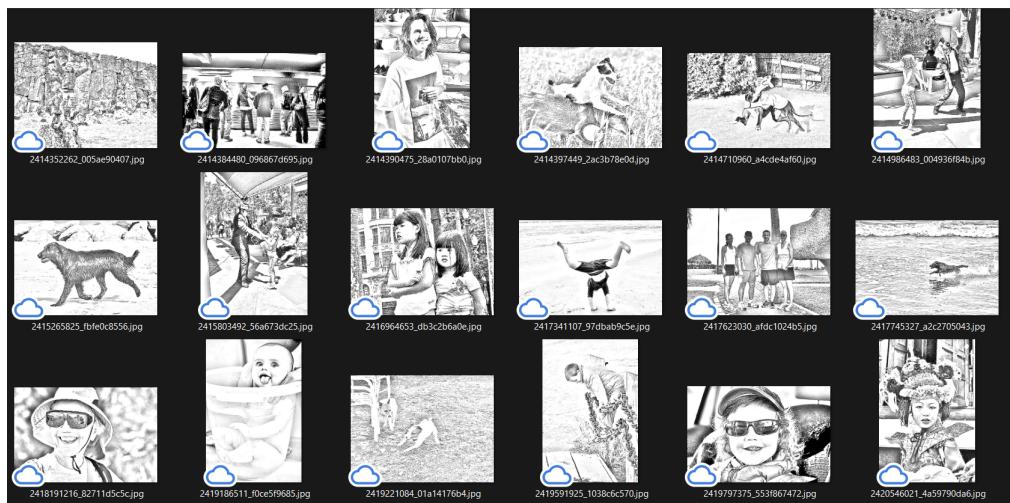


Figura 2.6: Alcune delle immagini presenti nel dataset Flickr8k Sketchified.

## 2.2 Librerie

Nel contesto della ricerca sull'Image Captioning, è stato fatto ampio uso di diverse librerie Python per agevolare lo sviluppo e l'analisi dei dati.

### 2.2.1 Keras/Tensorflow

TensorFlow è uno dei framework più utilizzati e supportati dalla comunità di machine learning. TensorFlow fornisce una solida base computazionale, offrendo funzionalità avanzate e potenza computazionale significativa, con le sue API basate su quelle di Keras nel contesto del deep learning e dell'addestramento di reti neurali.

Keras, a sua volta, è una libreria open source scritta in Python che fornisce un’interfaccia semplice e intuitiva per la creazione, l’addestramento e la valutazione di reti neurali. La modularità di Keras permette agli sviluppatori di creare reti neurali assemblando diversi strati in modo sequenziale o tramite modelli più complessi con più flussi di dati. Questa semplicità si riflette nella sua sintassi chiara e intuitiva, rendendo più facile per gli sviluppatori concentrarsi sulla progettazione del modello piuttosto che sulla sua implementazione dettagliata. Keras è estendibile, consentendo agli sviluppatori di adattarlo alle esigenze specifiche del progetto con nuovi layer, funzioni di attivazione, funzioni di perdita, ottimizzatori e altro ancora. Inoltre, offre un’interoperabilità con

TensorFlow, permettendo agli utenti di sfruttare la potenza e le funzionalità avanzate di TensorFlow all'interno dell'ambiente Keras.

Di seguito sono riportate le librerie di Keras che sono state fondamentali per la creazione dei modelli.

```
1 from keras.applications.xception import Xception, preprocess_input
2 from keras.preprocessing.image import load_img, img_to_array
3 from keras.preprocessing.text import Tokenizer
4 from keras.preprocessing.sequence import pad_sequences
5 from keras.utils import to_categorical, plot_model
6 from keras.models import Model, load_model
7 from keras.layers import Input, Dense, LSTM, SimpleRNN, Embedding, Dropout, add
8 from keras.optimizers import Adam
```

### 2.2.2 Altre Librerie

Nel contesto della ricerca, mentre TensorFlow e Keras sono state le librerie principali utilizzate per lo sviluppo dei modelli, altre librerie hanno svolto un ruolo significativo nel processo di analisi e manipolazione dei dati.

Tra queste, la libreria **numpy** ha rappresentato un pilastro fondamentale, supportando operazioni su array multidimensionali e calcoli matematici essenziali per il trattamento delle immagini e l'estrazione delle relative caratteristiche.

Parallelamente, l'utilizzo della libreria **PIL** (Python Imaging Library) è stato cruciale per la manipolazione e la gestione delle immagini.

Per la gestione dei file e delle operazioni di sistema, è stata impiegata la libreria **os**, che ha consentito di interagire in modo efficiente con il sistema operativo.

Inoltre, per la serializzazione e deserializzazione degli oggetti Python, è stato fatto affidamento su **pickle**, sfruttando le sue funzionalità di dump e load per salvare e caricare i modelli in modo efficiente.

Nel processo di preprocessing dei testi, sono state integrate le funzionalità offerte dalle librerie **ntlk**, **string** e **re**. **Ntlk** è stata utilizzata per la lemmatizzazione dei testi, una tecnica che ha consentito di ridurre le parole alla loro forma base, migliorando così la coerenza e la qualità delle didascalie generate. D'altra parte, **string** e **re** sono state utilizzate per operazioni di pulizia del testo, quali la rimozione della punteggiatura, la gestione dei caratteri speciali e altre operazioni di normalizzazione.

Per quanto riguarda la visualizzazione dei risultati e l'analisi dei dati, sono state sfruttate le potenzialità delle librerie **matplotlib** e **seaborn**. **Matplotlib** è stata la scelta principale per la creazione di grafici chiari e informativi, mentre **seaborn** ha fornito strumenti aggiuntivi per la visualizzazione e l'analisi avanzata dei dati.

Infine, è stata utilizzata la libreria **tqdm** per monitorare la progressione delle iterazioni nei cicli, offrendo una visione chiara dell'avanzamento dei processi.

# Capitolo 3

## Approccio Tecnico

Il Capitolo 3 di questa Tesi illustra dettagliatamente l’implementazione tecnica del modello Encoder-Decoder, descrivendo le fasi di estrazione delle caratteristiche visive, la pre-elaborazione dei testi, la creazione e il training del modello, e l’uso di differenti tecniche di generazione delle descrizioni, supportate da metriche per valutare la qualità delle descrizioni generate.

### 3.1 Feature Extraction

Una volta importate le librerie necessarie e caricato il dataset specifico, si comincia con la fase di estrazione delle features.

Come specificato nella sezione 1.2.2, si usano reti neurali convoluzionali (CNN) pre-addestrate sul dataset ImageNet. Queste reti possiedono già una vasta conoscenza nel campo della computer vision. Pertanto, i pesi della CNN sono già pre-inizializzati per l’attività di classificazione delle immagini su ImageNet, evitando così la necessità di ottimizzarli da zero.

La CNN pre-addestrata opera come Encoder per il modello Encoder-Decoder.

In pratica, le immagini del dataset vengono fornite in input alla CNN, che le elabora restituendo in output una rappresentazione vettoriale compatta chiamata feature vector, la quale cattura le informazioni visuali significative.

```
1 # Per Xception  
2 model = Xception(include_top=False, pooling='avg', weights='imagenet')
```

```
3
4 # Per VGG16
5 model = VGG16(weights='imagenet')
6 model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
```

Prima viene istanziato il modello, specificando `weights='imagenet'` per caricare i pesi preaddestrati nella rete. Poiché queste reti sono state progettate per l'Image Classification, mentre il compito attuale è diverso, è necessario adattare la struttura dei modelli. Gli strati responsabili della classificazione vengono rimossi. Nel caso della VGG16, viene utilizzato l'output del penultimo strato (`model.layers[-2].output`). Nel caso di Xception, viene impostato `include_top=False`.

A questo punto, è stata definita una funzione per estrarre le feature dalle immagini del dataset, e che allo stesso tempo crea una mappa denominata “features” che associa ad ogni immagine il suo feature vector. Per ogni immagine, il processo comprende:

- Lettura dell'immagine dal disco;
- Ridimensionamento dell'immagine (`image.resize`) per adattarla alle dimensioni di input accettate dalla CNN (224x224 per VGG16 e 299x299 per Xception);
- Preprocessamento dell'immagine;
- Utilizzo del metodo `model.predict` per estrarre le feature tramite la CNN.

In conclusione, viene creata una mappa che associa a ciascuna immagine il corrispondente feature vector.

Pur essendo un feature vector una rappresentazione astratta dell'aspetto dell'immagine, nella figura 3.2, si propone un'illustrazione del feature vector di un'immagine di un cane presa dal dataset Flickr8k, estratto con la rete Xception, dove ogni riga è una feature e ogni colonna il valore.

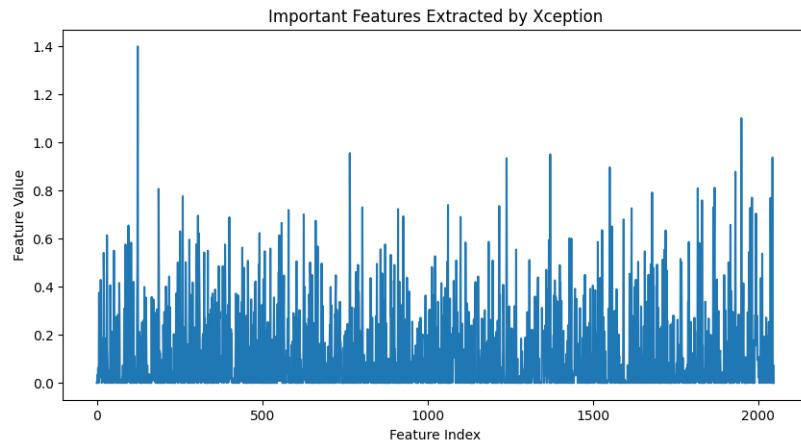


Figura 3.1: Feature Vector estratto da un’immagine di un cane.

La particolarità dei feature vector è che tendono ad avere caratteristiche simili per immagini simili, come evidenziato nell’esempio seguente.

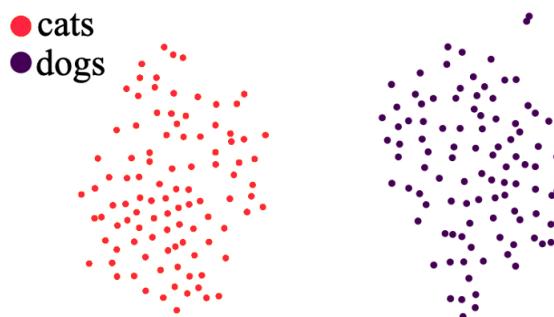


Figura 3.2: Esempio di distribuzione 2D dei Feature Vector per 2 classi: gatti e cani.

### 3.2 Mapping Image\_ID - Captions

Il passo successivo nel processo di image captioning comporta la creazione di una mappa che stabilisce un collegamento diretto tra ciascuna immagine e le relative captions associate. Questa mappa è implementata mediante l’utilizzo di un dizionario, dove ogni immagine è identificata univocamente attraverso un ID, e a tale ID sono associate una o più didascalie che la descrivono in modo significativo. È importante sottolineare che per il dataset di Flickr8k, ogni immagine è associata a cinque captions diverse, mentre per il dataset di sketch-scene, ad ogni immagine è associata una sola caption.

Il motivo principale per la creazione di questa mappa è per la convenienza nell'organizzare tutti gli elementi del dataset in una singola variabile, denominata "**mapping**". Questa variabile rende più agevole l'accesso e la gestione delle didascalie associate a ciascuna immagine all'interno del dataset, facilitando così l'iterazione e l'analisi dei dati durante le fasi successive del processo di image captioning.

### 3.3 Text Preprocessing

Il testo delle didascalie deve essere adeguatamente elaborato prima di essere utilizzato per l'addestramento del modello. Questo processo, noto come text preprocessing, comprende una serie di operazioni volte a pulire e standardizzare il testo delle didascalie al fine di rendere più agevole l'estrazione delle informazioni rilevanti durante la fase di addestramento.

#### 3.3.1 Cleaning

Inizialmente, per ogni singola caption, vengono eseguiti i seguenti passaggi:

```
1 caption = caption.lower()
2 caption = re.sub(r'\w\W\s+', ' ', caption)
3 caption = re.sub(r'\s+', ' ', caption)
```

- Il testo viene convertito in minuscolo (lowercase);
- Vengono rimossi i caratteri che non sono alfanumerici;
- Gli spazi multipli vengono sostituiti con uno spazio singolo, garantendo così la coerenza nella formattazione del testo.

#### 3.3.2 Lemmatization

Successivamente, è stata adottata la tecnica della **Lemmatization**. Questo processo algoritmico, ampiamente utilizzato nel campo del Natural Language Processing, consente di ridurre le parole alla loro forma base, chiamata "lemma".

Questa procedura riveste un'importanza significativa poiché comporta una riduzione del numero complessivo di parole nel dizionario. Di conseguenza, semplifica il processo

di addestramento del modello, poiché il modello dovrà gestire un numero inferiore di varianti lessicali per ciascuna parola.

Per illustrare l'importanza di questo approccio, si prendano in considerazione gli esempi seguenti:

Cats → Cat, People → Person (esempi di Lemmatization)

La parola "cats" viene ridotta al suo lemma "cat", mentre "people" diventa "person". Riduzioni come queste contribuiscono alla creazione di un dizionario più conciso e coerente.

E' stata preferita l'uso della Lemmatization rispetto allo Stemming, poiché lo Stemming, che riduce le parole alla loro forma radice eliminando prefissi o suffissi, può generare errori.

Male|s → Male, Play|ing → Play, R|ing → ?? (esempi di Stemming)

Questa tecnica, basata su regole predefinite come l'eliminazione di suffissi come "-ing" o "-s", non sempre produce risultati coerenti. Ad esempio, la parola "ring" verrebbe ridotta a "r", risultando priva di significato. Nonostante la sua semplicità d'uso, lo Stemming è meno efficiente rispetto alla Lemmatization, la quale riduce le parole alla loro forma base utilizzando un approccio basato sul dizionario, garantendo così una maggiore coerenza nel processo di riduzione delle parole.

