University of Florence
Engineering School of Florence
Department of Information Engineering

What Do Vision-Language Models Understand? Dissecting CLIP Model from Baseline Supervised Classification to Class-Agnostic Similarity Features

Authors:
Niccolò Benedetto

Supervised by:
Prof. Andrew David Baghdanov

Academic Year: 2024-2025

Dedication

.

A voi, mia famiglia, che mai avete vacillato nella fede che riponevate in me, che con sguardi silenziosi e sorrisi discreti mi avete sempre sorretto: il vostro supporto è stato, e sarà, la mia roccia taciturna.

Ma oggi, in questa pagina che segna la fine di un lungo capitolo, rivolgo il mio più sentito ringraziamento a me stesso.
A me, che ho scelto di intraprendere questo cammino impervio, accettando la fatica come unica alleata.
A me, che tra svariati fallimenti e qualche incertezza, mai ho smesso di credere nella forza della mia testa, nella fermezza di ogni mio passo e nel valore di ciascun sacrificio.
A me, che ho costruito pietra dopo pietra, con pazienza e ostinazione, ciò che oggi posso chiamare conquista.

Questa tesi non è soltanto la prova tangibile del sapere acquisito, tuttavia è anche il simbolo di un cammino compiuto con dignità e coraggio. È il punto culminante di una fase e, al contempo, l'alba di una nuova partenza, più ambiziosa e consapevole. Che ogni fatica fin qui sostenuta sia il fondamento su cui poggiare il futuro, e che ogni sogno diventi progetto, ogni progetto, realtà.

A chi crede nel valore del lavoro silenzioso, dell'impegno incrollabile e della volontà che non chiede aiuto ma cerca solo se stessa: questa pagina è per voi.
Ma, prima di tutti, è per me.

Nonno, ce l'ho fatta. So che hai visto tutto, e che oggi sorridi con me.

Statement

Si dichiara che la versione originale di questa tesi è stata scritta in lingua italiana.

È stato fatto uso del modello chatbot generativo ChatGPT come aiuto nella generazione e revisione di alcune porzioni di codice e nella produzione dei set di prompts utilizzati durante gli esperimenti.

Acknowledgments

Vorrei esprimere la mia sincera gratitudine al professore Andrew David Bagdanov, la cui guida e competenza sono state determinanti nel corso di questa tesi. Sono anche grato al Dipartimento di Informatica di Firenze per le risorse didattiche e il supporto.

Ringrazio di cuore la mia famiglia per il costante incoraggiamento e la comprensione. Infine, cito le comunità e le piattaforme open-source come Hugging Face e OpenAI, per aver reso modelli all'avanguardia come CLIP accessibili per la ricerca e la sperimentazione.

Abstract

I modelli Contrastive Language-Image Pre-training (CLIP) hanno mostrato un notevole successo nell'allineare semantiche visive e linguistiche attraverso l'apprendimento contrastivo. Tuttavia, capire cosa codificano realmente questi modelli riguardo alle categorie visive - soprattutto sotto una supervisione limitata - rimane una questione aperta.

Questa tesi indaga la qualità rappresentativa delle embeddings prodotte da CLIP attraverso una serie di esperimenti di classificazione su dataset standard come CIFAR-10 e CIFAR-100. Inizieremo con una pipeline di classificazione delle immagini baseline, utilizzando embeddings di CLIP e modelli lineari supervisionati. Valuteremo quindi CLIP in un contesto zero-shot utilizzando prompts testuali per esplorarne le prestazioni senza alcun training.

Il cuore del nostro lavoro esplora le rappresentazioni di somiglianza class-agnostic: invece di utilizzare nomi di classi, generiamo prompts generici per sondare la capacità di CLIP di separare categorie visive basate esclusivamente sulla somiglianza intermodale. Per valutare e migliorare questo approccio, identificheremo e mitigheremo problemi chiave come la sparsità delle logits, la bassa risoluzione delle immagini e la limitata espressività del vocabolario dei prompts. Sperimenteremo il ridimensionamento della temperatura, prompts definiti da un vocabolario più ampio e dataset ad alta risoluzione per sfruttare meglio le funzionalità preaddestrate di CLIP.