Per utilizzare la Lemmatization, nel codice è stata prevista l'installazione del database inglese WordNet, che fornisce correlazioni semantiche tra le parole.

```
1 nltk.download('wordnet')
```

### 3.3.3 Ulteriori Azioni di Preprocessing del Testo

Infine, sono stati aggiunti i tag di inizio e fine frase, "startseq" e "endseq", a ciascuna didascalia. Essendo le frasi di lunghezze diverse e non una prefissata, l'aggiunta dei tag fornisce un'indicazione chiara sulle estremità di ciascuna frase.

```
1 cleaned_caption = 'startseq ' + cleaned_caption + ' endseq'
```

È stato inserito un passaggio aggiuntivo esclusivamente per il dataset personalizzato "Flickr8k Sketchified". Questo dataset è stato creato trasformando le immagini in sketch, mantenendo però le stesse didascalie originali. Tuttavia, poiché le immagini sono state convertite in bianco e nero, mentre le didascalie sono rimaste invariate, è stato necessario rimuovere manualmente dal dizionario delle captions tutte le parole relative ai colori.

Di seguito viene mostrata la differenza tra le captions di un'immagine di Flickr8k prima e dopo il preprocessing.

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
'A girl going into a wooden building .',
'A little girl climbing into a wooden playhouse .',
'A little girl climbing the stairs to her playhouse .',
'A little girl in a pink dress going into a wooden cabin .']
```

```
['startseq a child in a pink dress is climbing up a set of stairs in an entry way endseq',
'startseq a girl going into a wooden building endseq',
'startseq a little girl climbing into a wooden playhouse endseq',
'startseq a little girl climbing the stairs to her playhouse endseq',
'startseq a little girl in a pink dress going into a wooden cabin endseq']
```

Figura 3.3: Captions di un'immagine di Flickr8k prima e dopo il preprocessing.

È importante sottolineare che nel caso di Flickr8k Sketchified, l'unica differenza riscontrata è l'assenza del termine "pink", in quanto identifica un colore.

Nel caso del dataset sketch-scene, vengono presentate cinque diverse immagini, ciascuna con la propria caption.

```
['startseq giraffe is eating leaf from the tree endseq'],
['startseq a zebra is eating grass endseq'],
['startseq people are riding on the horse endseq'],
['startseq some bird are sitting on a branch of a tree endseq'],
['startseq people gathered over a park endseq']
```

```
giraffe is eating leaves from the tree
A zebra is eating grass
people are riding on the horses
Some birds are sitting on a branch of a tree
People gathered over a park
```

Figura 3.4: Captions di un'immagine di sketch-scene prima e dopo il preprocessing.

Prima di procedere con la fase finale del Text Preprocessing, è stata adottata una scelta per comodità e semplicità: la creazione di una lista denominata "**all\_caption**s". Questa lista è stata ottenuta trasformando il dizionario "mapping" in una lista di tutte le didascalie presenti, consentendo un'elaborazione più agevole dei dati. Questa operazione consente di avere tutte le captions in una lista unica.

La lunghezza di questa lista, rappresentata da **len(all\_caption)**, sarà diversa a seconda del dataset considerato. Nel caso del dataset sketch-scene, contenente 10.000

immagini ciascuna con una caption, la lunghezza sarà di 10.000 elementi. Tuttavia, nel dataset Flickr8k, dove ogni immagine è associata a cinque didascalie diverse, la lunghezza sarà di 40.455 elementi (8.094 immagini moltiplicate per 5 captions ciascuna).

### 3.3.4 Tokenization

La tokenizzazione costituisce l'ultimo passaggio nella fase di Preprocessamento del Testo. Si tratta di un'operazione fondamentale nell'ambito dell'elaborazione del linguaggio naturale (NLP), che prevede la suddivisione di un testo in unità più piccole, note come token.

Nel contesto di questo lavoro, la tokenizzazione è stata implementata utilizzando il metodo **fit\_on\_texts** della classe Tokenizer fornita dalla libreria Keras.

```
1 tokenizer.fit_on_texts(all_captions)
```

Questo metodo richiede un input sequenziale e non accetta strutture dati complesse come le mappe. Pertanto, è stata creata la lista `all_captions` in precedenza, che contiene tutte le didascalie del dataset. Applicando il metodo `fit_on_texts`, tutte le captions vengono suddivise in token individuali, generando così un vocabolario che contiene tutte le parole presenti nelle didascalie.

Ogni parola nel vocabolario viene quindi assegnata a un indice univoco che riflette la sua frequenza di apparizione complessiva nel dataset. Infine, si è stampato il dizionario **tokenizer.word\_index**, che associa ogni parola nel vocabolario al suo indice assegnato. Le parole che compaiono più frequentemente nelle didascalie avranno indici più bassi, mentre quelle meno comuni avranno indici più alti.

Di seguito sono riportati esempi dei token iniziali per i vari dataset:

```
{'a': 1,
'startseq': 2,
'endseq': 3,
'in': 4,
'the': 5,
'on': 6,
'dog': 7,
'is': 8,
'and': 9,
'with': 10,
'man': 11,
'of': 12,
'two': 13,
```

(a) Parole più comuni nel vocabolario del dataset Flickr8k.

```
{'startseq': 1,
'endseq': 2,
'a': 3,
'the': 4,
'in': 5,
'on': 6,
'is': 7,
'with': 8,
'flying': 9,
'are': 10,
'standing': 11,
'of': 12,
'and': 13,
```

(b) Parole più comuni nel vocabolario del dataset sketch-scene.

Figura 3.5: Confronto tra i vocabolari dei dataset.

Risulta evidente che i token startseq e endseq sono tra i più comuni in entrambi i dizionari, poiché ognuno di essi è presente esattamente una volta in tutte le didascalie.

Di seguito sono riportate le dimensioni complessive dei dizionari relativi a ciascun caso analizzato:

Dataset	Senza Lemmatization	Con Lemmatization
sketch-scene	2376	2122
flickr8k	8811	7620
flickr8k sketchified	-	7605

Tabella 3.1: Dimensioni del vocabolario dei dataset con e senza lemmatizzazione.

Questi dati evidenziano come l'applicazione della Lemmatization, indipendentemente dal dataset utilizzato, abbia portato a una significativa riduzione del numero di parole nel dizionario. In ogni caso, si è osservata una diminuzione di almeno il 10%.

Nel caso specifico del dataset Flickr8k Sketchified, il numero di token nel dizionario è uguale a quello del dataset Flickr8k, fatta eccezione per le parole relative ai colori che erano state precedentemente rimosse.

### 3.3.5 max\_length