I nostri risultati forniscono nuove informazioni su come e cosa CLIP comprende dalle immagini, evidenziando i punti di forza e i limiti del modello nella codifica delle caratteristiche discriminative di classe anche senza alcun perfezionamento. Questa analisi potrebbe contribuire a una comprensione più profonda dell'efficacia e dei limiti dei modelli visivi-linguistici circa i vari tasks di classificazione sotto vari livelli di supervisione.

Contents

List of Figures

List of Tables

General introduction

0.1 Background and Context

Nel contesto dei modelli di apprendimento multimodale, nasce l'emergenza di un apprendimento congiunto da dati visivi e testuali. Il modello CLIP ne è un esempio suggestivo: tenta di allineare le immagini con le rispettive descrizioni testuali all'interno di uno spazio di embeddings condiviso, attraverso una loss di contrasto.

In questo modo, CLIP consente attività di classificazione come zero-shot, in cui il modello è in grado di riconoscere classi mai viste prima sulla base di prompts definiti attraverso il linguaggio naturale. Tutto questo senza richiedere alcun perfezionamento ad hoc del modello.

Tuttavia, se CLIP funziona bene utilizzando prompts semanticamente ricchi, il suo comportamento sotto un'impostazione class-agnostic - cioè dove ogni prompt testuale non è allineato alle etichette delle classi - rimane meno esplorato.

Agiremo quindi analizzando il comportamento e la funzionalità di CLIP in contesti in cui questo modello non sembra eccellere. Lo scopo di questa tesi è quindi quello di comprendere se le embeddings prodotte da CLIP sono strutturate semanticamente in modo tale da favorire la separabilità delle classi, anche in assenza di una supervisione esplicita.

0.2 Objectives and Goals

Questa tesi mira ad analizzare la qualità rappresentativa delle embeddings di CLIP attraverso un'analisi dei tasks di classificazione sia supervisionata che non. L'obiettivo centrale è capire cosa "conosce" CLIP circa le categorie visive quando valutate in classificazioni standard, e in scenari zero-shot e class-agnostic. Nello specifico gli obiettivi di questo lavoro sono:

- Comprendere l'architettura di CLIP, il suo addestramento e il comportamento rappresentazionale nei tasks downstream, in particolare la classificazione.

- Valutare le prestazioni della classificazione di base utilizzando le embeddings di immagini CLIP su dataset standard (CIFAR-10 e CIFAR-100) con classificatori lineari supervisionati.
- Esplorare la capacità di classificazione zero-shot di CLIP misurando le prestazioni senza perfezionamento del modello e utilizzando prompts scritti in linguaggio naturale come descrittori di classe.
- Studiare le rappresentazioni di similarità class-agnostic, utilizzando prompts generici (non strettamente coincidenti con i nomi delle classi) per generare logit features per la classificazione.
- Identificare e risolvere potenziali colli di bottiglia nelle prestazioni class-agnostic, come la sparsità dei logits, prompts non ottimali e basse risoluzioni.
- Testare varie strategie di miglioramento per incrementare la performance nella classificazione.

0.3 Methodology

Questo studio è di natura sperimentale e orientato all'implementazione, combinando valutazioni supervisionate e non per analizzare la qualità delle rappresentazioni delle embeddings prodotte da CLIP. La metodologia è articolata nei seguenti passaggi:

1. **Preparazione di Modello e Dataset**: il modello CLIP preaddestrato (`ViT-B/32`) viene caricato utilizzando la libreria Transformers di Hugging Face. Per gli esperimenti verranno utilizzati dataset standard per image classification, come CIFAR-10 e CIFAR-100.
2. **Classificazione Baseine Supervisionate**: le embeddings delle immagini di CLIP vengono estratte e utilizzate come features di input per classificatori classici (SVM) al fine di misurare le performance in uno scenario completamente supervisionato.
3. **Valutazione Classificazione Zero-Shot**: vengono valutate le capacità zero-shot del modello generando prompts testuali corrispondenti ai nomi delle classi del dataset CIFAR, codificandoli nello spazio latente condiviso e calcolando la cosine similarity con le embeddings delle immagini.
4. **Rappresentazione Class-Agnostic dei Logit**: viene introdotto un insieme di prompts generici e class-agnostic (ad esempio, "*a photo of a small*

object", "*a photo of a blurry environment*") per derivare logit features. Queste features vengono valutate tramite classificatori lineari per determinare se le rappresentazioni di CLIP contengano struttura discriminativa utile anche in assenza di supervisione class-specific.

5. **Analisi Diagnostica e Ottimizzazione delle Performance**: vengono testate diverse strategie per migliorare l'accuratezza della classificazione class-agnostic, dopo un'analisi dei principali problemi riscontrati. In particolare, si discute:

   - dello scaling della temperatura per regolare le distribuzioni di cosine similarity;
   - del refinement dei prompts per ottenere descrizioni più consistenti e object-centric;
   - dell'uso di dataset ad alta risoluzione, come Tiny ImageNet, per testare la sensibilità alla risoluzione.

6. **Valutazione e Interpretazione**: metriche di accuracy, analisi di entropia e visualizzazioni vengono utilizzate per interpretare il comportamento delle diverse configurazioni. I risultati vengono confrontati con gli esperimenti benchmarks di classificazione baseline e zero-shot per comprendere i limiti della comprensione visiva implicita di CLIP.

7. **Clustering**: apprendimento non supervisionato circa la classificazione delle features delle immagini e sugli esperimenti di class-agnostic logits representation.

Tutti gli esperimenti sono stati implementati in Python utilizzando PyTorch e HuggingFace Transformers. Il codice completo è disponibile all'interno della repository GitHub reperibile al seguente indirizzo:
`https://github.com/NiccoBene00/DissectingCLIP`

Chapter 1

Project Overview

1.1 Reviewing Literature on CLIP

CLIP (Contrastive Language-Image Pretraining) è un modello preaddestrato multimodale che cerca di allineare semanticamente immagini e descrizioni in linguaggio naturale all'interno di uno spazio di embeddings condiviso. Strutturalmente, coinvolge due reti neurali: un Vision Transformer (ViT) per l'encoding delle immagini e un secondo modello per l'encoding dei testi, anch'esso basato su tecnologia Transformer. Più in dettaglio, queste reti mappano i tensori che rappresentano immagini e testi in embeddings di lunghezza fissa d (tipicamente $d = 512$), che il modello CLIP allinea congiuntamente tramite una contrastive loss chiamata InfoNCE (Noise-Contrastive Estimation).

Possiamo quindi affermare che CLIP è stato ottimizzato per ciò che è noto come *inter-modal alignment*, ovvero l'allineamento tra coppie immagine-testo. D'altra parte, l'uso di una contrastive loss di questo tipo può rendere il modello meno efficiente nell'*intra-modal alignment*, cioè quando si richiede, ad esempio, di associare immagini semanticamente simili. Alcuni studi, come descritto in [2], propongono una tecnica chiamata *modality-inversion* per colmare questa lacuna.

Da un punto di vista matematico, dato un batch di N coppie immagine-testo, CLIP calcola una matrice di similarità $N \times N$ tra tutte le embeddings di immagini e testi. Il modello viene quindi ottimizzato per massimizzare la similarità delle coppie corrette (diagonale) e minimizzare quella delle coppie errate (fuori diagonale). Questa strategia di apprendimento consente a CLIP di associare un'ampia gamma di concetti visivi a prompts in linguaggio naturale, raggiungendo performance state-of-the-art in zero-shot su diversi task vision-language - inclusi classification, retrieval e captioning - senza necessità di perfezionamento (fine-tuning) specifico per task.