```
1 max_length = max(len(caption.split()) for caption in allCaptions)
```

Viene ricavato il valore `max_length`, che indica la lunghezza massima tra tutte le captions presenti nel dataset. Questo valore sarà utilizzato per aggiungere padding alle sequenze di testo più corte in modo che abbiano tutte la stessa lunghezza, pari a `max_length`. Questa pratica sarà vantaggiosa durante l'addestramento dei modelli neurali, poiché le reti neurali richiedono input di dimensioni fisse. È pertanto necessario uniformare la lunghezza di tutte le sequenze di testo affinché soddisfino questo requisito.

### 3.3.6 Istogramma della Lunghezza delle Didascalie

È stato ritenuto necessario generare un istogramma che visualizza la distribuzione delle lunghezze delle didascalie nel dataset. Per creare questo istogramma, sono state utilizzate entrambe le librerie Matplotlib e Seaborn. Ogni barra sull'istogramma rappresenta un intervallo di lunghezza delle didascalie, mentre l'altezza della barra indica la frequenza con cui le didascalie cadono in quell'intervallo, ossia quante sono le captions con quella particolare lunghezza.

Di seguito sono confrontati gli istogrammi relativi ai due dataset:

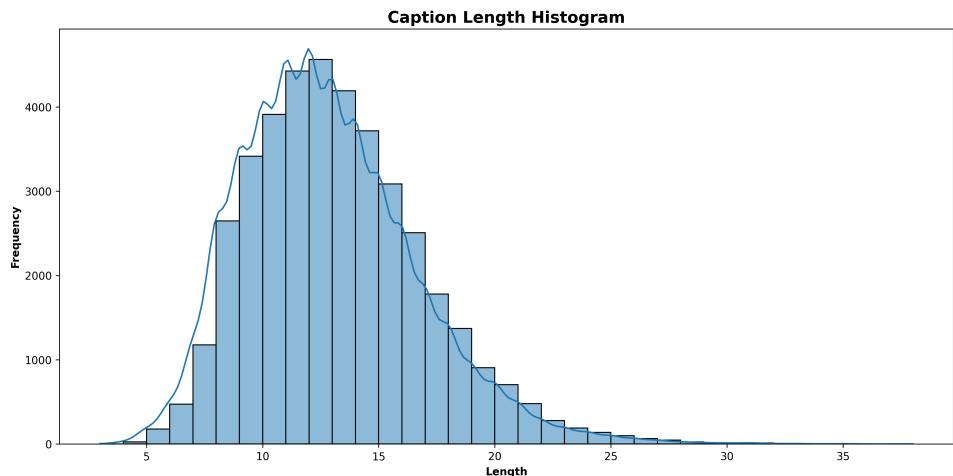


Figura 3.6: Istogramma della lunghezza delle captions, per il dataset Flickr8k.

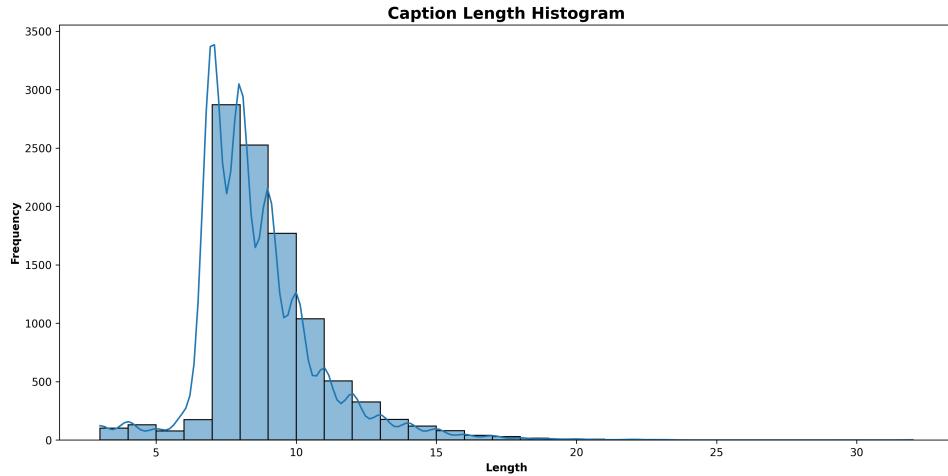


Figura 3.7: Istogramma della lunghezza delle captions, per il dataset sketch-scene.

L’istogramma relativo al dataset Flickr8k evidenzia una distribuzione equilibrata delle lunghezze delle captions, con una moda concentrata intorno ai 12 tokens.

In contrasto, l’analisi dell’istogramma del dataset sketch-scene evidenzia una moda delle lunghezze delle caption significativamente inferiore, concentrata tra 7 e 8 tokens, e una distribuzione sbilanciata delle stesse. È degna di nota la presenza frequente di captions con meno di 5 parole nel dataset sketch-scene. Questa osservazione solleva un possibile problema: le descrizioni più brevi potrebbero non essere in grado di fornire una visione completa o accurata delle immagini associate. Tale osservazione suggerisce una qualità inferiore delle caption nel dataset sketch-scene, il che può essere interpretato come un’indicazione della minore coerenza o della minor complessità semantica delle descrizioni fornite in confronto al dataset Flickr8k.

### 3.4 Training-Validation-Test Split

Nel presente contesto, i dati sono stati suddivisi in tre insiemi principali: training set, validation set e test set, con una distribuzione percentuale del 80% - 10% - 10%, rispettivamente. La divisione è stata eseguita manualmente, piuttosto che fare affidamento su funzioni predefinite, al fine di ottenere un maggiore controllo sul processo di suddivisione dei dati.

- Training Set: Questo set è stato costituito dall' 80% del totale delle immagini disponibili nel dataset. Il training set serve per consentire al modello di imparare i pattern e le caratteristiche dei dati di input.
- Validation Set: Il restante 20% delle immagini è stato suddiviso a metà per formare sia il set di validazione che il set di test, ciascuno contenente il 10% del totale delle immagini. Il validation set è utilizzato per valutare le prestazioni del modello durante il processo di addestramento. Serve per regolare gli iperparametri del modello e prevenire l'overfitting, consentendo di identificare e correggere eventuali problemi di performance del modello durante la fase di training.
- Test Set: Il restante 10% delle immagini è stato destinato al set di test. Questo set serve per valutare le prestazioni finali del modello dopo il completamento del processo di addestramento. Consente di valutare l'efficacia del modello su dati non visti durante l'addestramento e la validazione, fornendo una stima accurata delle prestazioni del modello in un contesto realistico.

### 3.5 Creazione del Modello Encoder-Decoder

Questa sezione spiega il processo di implementazione del modello Encoder-Decoder che rende possibile l'Image Captioning.

L'architettura della rete è costituita da due componenti principali: un'unità di estrazione delle caratteristiche (**encoder**) e un'unità di generazione del testo (**decoder**).

L'**encoder** riceve in input i feature vectors estratti precedentemente dalle immagini, che è uguale all'output restituito dalla CNN pre-addestrata. I feature vectors estratti usando la rete Xception hanno 2048 dimensioni, mentre per la VGG16 ne hanno 4096. Queste dimensioni sono ridotte a 256 neuroni, usando uno strato completamente connesso con attivazione ReLU. Questo migliora considerevolmente la velocità di addestramento e previene l'errore di "memoria esaurita". Infine viene aggiunto uno strato di dropout per ridurre il rischio di overfitting.

Il **decoder**, invece, riceve sequenze di parole della didascalia come input. Queste parole vengono convertite in vettori tramite uno strato di **word embedding**. Il word

embedding è una tecnica utilizzata nel campo del Natural Language Processing (NLP) per rappresentare le parole in un formato numerico, consentendo ai modelli di apprendere relazioni semantiche tra le parole e di comprendere il contesto in cui vengono utilizzate. Analogamente ai feature vector delle immagini, che raggruppano immagini simili, il word embedding mappa ogni parola del vocabolario in uno spazio vettoriale multidimensionale, dove parole semanticamente simili sono rappresentate da vettori vicini tra loro. Di conseguenza, i vettori di embedding delle parole possono essere paragonati ai feature vector delle immagini, che talvolta vengono denominati “image embeddings”. Successivamente, viene applicato uno strato di dropout per la regolarizzazione. Infine, le sequenze di vettori sono elaborate tramite una rete neurale ricorrente (RNN), come LSTM o SimpleRNN. È stata esplorata l’efficacia di diverse architetture di RNN al fine di analizzare eventuali differenze nei risultati ottenuti.

Le uscite dell’encoder e del decoder sono quindi combinate e passate attraverso ulteriori fully connected layer con attivazione ReLU, culminando in un livello di output con attivazione softmax che normalizza e restituisce la probabilità di ogni token nel vocabolario.

Il modello viene allenato minimizzando la **categorical cross-entropy loss**, utilizzando l’ottimizzatore Adam con un tasso di apprendimento **alpha = 0.001**. Durante l’addestramento, il modello predice una parola ad ogni istante temporale, corrispondente all’output con la probabilità più alta tra tutti i token del vocabolario. La funzione di loss misura la differenza tra l’output predetto e l’output di riferimento.

Infine, la struttura complessiva del modello viene visualizzata graficamente (figure 3.8 e 3.9) utilizzando il metodo **plot\_model**.

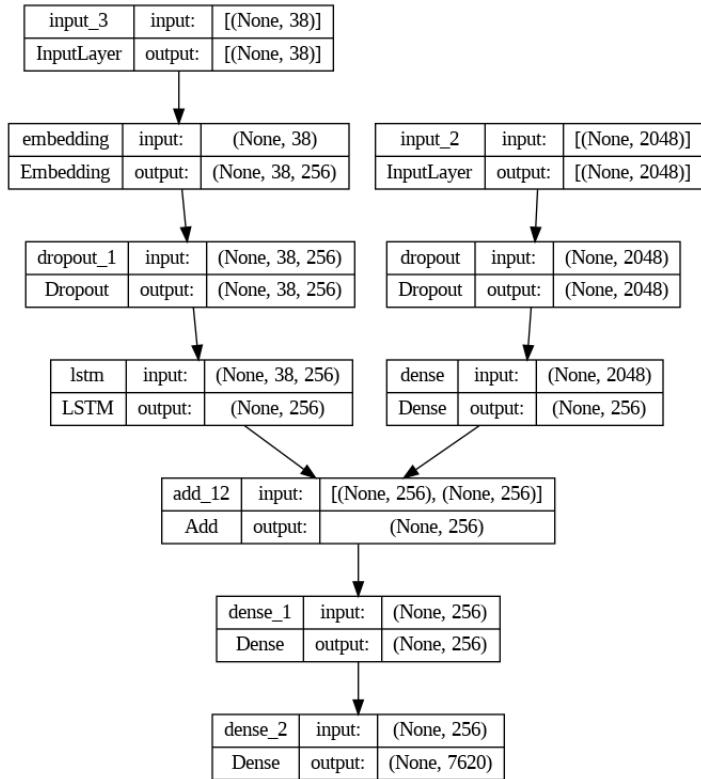


Figura 3.8: Modello Encoder-Decoder per Image Captioning, usando una rete Xception come CNN e una LSTM come RNN.

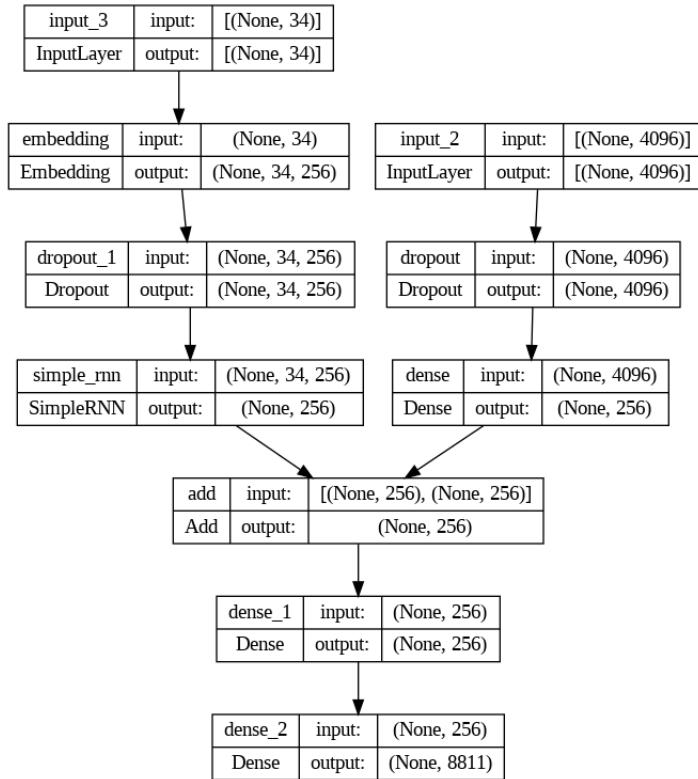


Figura 3.9: Modello Encoder-Decoder per Image Captioning, usando una rete VGG16 come CNN e una SimpleRNN come RNN.

## 3.6 Training e Data Generator

In questa sezione, viene illustrato il processo di addestramento del modello. Per una comprensione completa, è necessario delineare la funzione “**data\_generator**”, impiegata per la suddivisione dei dati in batch durante il processo di apprendimento.

### 3.6.1 Data Generator

La necessità di utilizzare la funzione `data_generator` deriva dalla vastità del training set, che contiene un numero significativo di immagini (6400 nel caso di Flickr8k, 7000 nel caso di sketch-scene), ciascuna delle quali è rappresentata da un feature vector di lunghezza 2048 e una caption rappresentata come una sequenza di numeri. La quantità di dati associata a tutte queste immagini non è gestibile in memoria. Pertanto,

per ottimizzare l'efficienza computazionale, è necessario adottare un metodo che consenta di processare i dati a gruppi. La funzione `data_generator` svolge proprio questo ruolo, permettendo di suddividere i dati in batch di dimensioni specificate, agevolando un'elaborazione efficiente e una gestione ottimizzata dei dati. Questa pratica è importantissima per evitare eventuali problemi di memoria, come il session crash, e per garantire una maggiore stabilità durante il training del modello.

La funzione `data_generator` accetta diversi parametri in input:

- `data_keys`: l'elenco delle chiavi corrispondenti agli identificatori delle immagini nel dataset;
- `mapping`: il dizionario che associa ciascuna immagine alle rispettive captions;
- `features`: il dizionario che associa ogni immagine al relativo feature vector estratto;
- `tokenizer`: l'oggetto `tokenizer` utilizzato per convertire le parole in sequenze di token;
- `max_length`: la lunghezza massima prevista per le didascalie, in termini di numero di token;
- `vocab_size`: la dimensione del vocabolario, ossia il numero totale di token diversi nel corpus;
- `batch_size`: la dimensione desiderata del batch di dati.

All'interno della funzione, si utilizza un ciclo `while` che viene eseguito continuamente. Per ogni iterazione del ciclo, vengono processate le didascalie associate ad ogni immagine nel dataset.

Ogni didascalia viene quindi convertita in una sequenza di token utilizzando il `tokenizer`. Successivamente, questa sequenza di token viene suddivisa in coppie di input-output, dove l'input è una porzione della sequenza e l'output è la parola successiva nella sequenza.

Dopo la suddivisione delle sequenze, viene eseguito il padding dell'input per uniformare la lunghezza delle sequenze e viene applicata la codifica one-hot dell'output per rappresentare le parole come vettori binari.

Questa procedura è chiaramente illustrata nell'esempio seguente:

Tabella 3.2: Processo di generazione delle descrizioni delle immagini.

<b>X</b>	<b>y</b>
startseq	dog
startseq dog	walking
startseq dog walking	in
startseq dog walking in	grass
startseq dog walking in grass	endseq

Questo processo illustra come vengono generate le sequenze di input e output per l'addestramento del modello di Image Captioning. Durante l'addestramento, il modello riceve ciascuna sequenza di input insieme al feature vector dell'immagine corrispondente e predice la parola successiva nella sequenza di output. Ovviamente, nella pratica, il modello non opera direttamente con le parole come nel semplice esempio sopra, ma utilizza il tokenizer definito prima per convertire ogni parola in un indice numerico corrispondente. Quindi, la sequenza "startseq dog walking in grass endseq" viene convertita in una sequenza di indici numerici, ad esempio "[1, 7, 58, 4, 2]", dove ogni numero rappresenta l'indice della parola nel vocabolario. Questo processo ricorsivo consente al modello di generare descrizioni coerenti delle immagini. Nella pratica dell'Image Captioning, la sequenza di parole della didascalia funge da input per la rete neurale ricorrente (RNN), con ogni parola rappresentata come un passaggio temporale. L'input iniziale alla RNN è il token di inizio "startseq", e successivamente, l'output generato diventa l'input per il passo successivo, insieme al feature vector dell'immagine. Questo processo continua fino a quando il modello genera il token di fine "endseq", segnalando il completamento della didascalia. Infine, le sequenze di input e output vengono memorizzate in liste e convertite in array numpy quando il numero di esempi raggiunge la dimensione del batch specificata, e vengono restituite come output della funzione.

### 3.6.2 Training

Dopo aver definito il modello e il generatore di dati, è possibile avviare l'addestramento del modello di Image Captioning.

Gli **iperparametri** del modello sono stati impostati in precedenza: il tasso di apprendimento **alpha** è stato fissato a 0.001 durante la definizione del modello Encoder-Decoder, mentre la **dimensione del batch** è stata selezionata come 64. Il **numero di epoche** di addestramento varierà in base all'andamento della loss sul validation set, ossia sarà interrotto quando la loss non diminuirà più.

Il numero di step per epoca è ricavato dividendo la lunghezza del set di addestramento per la dimensione del batch. Analogamente, il numero di passi per la validazione (val\_steps) viene calcolato sulla base della lunghezza del set di validazione.

Durante il processo, vengono creati due generatori di dati distinti: uno per il training set e uno per il validation set, utilizzando la funzione `data_generator` definita in precedenza.

Per ciascuna epoca, il modello viene addestrato utilizzando i dati ottenuti dai due generatori di dati dedicati. Questo metodo permette di iterare sull'intero dataset senza dover caricare l'intero dataset in memoria contemporaneamente, garantendo una gestione efficiente delle risorse computazionali. L'addestramento viene effettuato chiamando la funzione `fit` del modello.

Questo approccio implica l'utilizzo della tecnica di ottimizzazione nota come **Stochastic Gradient Descent** (SGD) [9]. Con la SGD, i pesi del modello vengono aggiornati iterativamente utilizzando un singolo batch di dati alla volta anziché l'intero dataset. Ogni iterazione dell'addestramento coinvolge un sottoinsieme casuale dei dati disponibili, rendendo il processo di ottimizzazione più dinamico e scalabile. Grazie a questa strategia, il modello può convergere verso un minimo locale della funzione di loss in modo più rapido ed efficiente.

Durante questa fase, vengono monitorate le loss sia sul training set che sul validation set, al fine di valutare le prestazioni del modello e prevenire l'overfitting.

### 3.7 Metodi di Generazione delle Didascalie

[10] Questa sezione delinea il processo di generazione di didascalie per le immagini tramite due strategie: **Greedy** e **Beam Search**, implementate manualmente. Attraverso

un confronto diretto, sono analizzate le differenze tra i due approcci, esemplificandone l'efficacia e valutando i relativi costi computazionali.

Di conseguenza, per ciascuna immagine saranno generate due didascalie distinte: una con la funzione `predict_caption_greedy` e l'altra con `predict_caption_beam_search`.

Prima di descrivere la procedura di generazione della caption, è opportuno menzionare la funzione di supporto `idx_to_word`. Questa funzione svolge un'operazione di mapping inverso, consentendo la traduzione da indici numerici a parole del vocabolario utilizzando il tokenizer.

### 3.7.1 Greedy

La funzione `predict_caption_greedy` genera una didascalia per un'immagine utilizzando l'algoritmo Greedy.

Inizia aggiungendo il token di inizio "startseq" alla sequenza di testo in input.

Successivamente, per ogni parola nella didascalia (entro la lunghezza massima consentita), l'algoritmo esegue le seguenti operazioni:

1. Codifica la sequenza di testo in input, convertendo le parole in interi utilizzando il tokenizer;
2. Effettua il padding della sequenza di input per adattarla alla lunghezza massima `max_length`;
3. Predice la parola successiva utilizzando il metodo `predict` del modello. L'input per la previsione è costituito dal feature vector dell'immagine e dalla caption di testo codificata fino a quel punto;
4. Seleziona l'indice con la probabilità più alta dalla distribuzione di probabilità predetta;
5. Converte l'indice predetto in una parola utilizzando la funzione `idx_to_word`;
6. Aggiunge la parola predetta alla sequenza di testo in input per generare la parola successiva;

7. Termina il processo se viene predetto il token di fine "endseq" o se la lunghezza massima max\_length è stata raggiunta.

In breve, l'algoritmo Greedy seleziona la parola con la massima probabilità predetta in ciascun passaggio, e genera la didascalia parola per parola fino alla fine, ossia fino alla lunghezza massima consentita o finché non viene predetto il token ‘endseq’.

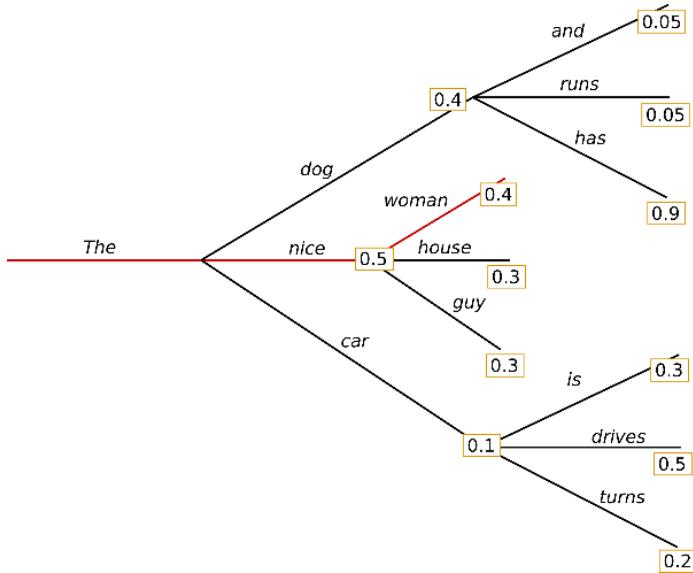


Figura 3.10: Meccanismo di generazione di una caption, usando la tecnica Greedy (rosso).

Nell'esempio fornito, la funzione predict\_caption\_greedy selezionerebbe la parola con la probabilità più alta per ogni istante, determinando la sequenza `["The", "nice", "woman"]`, con una probabilità totale di  $0.5 \cdot 0.4 = 0.20$ .

L'approccio precedentemente utilizzato si basa sul metodo Greedy, il quale, ad ogni chiamata della funzione `model.predict`, genera una distribuzione di probabilità su ciascun token del vocabolario, selezionando sempre il token con la massima probabilità per costituire la successiva parola della caption.

In altre parole, il metodo Greedy tende a selezionare sempre la scelta **localmente ottima**, che può risultare efficace con un costo computazionale contenuto. Tuttavia, c'è il rischio che ignori o non identifichi la soluzione **globalmente ottima**, ovvero la caption in cui il prodotto delle probabilità di tutte le parole è massimo.

### 3.7.2 Beam Search

La ricerca della soluzione globalmente ottima presenta una sfida significativa, poiché l'enumerazione di ogni possibile percorso (utilizzando un approccio ingenuo) comporterebbe un aumento esponenziale del numero di sequenze da valutare, risultando in un costo computazionale insostenibile.

Per affrontare questo problema, nel presente lavoro è stato implementato l'algoritmo di Beam Search, che rappresenta una via di mezzo tra la ricerca Greedy e l'enumerazione completa dei percorsi.

In Beam Search, vengono mantenuti i migliori  $k$  candidati per ciascuna decisione, dove  $k$  è noto come "**beam width**". Nell'implementazione di questo lavoro, è stata utilizzata una beam width di  $k=5$ , il che significa che durante l'esecuzione dell'algoritmo sono stati considerati e mantenuti i migliori cinque candidati, corrispondenti alle parole con le cinque probabilità più alte per ciascun passaggio.

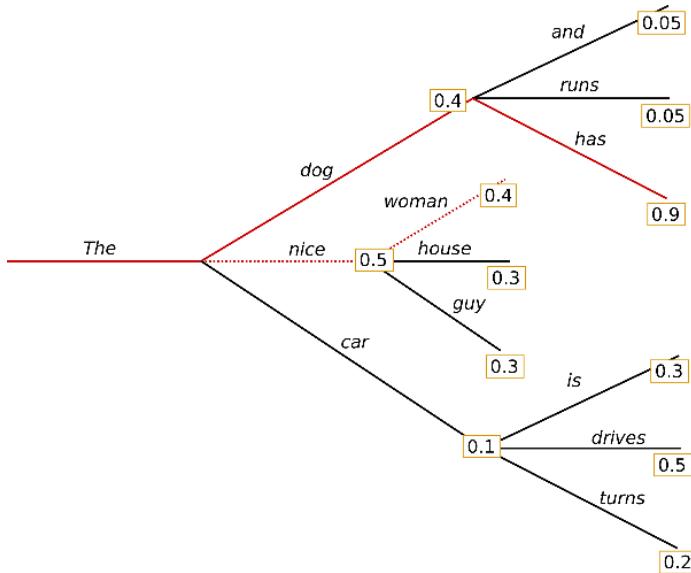


Figura 3.11: Meccanismo di generazione di una caption, usando la tecnica Beam Search (rosso).

Nel caso dell'esempio illustrato, per semplicità si è assunto  $k=2$ .

Al primo passo dell'algoritmo, vengono selezionate due sequenze candidate: ["The", "nice"] e ["The", "dog"], mentre la sequenza ["The", "car"] viene scartata.

Al secondo passo, la Beam Search confronta le sequenze ["The", "dog", "has"] e ["The", "nice", "woman"], calcolando le probabilità totali. La prima sequenza ha una probabilità totale di  $0.4 \cdot 0.9 = 0.36$ , mentre la seconda ha una probabilità totale di  $0.5 \cdot 0.4 = 0.20$ . Di conseguenza, la tecnica della Beam Search individua la sequenza con la probabilità complessiva massima in questo esempio semplificato.

In pratica, durante i calcoli di probabilità nella Beam Search, anziché moltiplicare direttamente le probabilità, si preferisce applicare il logaritmo naturale a ciascuna probabilità ( $\ln(P_i)$  invece di  $P_i$ ). Questo approccio semplifica notevolmente i calcoli per il calcolatore.

$$\max \left( \prod_i P_i \right) \rightarrow \text{difficile}$$

Infatti, trovare il massimo prodotto delle probabilità risulta complesso.

$$\max \left( \ln \left( \prod_i P_i \right) \right) = \max \left( \sum_i \ln(P_i) \right) \rightarrow \text{più facile}$$

Tuttavia, grazie alle proprietà dei logaritmi, il logaritmo del prodotto di una serie di numeri si scomponete nella somma dei logaritmi dei singoli numeri. Questo cambiamento velocizza il tempo di esecuzione dell'algoritmo perché lavorare con sommatorie è molto più agevole rispetto alle produttorie per un calcolatore.

È importante sottolineare che, sebbene Beam Search trovi sempre una sequenza di output con probabilità complessiva superiore rispetto alla Greedy Search, ciò non garantisce necessariamente un output di qualità superiore.

### 3.8 Metriche

Nel vasto panorama delle valutazioni nel campo del Machine Learning, metriche come l'accuracy o la F1-score sono comunemente utilizzate per valutare le prestazioni dei modelli. Tuttavia, quando si tratta di valutare la qualità delle captions generate dai modelli di Image Captioning, è necessario ricorrere a metriche più specifiche. In questo contesto, sono state considerate tre metriche principali nel campo del Natural Language

Processing (NLP): BLEU, CIDEr e ROUGE. Queste metriche forniscono un punteggio che misura quanto le captions generate si avvicinano a quelle di riferimento, offrendo così un'indicazione della qualità delle previsioni del modello. La scelta di utilizzare queste tre metriche deriva dalla loro ampiezza di utilizzo nella comunità scientifica e dalla loro capacità di fornire una valutazione completa e accurata delle prestazioni dei modelli di Image Captioning.

Le metriche attribuiscono un punteggio compreso tra 0 e 1, dove valori più prossimi a 1 indicano una predizione migliore. Non è realistico neanche per un essere umano ottenere un punteggio vicino a 1; se ciò accadesse, potrebbe indicare un potenziale problema di overfitting del modello ai dati di addestramento.

### 3.8.1 Metrica BLEU

La BLEU-score (Bilingual Evaluation Understudy), originariamente sviluppata per valutare traduzioni linguistiche, viene ampiamente impiegata anche nel contesto di Image Captioning.

Per valutare la qualità di una caption generata, si confrontano gli **n-gram** nella caption predetta con quelli presenti nelle caption di riferimento. Gli n-gram rappresentano blocchi di n parole consecutive.

Un approccio per valutare una frase generata è calcolare la "n-gram precision", definita come di seguito:

$$\text{n-gram precision} = \frac{\# \text{ n-gram matches}}{\# \text{ n-grams in generation}}$$

Questo implica il conteggio dei n-grammi presenti sia nella frase generata che in quella di riferimento, e la normalizzazione di questo valore in base al totale dei n-grammi nella frase generata dal modello.

A titolo esplicativo, gli unigrammi della frase "Mi chiamo Tommaso Senatori" sarebbero ["Mi", "chiamo", "Tommaso", "Senatori"], ovvero le singole parole, mentre i bigrammi sarebbero ["Mi chiamo", "chiamo Tommaso", "Tommaso Senatori"].

Di seguito viene presentato un esempio di calcolo della precisione unigramma:

Figura 3.12: **Esempio 1** BLEU: Precisione Unigramma.

**Caption effettiva:** A dog is jumping out of the water.

**Caption generata:** A dog is swimming in the water.

$$\text{Precisione Unigramma} = \frac{\text{numero di parole correttamente predette}}{\text{numero di parole nella caption generata}} = \frac{5}{7}$$

Un inconveniente della precisione unigramma è l'assenza di considerazione dell'ordine delle parole all'interno di una frase. Per superare questa limitazione, vengono impiegati anche la precisione 2-gram, 3-gram e 4-gram.

Si illustra un ultimo esempio in cui la precisione bigramma tiene conto dell'ordine delle parole, assegnando una score più bassa, come previsto.

Figura 3.13: **Esempio 2** BLEU: Confronto tra precisione Unigramma e Bigramma.

**Caption effettiva:** A dog is jumping out of the water.

**Caption generata:** Out the dog a water is of jumping.

$$\text{Precisione unigramma} = \frac{8}{8} = 1.$$

$$\text{Precisione bigramma} = \frac{\text{numero di bigrammi correttamente predetti}}{\text{numero di bigrammi nella caption generata}} = \frac{0}{7} = 0.$$

Nell'ambito di questo studio, è stato deciso di calcolare le score **BLEU-1** e **BLEU-2** utilizzando la funzione `corpus_bleu` fornita dalla libreria `nltk`. Questa funzione calcola le BLEU score per l'intero corpus di testo, ovvero la media delle score per ogni caption generata nel test set.