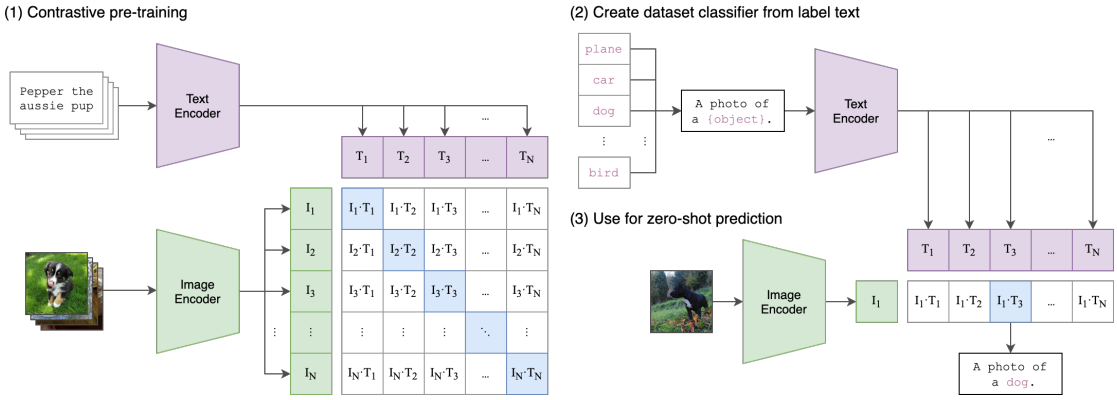

Figure 1.1: Struttura architetturale del modello CLIP. Immagine adattata da [1]

1.2 Mathematical Reminders

In questa sezione vengono presentati alcuni richiami matematici utili per comprendere meglio la struttura operativa di CLIP e il suo paradigma di contrastive learning. Vengono inoltre richiamati concetti di classificazione lineare e le metriche di accuracy sulle predizioni del modello. Tali richiami costituiscono una buona base teorica per muoversi consapevolmente all'interno degli esperimenti condotti.

Vector Spaces and Embeddings

Siano $\mathcal{X}$ e $\mathcal{Y}$ gli spazi di input per immagini e testi, rispettivamente. Gli encoder di CLIP $f_{\text{img}} : \mathcal{X} \to \mathbb{R}^d$ e $f_{\text{text}} : \mathcal{Y} \to \mathbb{R}^d$ proiettano gli input in uno spazio di embeddings condiviso di dimensione d (generalmente d=512). In particolare, CLIP accetta in input tensori che rappresentano immagini nello spazio $\mathbb{R}^{3\times224\times224}$, quindi per tutte le immagini nel dataset che non rispettano questo vincolo è necessario un resize. La procedura per le descrizioni testuali è differente: esse devono essere prima tokenizzate, così che, ad esempio, *"a photo of a lion"* diventi una sequenza di indici $\{t_1, t_2, \ldots, t_n\}$ del vocabolario, con $t_i \in \mathbb{Z}$.

A questo punto, i vettori codificati vengono normalizzati in norma ℓ_2 affinché $\|f(x)\|_2 = 1$ per ogni x, rendendo possibile l'utilizzo della cosine similarity come misura di similarità:

$$\text{sim}(x, y) = \frac{f_{\text{img}}(x) \cdot f_{\text{text}}(y)}{\|f_{\text{img}}(x)\| \, \|f_{\text{text}}(y)\|}.$$

Poiché le image embeddings f_{img} e i text embeddings f_{text} sono proiettate nello stesso spazio, la cosine similarity fornisce un modo scale-invariant per misurare

il loro allineamento. Essa garantisce che solo la direzione dei vettori influenzi la misura di similarità - ideale per confrontare rappresentazioni semantiche in alta dimensionalità. In pratica, entrambi le embeddings sono ℓ_2-normalizzate, quindi la cosine similarity si riduce al prodotto scalare $f_{\text{img}}(x) \cdot f_{\text{text}}(y) \in [-1; 1]$, dove 1 indica similarità perfetta. Nel contesto di CLIP, queste quantità sono chiamate logits.