```
1 bleu1 = corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0))
2 bleu2 = corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0))
```

La BLEU-1 score corrisponde alla precisione unigramma, mentre la BLEU-2 assegna uguale importanza sia ai 1-grammi che ai 2-grammi.

La BLEU score, pur basandosi esclusivamente sulla presenza di n-grammi corrispondenti, fornisce una valutazione numerica pressoché oggettiva della qualità delle frasi generate rispetto alle frasi di riferimento.

### 3.8.2 Metrica CIDEr

La CIDEr (Consensus-based Image Description Evaluation) rappresenta una metrica di valutazione della qualità delle descrizioni di immagini. Va oltre la semplice verifica della correttezza grammaticale e della presenza delle parole corrette, analizzando anche il significato e il contenuto delle descrizioni generate.

Come suggerisce il suo nome, la CIDEr score è basata sul consenso, cioè riflette il punto di vista condiviso dei testi di riferimento. Questo significa che assegna un punteggio più alto alle descrizioni generate che sono simili a più descrizioni di riferimento, rispetto a quelle che corrispondono solo a una singola descrizione.

L'idea alla base è quella di attribuire un peso inferiore agli n-grammi che compaiono frequentemente in tutte le immagini del dataset (**common globally**), poiché è probabile che siano meno informativi. Grazie al tokenizer utilizzato precedentemente, che ha creato un vocabolario ordinando i token in base alla loro frequenza, questo processo risulta più agevole. Parole come "the", "is" e "and" sono considerate meno informative rispetto a parole che compaiono raramente (**rare globally**), come ad esempio "monkey", che è solo la 982<sup>a</sup> parola più comune nel dataset Flickr8k.

In pratica, il consenso è valutato utilizzando la pesatura **TF-IDF** (Term Frequency-Inverse Document Frequency). La TF-IDF è una statistica numerica che riflette l'importanza di un termine  $t$  in un documento  $d$  rispetto a una collezione di documenti  $D$ . La formula per la TF-IDF è la seguente:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Dove:

- $\text{TF}(t, d)$ : Term Frequency per un termine  $t$  in un documento  $d$ , ovvero misura quante volte un termine compare in un documento.
- $\text{IDF}(t, D)$ : Inverse Document Frequency per un termine  $t$  in un corpus di documenti  $D$ , e misura invece quanto sia raro un termine in un insieme di documenti.

Si calcola come:

$$\text{IDF}(t, D) = \log \left( \frac{N}{df_t} \right)$$

con

- $N$ : Numero totale di documenti nel corpus (nel caso del progetto, di descrizioni nel test set).
- $df_t$ : Document Frequency per un termine  $t$ , ovvero quanti sono i documenti che contengono il termine  $t$ .

In sintesi, la CIDEr score rappresenta un’evoluzione rispetto alla BLEU score, premiando non solo la correttezza formale delle descrizioni, ma anche la loro originalità e creatività, offrendo così una valutazione più completa delle capacità espressive del modello di Image Captioning.

### 3.8.3 Metrica ROUGE-L

Rouge (Recall-Oriented Understudy for Gisting Evaluation) è una metrica comunemente usata per valutare la qualità di riassunti di testi, ma trova applicazione anche nel contesto dell’Image Captioning.

Nel contesto del presente lavoro, si è scelto di calcolare la ROUGE-L score, poiché questa metrica non si basa sulla comparazione di n-grammi, ma individua la sottosequenza comune più lunga (**LCS**) tra le descrizioni. Una sottosequenza comune è definita come una sequenza di parole che appare in entrambe le descrizioni nell’ordine specificato, senza necessariamente essere contigua.

La scelta di questa metrica è motivata dal suo approccio diverso rispetto alla BLEU score: non si basa su corrispondenze consecutive di n-grammi, il che consente di catturare in modo più accurato la struttura delle frasi.

Per calcolare Rouge-L si utilizzano i seguenti parametri:

- $\text{Recall} = \frac{\text{LCS}(\text{gen}, \text{rif})}{\# \text{ parole in rif}}$  N.B.
- $\text{Precision} = \frac{\text{LCS}(\text{gen}, \text{rif})}{\# \text{ parole in gen}}$
- $\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- gen: caption generata dal modello
- rif: caption di riferimento

**ROUGE-L recall** valuta la quantità di parole presenti nella caption generata che coincidono con quelle nella caption di riferimento, offrendo così un’indicazione della completezza della descrizione generata.

**ROUGE-L precision**, d'altra parte, valuta la rilevanza della caption generata, misurando quanto della caption prodotta è effettivamente pertinente al contenuto della scena rappresentata.

Infine, la **F1-score**, ottenuta dalla media armonica di precision e recall, fornisce un singolo punteggio che tiene conto sia della completezza che della rilevanza della descrizione generata, offrendo così una valutazione complessiva della qualità della caption rispetto alla caption di riferimento. Quest'ultimo punteggio è stato adottato come metrica di valutazione nel presente lavoro.

## Capitolo 4

# Risultati e Valutazione Sperimentale

Il capitolo dei risultati rappresenta il nucleo centrale di questa ricerca, in cui vengono presentati e analizzati i risultati ottenuti dall'implementazione e dall'esperimento dei diversi approcci proposti per la generazione di didascalie per immagini, con particolare attenzione alle strategie Greedy e Beam Search. Questo capitolo offre una panoramica dettagliata delle performance dei modelli utilizzati sui diversi dataset considerati, includendo valutazioni quantitative attraverso metriche standard come BLEU, CIDEr e ROUGE-L, nonché una disamina qualitativa attraverso esempi di didascalie generate. Attraverso l'analisi dei risultati, sarà possibile trarre conclusioni significative sulle prestazioni dei vari metodi e sull'impatto delle diverse configurazioni sperimentali sui risultati finali.

Sono stati sviluppati complessivamente 5 programmi distinti: 2 dedicati al dataset FLICKR8K, 2 per il dataset SKETCH-SCENE, e 1 specificamente per il dataset personale FLICKR8K SKETCHIFIED.

La diversificazione dei programmi all'interno dello stesso dataset è stata realizzata per valutare e analizzare le differenze nei risultati in base alle variazioni delle caratteristiche implementate.

Per il dataset FLICKR8K, sono stati sviluppati due programmi distinti al fine di esaminare gli effetti della lemmatizzazione e delle diverse architetture di reti neurali utilizzate. Nel programma denominato "**FLICKR8K Standard**", dove non è stata implementata la lemmatizzazione, sono state utilizzate una VGG16 come rete neurale convoluzionale (CNN) e una SimpleRNN come rete neurale ricorrente (RNN). Dall'altro lato, nel programma "**FLICKR8K Advanced**", che include l'implementazione della lemmatizzazione, sono state adottate una Xception come CNN e una LSTM come RNN.

Per quanto riguarda il dataset creato personalmente **FLICKR8K SKETCHIFIED**, è stato deliberatamente mantenuto tutto identico al programma FLICKR8K Advanced, compresi gli iperparametri, le reti neurali utilizzate e altre configurazioni. Ciò consente un'analisi delle differenze che derivano esclusivamente dall'utilizzo di immagini in formato sketch anziché a colori.

Infine, per il dataset SKETCH-SCENE, sono stati creati due programmi che utilizzano la rete neurale convoluzionale Xception e la rete neurale ricorrente LSTM, differenziandosi unicamente per l'adozione o meno della **lemmatizzazione**. Questa scelta è stata appositamente voluta al fine di dimostrare e valutare l'impatto della lemmatizzazione nel contesto di Sketch Captioning.

La tabella 4.1 fornisce una panoramica chiara delle configurazioni adottate per ciascun programma, dalle scelte degli iperparametri all'utilizzo o meno della lemmatizzazione, fino alla selezione delle reti neurali utilizzate per il modello encoder-decoder.

	flickr8k standard	flickr8k advanced	flickr8k SKETCHIFIED	sketch-scene (lemma)	sketch-scene (no lemma)
<b>epochs</b>	5	12	7	6	7
<b>learning_rate</b>	0.001	0.001	0.001	0.001	0.001
<b>batch_size</b>	64	64	64	64	64
<b>CNN</b>	VGG16	Xception	Xception	Xception	Xception
<b>RNN</b>	SimpleRNN	LSTM	LSTM	LSTM	LSTM
<b>Lemmatization</b>	NO	SI	SI	SI	NO

Tabella 4.1: Configurazioni dei 5 programmi distinti sviluppati per Image Captioning.

## 4.1 Grafici di Loss

Nella presente sezione sono analizzati e confrontati i grafici delle funzioni di loss del training e del validation set dei vari programmi, evidenziando come differenti dataset, tecniche e architetture di reti neurali influenzino l'andamento della loss e la riduzione dell'overfitting.

	flickr8k standard	flickr8k advanced	flickr8k SKETCHIFIED	sketch-scene (lemma)	sketch-scene (no lemma)
epochs	5	7	7	6	7
validation_loss	3.9655	3.2552	3.3376	2.767	3.0979

Tabella 4.2: Numero di epoche e loss del validation set per ogni programma.

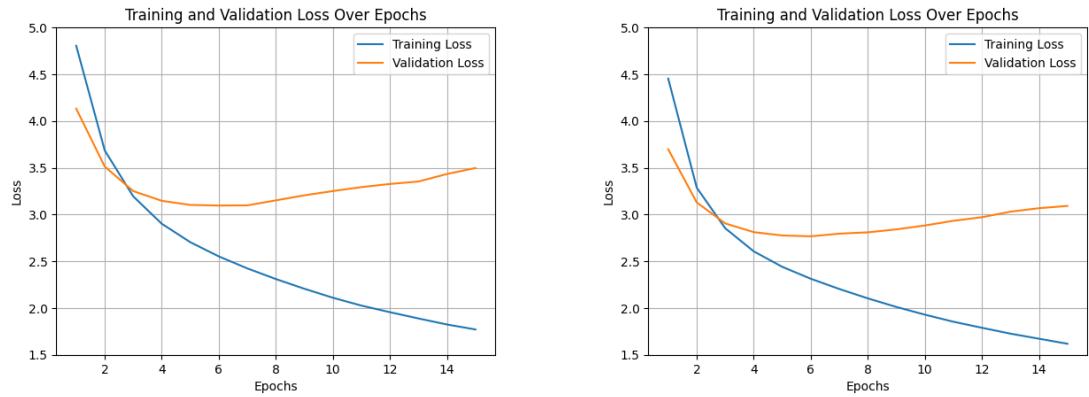
Per ogni programma, sono riportati il numero di epoche selezionate e la loss del validation set. Il numero di epoche è stato determinato al fine di minimizzare la validation loss, evitando l'overfitting. È stato osservato che il numero ottimale di epoche si situa generalmente tra la 5<sup>a</sup> e la 7<sup>a</sup> epoca.

Una prima osservazione preliminare può essere fatta considerando i valori della validation loss all'epoca selezionata.

Si nota che il programma FLICKR8K Advanced presenta una validation loss inferiore rispetto a FLICKR8K Standard.

Inoltre, esaminando i programmi SKETCH-SCENE, si nota che l'implementazione della lemmatizzazione è associata a una validation loss inferiore rispetto alla versione senza lemmatizzazione. Ciò suggerisce un possibile impatto positivo di questa tecnica.

Per poter condurre un'analisi più dettagliata, sono stati realizzati grafici che illustrano l'andamento delle loss del training e del validation set al variare del numero di epoche dei modelli. Oltre a fornire una visualizzazione della performance del modello, questi grafici consentono di individuare il punto in cui il modello inizia a manifestare overfitting, indicato da un significativo divario tra le curve delle funzioni di loss del training e del validation set.

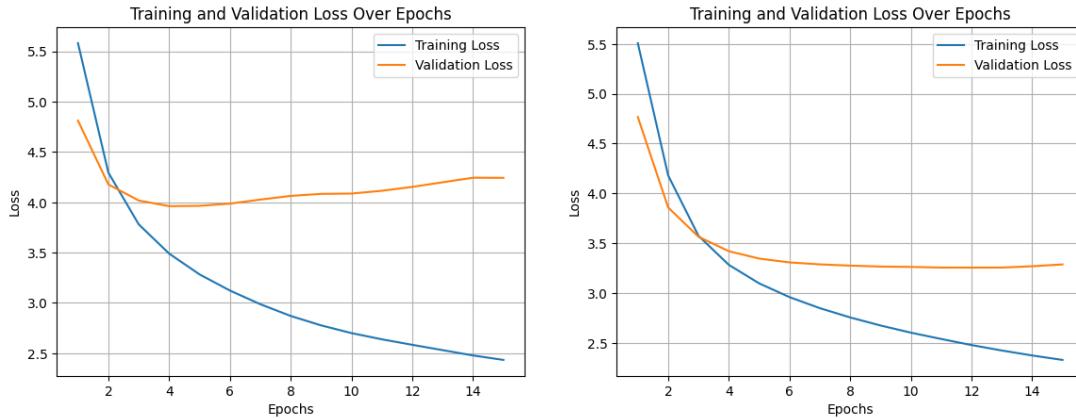


(a) Funzioni di loss in SKETCH-SCENE senza lemmatization.

(b) Funzioni di loss in SKETCH-SCENE con lemmatization.

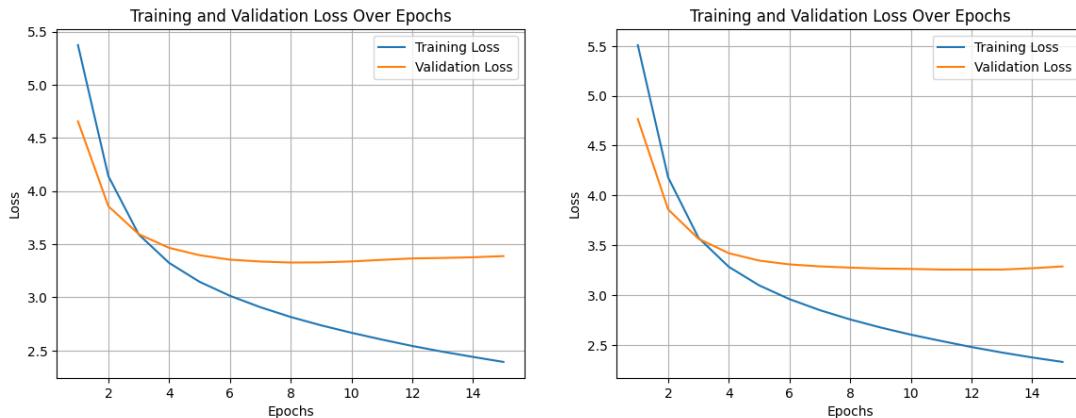
Figura 4.1: Confronto delle funzioni di loss tra i due programmi basati sul dataset SKETCH-SCENE.

L’analisi delle loss durante l’addestramento del dataset SKETCH-SCENE rivela l’impatto dell’uso della lemmatization. In particolare, quest’ultima non aiuta a prevenire l’overfitting, che avviene a prescindere, come evidenziato dall’aumento della validation loss a partire dalla 7° epoca. Tuttavia, si nota che il valore della loss, a parità di epoche, risulta inferiore. L’andamento delle funzioni di loss con l’aggiunta della lemmatization rimane invariato, ma si osserva una traslazione verso il basso lungo l’asse y. Questo fenomeno può essere spiegato dal fatto che, come detto in precedenza, la semplificazione dei termini alla loro forma base ha determinato una riduzione del 10% della dimensione del vocabolario. Ciò ha comportato una diminuzione delle opzioni disponibili per il modello durante il training e, di conseguenza, una ridotta probabilità di errore.



(a) Funzioni di loss per il programma FLICKR8K standard. (b) Funzioni di loss per il programma FLICKR8K advanced.

Figura 4.2: Confronto delle funzioni di loss tra i due programmi basati sul dataset FLICKR8K: "FLICKR8K standard" (a sinistra) e "FLICKR8K advanced" (a destra).



(a) Funzioni di loss per il dataset FLICKR8K SKETCHIFIED. (b) Funzioni di loss per il programma FLICKR8K advanced.

Figura 4.3: Confronto delle funzioni di loss tra il programma basato sul dataset FLICKR8K SKETCHIFIED (a sinistra) e il programma basato su FLICKR8K "FLICKR8K advanced" (a destra).

Nell'analisi dei grafici relativi al dataset FLICKR8K (Figura 4.2) emerge un notevole cambiamento, attribuibile sia all'introduzione della lemmatization, sia soprattutto all'adozione di reti neurali più efficienti.

Non solo si osserva un abbassamento delle curve dei grafici di circa 0.7, ma viene del tutto risolto il problema dell'overfitting. Nel programma "FLICKR8K Standard",

la validation loss aumenta dopo poche epoche, mentre la training loss continua a diminuire, creando un divario sempre più ampio tra le due. Al contrario, nel programma "FLICKR8K Advanced", anche con un numero relativamente elevato di epoche, la validation loss raggiunge un plateau, senza mai aumentare, mostrando un andamento a derivata sempre minore o uguale a 0.

Infine, viene mostrato in figura 4.3 il grafico relativo al dataset "FLICKR8K SKETCHIFIED", che attualmente non presenta differenze rispetto al grafico "FLICKR8K advanced".

Una nota finale è che, utilizzando le stesse tecniche e modelli di addestramento, questo dataset personalizzato di sketch non presenta overfitting, a differenza del dataset "sketch-scene".

## 4.2 Metriche

In questa sezione, sono presentati i risultati delle metriche BLEU-1, BLEU-2, CIDEr e ROUGE-L per ciascun programma considerato. Queste metriche, introdotte in modo generale nel capitolo precedente, forniscono una valutazione della qualità delle descrizioni generate dal modello.

Poiché sono stati sviluppati due approcci differenti per la generazione delle didascalie delle immagini, il metodo Greedy e il Beam Search, entrambi sono stati utilizzati per valutare la qualità delle immagini, consentendo così un confronto numerico diretto tra i due approcci.

Di seguito è riportata la tabella completa dei risultati:

	flickr8k standard	flickr8k advanced	flickr8k SKETCHIFIED	sketch-scene (lemma)	sketch-scene (no lemma)
epochs	5	12	7	6	7
learning_rate	0.001	0.001	0.001	0.001	0.001
batch_size	64	64	64	64	64
CNN	VGG16	Xception	Xception	Xception	Xception
RNN	SimpleRNN	LSTM	LSTM	LSTM	LSTM
Lemmatization	NO	SI	SI	SI	NO
validation loss	3.9655	3.2552	3.3376	2.767	3.0979
BLEU greedy	0.568451, 0.347878	0.593739, 0.415030	0.637837, 0.435486	0.445831, 0.247458	0.429682, 0.211213
BLEU beam search	-	0.623245, 0.443671	0.641945, 0.443784	0.441479, 0.246538	-
CIDEr greedy	0.364335	0.404985	0.344474	0.643714	0.618959
CIDEr beam search	-	0.481412	0.401337	0.68253	-
ROUGE-L greedy	0.368971	0.413368	0.412261	0.453774	0.433981
ROUGE-L beam search	-	0.421676	0.417325	0.45848	-

Tabella 4.3: Configurazioni e risultati dei 5 programmi distinti sviluppati per Image Captioning.

#### 4.2.1 BLEU-1 e BLEU-2

Si comincia ad osservare i programmi con il dataset flickr8k.

Una prima considerazione è che le BLEU score per il programma “FLICKR8K advanced” risultano notevolmente superiori rispetto a quelle del programma “FLICKR8K standard”. Questo può essere attribuito all’impiego di reti neurali più avanzate, integrate con il processo di lemmatizzazione, il quale ha influito positivamente sui risultati ottenuti.

Un altro risultato molto positivo si osserva nella colonna di “FLICKR8K advanced”: l’adozione del metodo di generazione delle captions Beam Search ha portato a un ulteriore aumento delle BLEU score rispetto al metodo Greedy, con un incremento del 3%. Tale miglioramento è stato evidenziato sia nel dataset FLICKR8K che in quello Sketchified, sebbene in quest’ultimo il vantaggio sia stato meno marcato, inferiore all’1%.

Un risultato eclatante emerso dall’analisi è che, indipendentemente dal metodo di generazione delle didascalie utilizzato, il dataset FLICKR8K Sketchified (creato come iniziativa personale) ha consistentemente registrato score BLEU superiori rispetto al dataset FLICKR8K. In particolare, le BLEU score ottenute con il metodo Beam Search, pari rispettivamente a 0.64 e 0.44, rappresentano i punteggi più alti raggiunti nell’ambito di questo intero lavoro.

Si ricorda che le differenze tra questi due dataset risiedono esclusivamente nella trasformazione delle immagini in sketch e nella rimozione dei token relativi ai colori dal vocabolario delle captions.

Una possibile spiegazione per i punteggi BLEU più elevati osservati nel dataset di sketch rispetto a quello originale di Flickr8k potrebbe risiedere nel fatto che la rimozione dei token relativi ai colori abbia semplificato il compito di generazione delle didascalie per il modello. Questa semplificazione potrebbe aver portato a un miglioramento delle prestazioni complessive del modello, poiché il task di generazione delle didascalie sarebbe diventato meno complesso e maggiormente orientato alla comprensione del contenuto dell'immagine piuttosto che ai dettagli visivi. In altre parole, eliminando la necessità di prevedere i colori, il modello potrebbe aver avuto una maggiore precisione nel catturare il significato delle immagini, riducendo così il margine di errore.

Si esamina ora l'altro dataset di sketch, denominato sketch-scene, concentrandosi sulle relative BLEU score.

Comparando i due programmi creati utilizzando questo dataset, che differiscono solo per l'impiego della lemmatizzazione, emerge un miglioramento netto nelle BLEU score, attribuibile all'utilizzo di questa tecnica.

Un risultato inatteso si osserva nel programma sketch-scene con lemmatizzazione. Nonostante le aspettative, l'adozione del metodo Beam Search non sembra influenzare in modo significativo le BLEU score, anzi si registra un leggero decremento rispetto all'impiego del metodo greedy.

Come già evidenziato nella sezione dedicata, sebbene Beam Search tenda a produrre sequenze di output con una probabilità complessiva maggiore rispetto alla Greedy Search, ciò non garantisce necessariamente una qualità superiore dell'output, come dimostrato da questa osservazione.

Poiché il dataset contiene solo una caption di riferimento per immagine, in contrasto con le cinque presenti nel dataset FLICKR8K, e considerando che le caption stesse potrebbero contenere imprecisioni linguistiche, entrambi i metodi potrebbero avere difficoltà nel produrre descrizioni migliorative o più accurate rispetto all'altra.

Infine, una considerazione generale rivela che le BLEU score del dataset sketch-scene sono di gran lunga inferiori rispetto a flickr8k.

Questo risultato è in linea con le aspettative, considerando che sketch-scene è un dataset con qualità oggettivamente inferiore, caratterizzato da caption contenenti errori grammaticali e immagini poco chiare. Tuttavia, nonostante queste avversità, è stato comunque raggiunto un punteggio più che soddisfacente.

#### 4.2.2 CIDEr

Per quanto riguarda i risultati della CIDEr score, si nota un significativo impatto dell'uso della Beam Search, con un drastico aumento del punteggio in ogni contesto.

Quello che emerge però da altre osservazioni è che, contro ogni aspettativa, CIDEr offre una visione diametralmente opposta rispetto a BLEU.

Per esempio, si osserva che il punteggio ottenuto dal programma "Flickr8k advanced" è superiore sia rispetto a quello di "Flickr8k standard", come previsto, sia rispetto al dataset Flickr8k di sketch. Sorprendentemente, il punteggio per quest'ultimo è perfino inferiore al programma di flickr8k senza lemmatizzazione e con uso di reti neurali meno avanzate.

Tuttavia, l'osservazione in assoluto più inaspettata risiede nel punteggio registrato dal dataset "SKETCH-SCENE", il quale supera nettamente quello di "FLICKR8K", raggiungendo un massimo di 0.68, un risultato contrastante con le aspettative date dalla metrica BLEU.

Questo fenomeno potrebbe essere spiegato dalla minore varietà di immagini presenti nel dataset SKETCH-SCENE, che potrebbe facilitare la predizione delle descrizioni, poiché i modelli tendono a generare frasi più conservative e ripetitive quando sono esposti a un insieme limitato di contesti visivi.

#### 4.2.3 ROUGE-L

L'analisi finale riguarda i risultati della ROUGE score, in particolare della ROUGE-L, che rivela interessanti discrepanze rispetto alle metriche precedentemente esaminate.

Anche in questo caso, l'impatto della Beam Search rispetto alla Greedy Search è evidente, sebbene l'aumento sia marginale (inferiore all'1%), indipendentemente dal dataset utilizzato. La lemmatizzazione conferma il suo impatto positivo, aumentando la ROUGE-L score in tutte le analisi.

Nonostante la trasformazione delle immagini di FLICKR8K in sketch abbia influenzato positivamente la BLEU score e negativamente la CIDEr, la ROUGE-L score non registra particolari differenze.

Sorprendentemente, anche questa metrica evidenzia un vantaggio a favore del dataset di qualità inferiore "sketch-scene" rispetto a flickr8k, sebbene il divario non sia così marcato come nella CIDEr.

### 4.3 Esempi di Immagini

In questa sezione finale del capitolo, sono presentati esempi rappresentativi di immagini presenti nei dataset, analizzando le descrizioni generate dai modelli.

A tale scopo, è stata creata una funzione chiamata “**generate\_caption**”, che riceve il nome di un'immagine come input e stampa l'immagine stessa insieme alle caption di riferimento e alle due caption generate dal modello: una utilizzando il metodo Greedy e l'altra utilizzando il metodo Beam Search.

Per ciascun programma, sono mostrati esempi di immagini provenienti sia dal training set che dal test set. Tuttavia, nell'ambito della Tesi, sono incluse solo le immagini prese dal test set. Questo approccio è motivato dal fatto che il modello tende a riconoscere più facilmente le immagini del set di addestramento, poiché è stato appunto addestrato su di esse. Utilizzare solo le immagini del test set consente quindi di valutare in modo più accurato le capacità di generalizzazione del modello, poiché deve generare descrizioni per immagini che non ha mai incontrato durante l'addestramento.

### 4.3.1 Immagini del dataset sketch-scene

#### 4.3.1.1 Esempi di Immagini con Didascalie Corrette

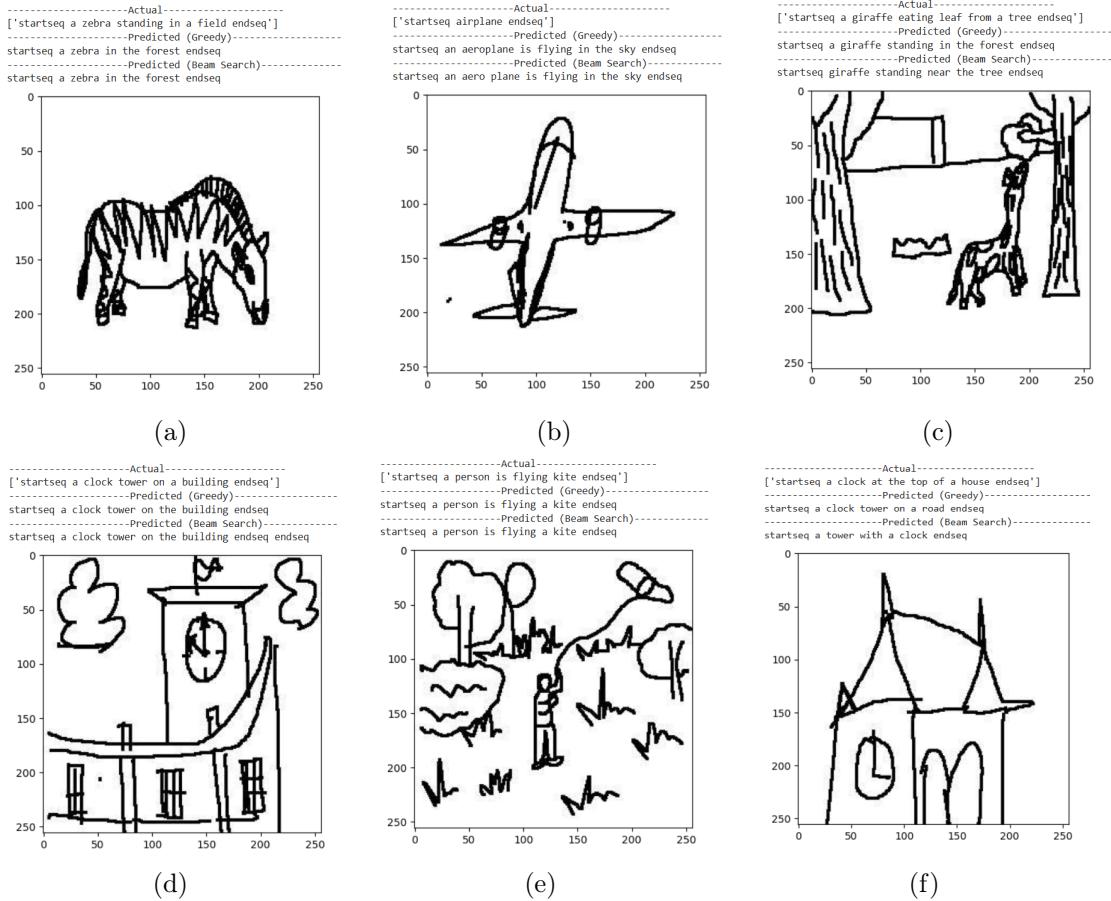


Figura 4.4

Negli esempi precedenti, le captions generate dal modello catturano l'essenza delle immagini. Entrambi i metodi di generazione mostrano coerenza con le descrizioni di riferimento, confermando il successo del modello nel compito di Sketch Captioning.

### 4.3.1.2 Esempi di Performance Superiore di Beam Search rispetto a Greedy

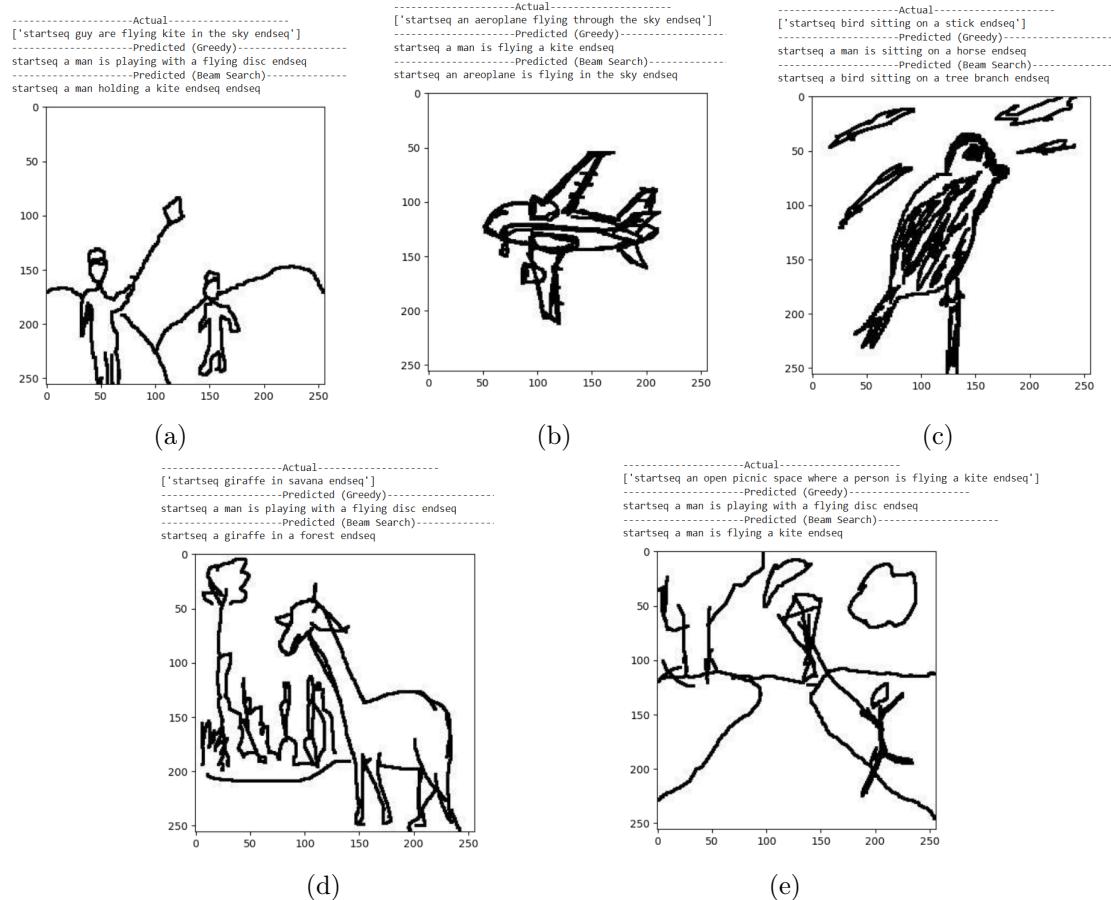


Figura 4.5

In particolare, mentre Beam Search è riuscito a fornire descrizioni più precise e contestualmente coerenti con le immagini, il metodo Greedy ha mostrato una tendenza a produrre descrizioni che risultavano poco o per niente correlate con il contenuto delle immagini stesse.

### 4.3.1.3 Esempi di Performance Superiore di Greedy rispetto a Beam Search

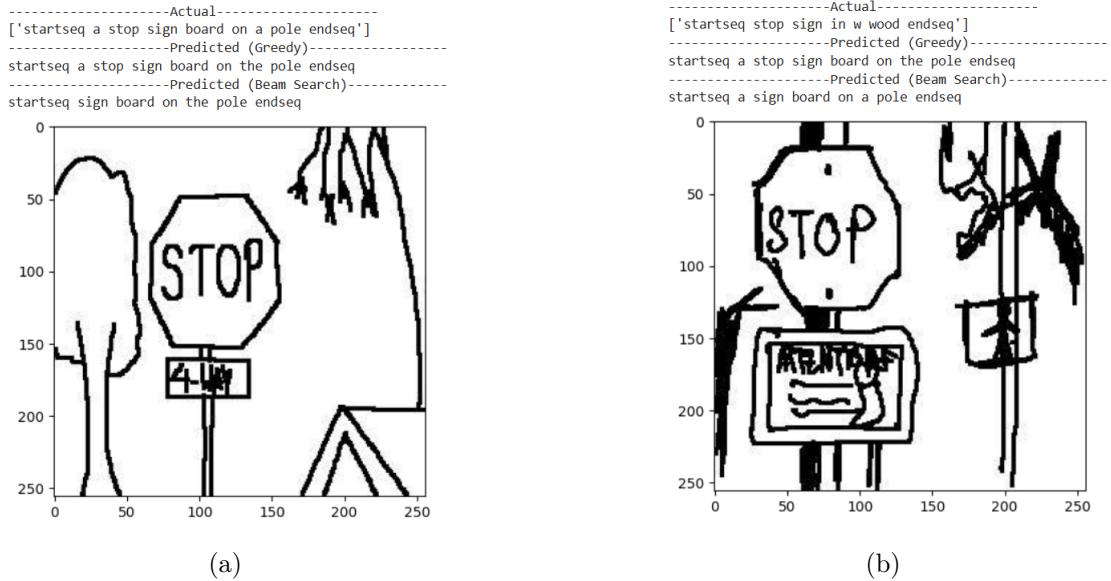


Figura 4.6

Vi sono circostanze specifiche in cui il metodo Greedy si è dimostrato più efficace rispetto al metodo Beam Search. Ad esempio, nel caso di un cartello stradale con la scritta "STOP", il metodo Greedy è in grado di specificare "STOP SIGN BOARD", mentre Beam Search genera semplicemente "SIGN BOARD". È importante notare che, sebbene Beam Search tenda a essere meno dettagliato, riesce comunque a catturare l'essenza dell'immagine.

#### 4.3.1.4 Errori Comuni

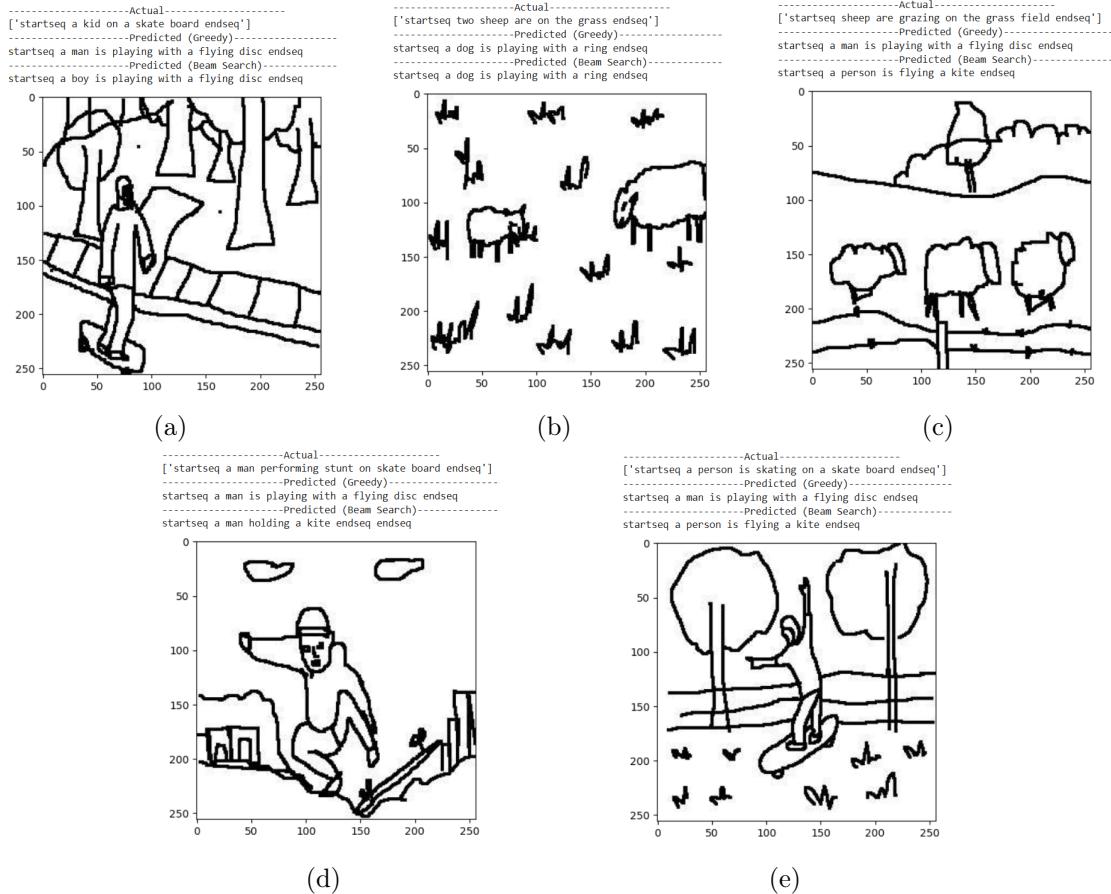


Figura 4.7

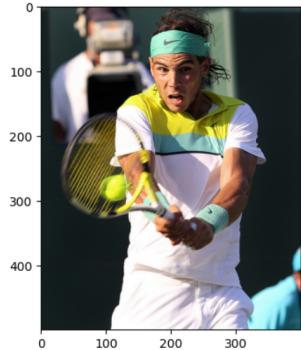
Analizzando le captions errate, emerge un trend ricorrente di errori riguardanti soggetti come gli skateboard o il gregge di pecore sull'erba. È interessante notare che quando il modello fallisce o non è sicuro della previsione, tende spesso a generare descrizioni simili, come "a man is playing with a flying disc". Questo potrebbe suggerire una sorta di fallback per il modello quando non è in grado di determinare correttamente il soggetto dell'immagine.

### 4.3.2 Immagini dei dataset Flickr8k e Flickr8k Sketchified

#### 4.3.2.1 Esempi di Immagini con Didascalie Corrette

Di seguito sono illustrati esempi di immagini correttamente predette dai modelli. Ogni esempio include un'immagine dal dataset FLICKR8K insieme alla sua controparte sketchificata dal dataset FLICKR8K sketchified.

-----Actual-----  
 startseq a person holding a tennis racket hit a yellow tennis ball endseq  
 startseq a tennis player hitting the ball endseq  
 startseq a tennis player hitting the ball endseq  
 startseq the tennis ball is about to be hit endseq  
 startseq this man is hitting a tennis ball with a tennis racket endseq  
 -----Predicted (Greedy)-----  
 startseq a man in a white shirt hit a tennis ball endseq  
 -----Predicted (Beam Search)-----  
 startseq a man hit a tennis ball endseq



(a)

-----Actual-----  
 startseq a black dog playing with a purple toy in the snow endsq  
 startseq a black dog run through the snow carrying a blue toy endseq  
 startseq a dog play in the snow endseq  
 startseq dog running with a purple toy in the snowy field endseq  
 startseq the black and brown dog carry a purple toy in the snow endseq  
 -----Predicted (Greedy)-----  
 startseq a black dog is running through the snow endseq  
 -----Predicted (Beam Search)-----  
 startseq a black dog is running through the snow endseq



(c)

-----Actual-----  
 startseq a black dog jumping into a lake endseq  
 startseq a black dog with a red collar is jumping in the water endseq  
 startseq a black dog with a red collar is jumping out of the water endseq  
 startseq black dog with red collar splashing in water endseq  
 startseq the black dog with a red collar is jumping through the water endseq  
 -----Predicted (Greedy)-----  
 startseq a black dog is running through the water endseq  
 -----Predicted (Beam Search)-----  
 startseq a black dog is running through the water endseq



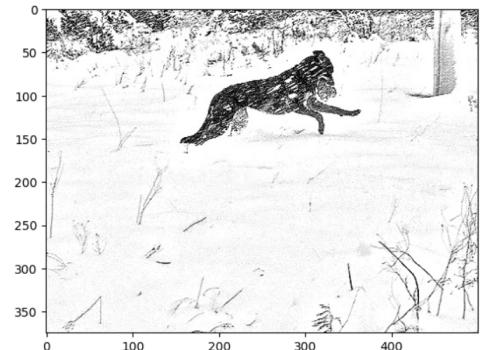
(e)

-----Actual-----  
 startseq a person holding a tennis racket hit a tennis ball endseq  
 startseq a tennis player hitting the ball endseq  
 startseq a tennis player hitting the ball endseq  
 startseq the tennis ball is about to be hit endseq  
 startseq this man is hitting a tennis ball with a tennis racket endseq  
 -----Predicted (Greedy)-----  
 startseq a tennis player in a shirt is hitting a tennis ball endseq  
 -----Predicted (Beam Search)-----  
 startseq a tennis player hitting a tennis ball endseq



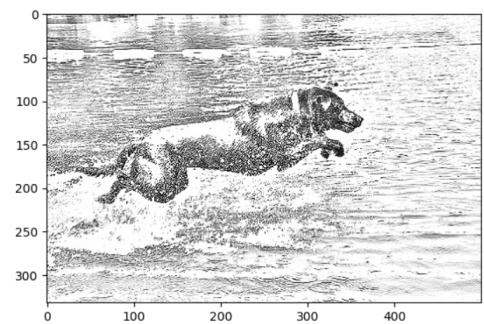
(b)

-----Actual-----  
 startseq a dog playing with a toy in the snow endsq  
 startseq a dog run through the snow carrying a toy endseq  
 startseq a dog play in the snow endseq  
 startseq dog running with a toy in the snowy field endseq  
 startseq the and dog carry a toy in the snow endseq  
 -----Predicted (Greedy)-----  
 startseq a dog running through the snow endseq  
 -----Predicted (Beam Search)-----  
 startseq a dog running in the snow endseq



(d)

-----Actual-----  
 startseq a dog jumping into a lake endseq  
 startseq a dog with a collar is jumping in the water endseq  
 startseq a dog with a collar is jumping out of the water endseq  
 startseq dog with collar splashing in water endseq  
 startseq the dog with a collar is jumping through the water endseq  
 -----Predicted (Greedy)-----  
 startseq a dog is running through water endseq  
 -----Predicted (Beam Search)-----  
 startseq a dog swimming in water endseq



(f)

Figura 4.8

Entrambi i metodi di predizione catturano l'essenza dell'immagine molto bene, ma un problema frequente riscontrato nel dataset FLICKR8K è il riconoscimento accurato dei colori nelle immagini (esempi qui di seguito). Questo problema, ovviamente, non si presenta nel dataset FLICKR8K sketchified. Questa differenza può spiegare il motivo per cui alcune metriche mostrano un punteggio leggermente più alto nel dataset FLICKR8K sketchified rispetto a FLICKR8K.

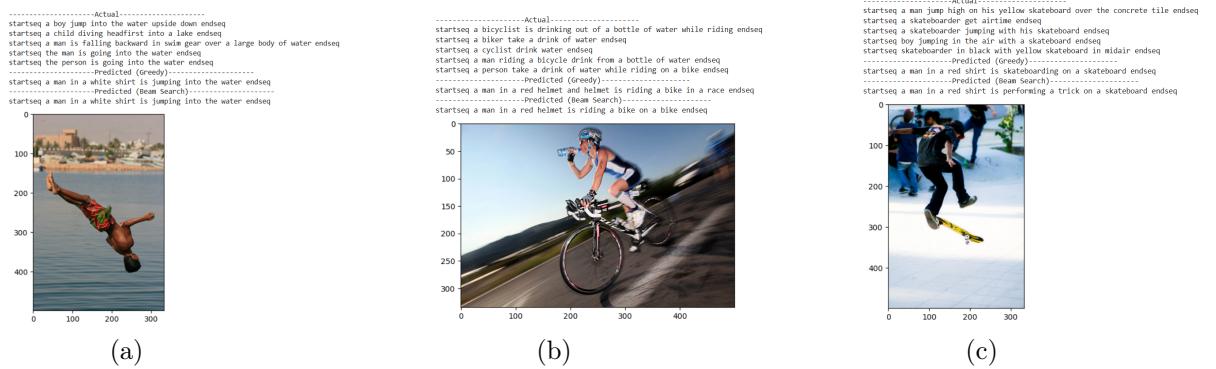


Figura 4.9

#### 4.3.2.2 Esempi di Risultati Variabili tra Greedy e Beam Search

Si mostrano anche immagini che sono state riconosciute meglio con una tecnica che con un'altra.

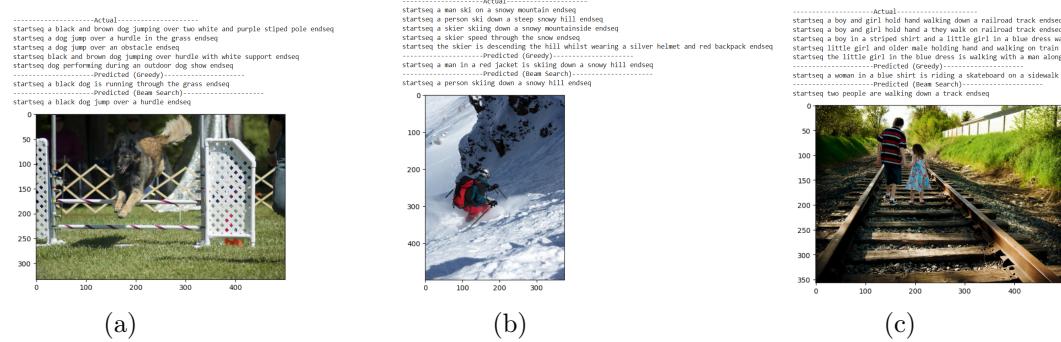


Figura 4.10

### 4.3.2.3 Ridondanza

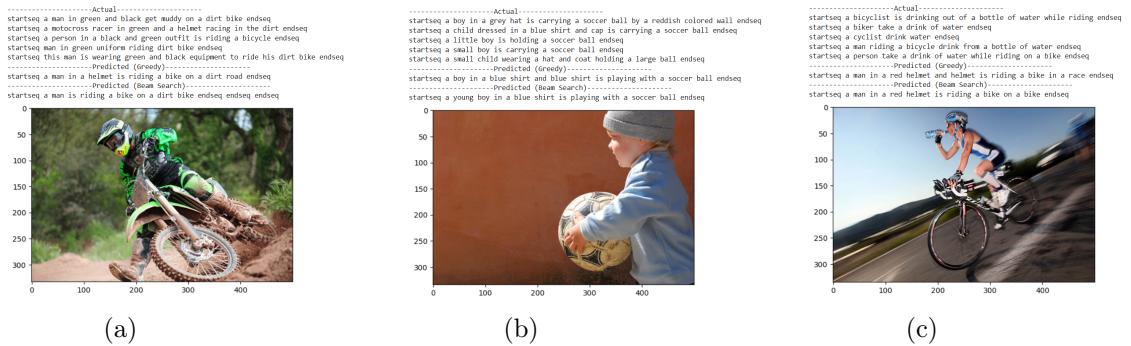


Figura 4.11

In alcuni casi specifici, si sono riscontrati esempi di ridondanza nelle caption generate, come “bike on a bike” o “helmet and helmet”. Nonostante queste occasionali ripetizioni, il contenuto delle immagini è solitamente catturato correttamente.

Concentrandosi sul programma di Flickr8k standard, implementato utilizzando una SimpleRNN anziché una LSTM, si osserva un esempio tangibile del Vanishing Gradient Problem. Questo fenomeno si verifica quando la rete neurale affronta sequenze di input lunghe, causando la perdita di memoria a lungo termine e compromettendo la corretta memorizzazione delle informazioni precedenti. Di conseguenza, il modello potrebbe non essere in grado di richiamare adeguatamente le parole o le caratteristiche delle sequenze precedenti, portando a una situazione in cui si ripetono ciclicamente le stesse previsioni senza progredire nella generazione della sequenza corretta. Nell'esempio presentato in figura 4.12, si evidenzia come il modello generi ripetutamente la stessa sequenza fino a quando non raggiunge la massima lunghezza consentita (`max_length`), momento in cui si blocca senza mai predire il token "endseq".

-----Actual-----

startseq man in crowd wears red hat and red bandanna over his face endseq  
startseq person with dreadlocks and red hat with their face covered by red bandanna endseq  
startseq woman wearing hat and red scarf stands among others wearing similar clothes endseq  
startseq woman wearing red hat and red bandanna followed by man in red bandanna endseq  
startseq the lady is wearing bandanna around her face to protect her from dust endseq

### Predicted

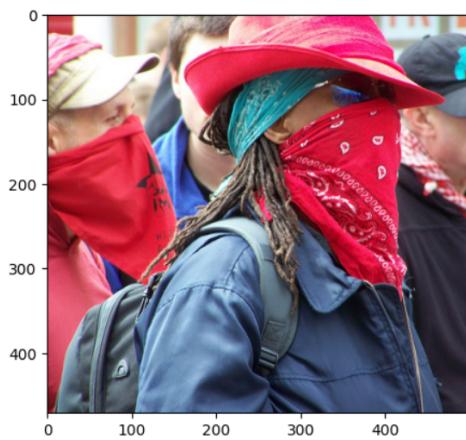


Figura 4.12

#### 4.3.2.4 Altre immagini di Flickr8k Sketchified

Per ultime, si mostrano altre immagini del dataset FLICKR8K SKETCHIFIED.

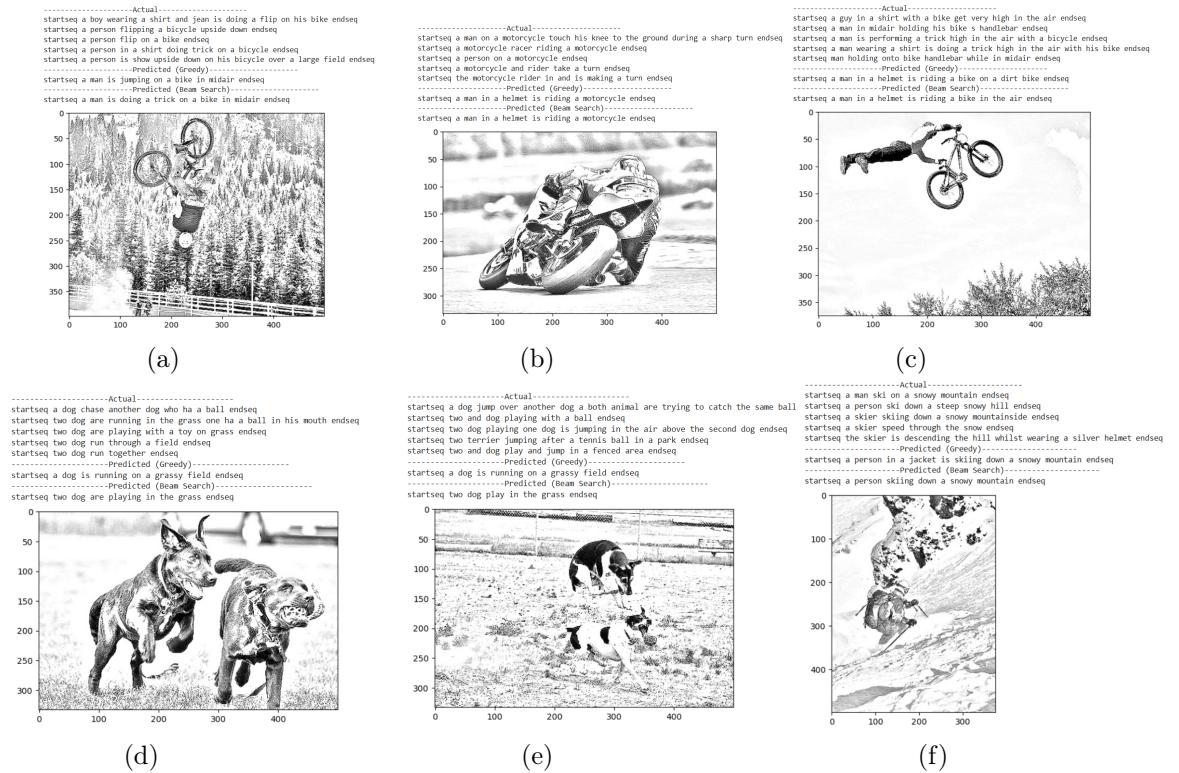


Figura 4.13

### 4.3.3 Confronto tra i Dataset

Come confronto conclusivo tra i dataset, in FLICKR8K, anche quando un metodo di generazione delle descrizioni prevale sull'altro, entrambi risultano abbastanza accurati. Ad esempio, le descrizioni generate sia dal programma "FLICKR8K standard" che dal "FLICKR8K advanced" hanno mostrato un alto livello di accuratezza, pur con lievi differenze di performance. Al contrario, nel caso di SKETCH-SCENE, si osserva occasionalmente una disparità tra le descrizioni generate dai due metodi, particolarmente evidente nei casi in cui le immagini contengono scene complesse o dettagliate. Ad esempio, in alcune immagini di sketch, le descrizioni generate senza l'uso della lemmatization risultano meno precise e più generiche rispetto a quelle generate con l'uso della lemmatization, che riescono a catturare meglio i dettagli specifici delle scene rappresentate. Queste osservazioni evidenziano come la complessità e la natura del dataset possano influenzare significativamente le performance dei modelli di image captioning.

# Conclusioni e sviluppi futuri

## Conclusioni

Il lavoro svolto in questa tesi ha portato allo sviluppo di un modello Encoder-Decoder combinato con CNN e RNN, in grado di generare descrizioni testuali coerenti e adeguate sia per immagini che per sketch. I risultati ottenuti hanno dimostrato l'efficacia di questo approccio, evidenziando la capacità del modello di comprendere e descrivere contesti visivi con una notevole accuratezza.

In particolare, è stata fornita un'approfondita analisi focalizzandosi sui dataset FLICKR8K, FLICKR8K Sketchified e sketch-scene. Attraverso l'implementazione di diverse strategie di generazione del testo (Beam Search e Greedy Search) e l'uso di metriche di valutazione quali BLEU, CIDEr e ROUGE-L, sono emersi risultati rilevanti.

Nel confronto tra le diverse metodologie di generazione del testo, è stato osservato che la Beam Search tende generalmente a produrre descrizioni più accurate e contestualmente coerenti con le immagini rispetto alla Greedy Search. L'analisi delle metriche ha rivelato che la lemmatizzazione ha un impatto positivo sulle prestazioni dei modelli, migliorando i punteggi delle metriche in tutte le condizioni esaminate. Inoltre, è emerso che il dataset sketch-scene, nonostante la sua qualità inferiore rispetto agli altri due, ha prodotto risultati sorprendentemente competitivi in alcune metriche di valutazione, dimostrando che i modelli possono essere addestrati con successo su dataset eterogenei.

Tuttavia, sono state riscontrate diverse limitazioni e sfide nel corso del progetto. La generazione di descrizioni per sketch, che contengono meno dettagli visivi rispetto alle

immagini, ha rappresentato una sfida significativa. Inoltre, la scarsità di dataset open source disponibili per lo sketch captioning ha richiesto la creazione del dataset Flickr8k Sketchified, partendo da Flickr8k originale. Questa mancanza di dati ha limitato le possibilità di addestramento del modello, sottolineando la necessità di una maggiore disponibilità di dataset specifici per lo sketch captioning.

## Sviluppi futuri

Per futuri sviluppi, vi sono diverse direzioni promettenti. Un approccio potenzialmente fruttuoso sarebbe l'implementazione di architetture Transformer con meccanismi di attenzione, che potrebbero migliorare considerevolmente la qualità delle descrizioni generate. Inoltre, sperimentare con diverse tecniche di pre-elaborazione delle immagini e degli sketch potrebbe ottimizzare ulteriormente l'estrazione delle caratteristiche visive. Ulteriori metriche di valutazione, oltre a quelle già utilizzate, potrebbero fornire una comprensione più completa delle performance del modello. Infine, l'utilizzo di dataset più grandi e diversificati, come Flickr30k o MS COCO, potrebbe dare un enorme boost al modello, grazie alla maggiore quantità e varietà di immagini disponibili.

Estensioni del lavoro di ricerca potrebbero includere l'applicazione del modello ad altri domini, come il video captioning, offrendo così nuove opportunità per l'uso pratico di combinazioni di tecniche di computer vision e NLP. Esempi di applicazioni potrebbero includere chatbot che forniscono feedback in base a un'immagine, o sistemi di supporto per non vedenti, che descrivono le scene visive in tempo reale per aiutare nella comprensione dell'ambiente circostante.

# Bibliografia

- [1] M. Azure, “What is computer vision?,” 2024.
- [2] M. Elgendi, *Deep Learning for Vision Systems*. Manning Publications Co., 2020.
- [3] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.
- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [5] “Corso di deep learning,” 2023-2024.
- [6] M. Hodosh, P. Young, K. Li, and J. Hockenmaier, “Flickr8k text dataset,” 2013.
- [7] Zoheb, “zoheb/sketch-scene dataset,” 2024.
- [8] P. N. Chowdhury, A. Sain, A. K. Bhunia, T. Xiang, Y. Gryaditskaya, and Y.-Z. Song, “Fs-coco: Towards understanding of freehand sketches of common objects in context.,” in *ECCV*, 2022.
- [9] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., second ed., 2019.
- [10] H. Face, “How to generate text: using different decoding methods for language generation with transformers,” 2023.