Contrastive Learning

Il modello viene addestrato utilizzando una contrastive loss, in particolare la versione simmetrica della InfoNCE loss. Questa viene usata nel contrastive learning per avvicinare nello spazio delle embeddings le coppie positive (immagine-testo corretta) e allontanare le coppie negative (associazioni scorrette). Dato un batch di N coppie immagine-testo $\{(x_i, y_i)\}_{i=1}^{N}$, la loss per un'immagine x_i e il suo testo corrispondente y_i è:

$$\mathcal{L}_{\text{contrast}}^{\text{(image-to-text)}} = -\log \frac{\exp(\text{sim}(f_{\text{img}}(x_i), f_{\text{text}}(y_i))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f_{\text{img}}(x_i), f_{\text{text}}(y_j))/\tau)},$$

dove $\tau > 0$ è un parametro di temperatura, ovvero uno scalare che regola il livello di concentrazione della distribuzione. Un valore più basso di τ rende la distribuzione più "acuta", e il modello più sicuro nelle proprie predizioni. Tutto ciò è equivalente a una cross-entropy loss in cui il testo corretto è la classe target per l'immagine input.

Inoltre, viene calcolato anche un termine simmetrico (text-to-image) invertendo le modalità, così da garantire un allineamento bidirezionale:

$$\mathcal{L}_{\text{contrast}}^{\text{(text-to-image)}} = -\log \frac{\exp(\text{sim}(f_{\text{text}}(y_i), f_{\text{img}}(x_i))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f_{\text{text}}(y_i), f_{\text{img}}(x_j))/\tau)},$$

Questo vincola il modello affinché non solo le immagini ritrovino i testi corretti, ma anche i testi ritrovino le immagini corrispondenti. La loss finale usata durante l'addestramento è la media delle due direzioni:

$$\mathcal{L}_{\text{total}} = \frac{1}{2}\left(\mathcal{L}_{\text{contrast}}^{\text{(image-to-text)}} + \mathcal{L}_{\text{contrast}}^{\text{(text-to-image)}}\right)$$

Questo valore finale cade nell'intervallo $[0, \log(N)]$, dove 0 indica massima similarità tra immagine e testo. Una InfoNCE loss simmetrica di questo tipo si è dimostrata efficace nell'apprendere embedding congiunti per visione e linguaggio, come descritto anche in [3].

Numerical Example

In CLIP, sia le embeddings delle immagini che quelle dei testi vengono normalizzate a norma unitaria. Dati due vettori $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, la similarità coseno tra essi è definita come:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \mathbf{u}^\top \mathbf{v}$$

dal momento che $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$ per normalizzazione.

Sia

$$\mathbf{u} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}, \quad \mathbf{v}_1 = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$$

Allora,

$$\text{sim}(\mathbf{u}, \mathbf{v}_1) = 0.6 \cdot 0.707 + 0.8 \cdot 0.707 = 0.9898$$
$$\text{sim}(\mathbf{u}, \mathbf{v}_2) = 0.6 \cdot (-0.707) + 0.8 \cdot 0.707 = 0.1414$$

Questi valori rappresentano la similarità, secondo il modello, tra una determinata immagine e due differenti prompt testuali.

CLIP utilizza la InfoNCE loss per l'apprendimento contrastivo. Per la direzione image-to-text, la loss è definita come:

$$\mathcal{L}_{\text{img}\to\text{text}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp\left(\text{sim}(\mathbf{u}_i, \mathbf{v}_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(\mathbf{u}_i, \mathbf{v}_j)/\tau\right)}$$

Consideriamo ora un Batch Size pari a 2 con le seguenti similarità:

- $\text{sim}(\text{Img}_1, \text{Text}_1) = 0.746$
- $\text{sim}(\text{Img}_1, \text{Text}_2) = -0.232$
- $\text{sim}(\text{Img}_2, \text{Text}_2) = 0.710$
- $\text{sim}(\text{Img}_2, \text{Text}_1) = -0.250$

Assumendo $\tau = 0.07$:

$$\text{sim}_{1,1}/\tau = \frac{0.746}{0.07} \approx 10.657, \quad \text{sim}_{1,2}/\tau = \frac{-0.232}{0.07} \approx -3.314$$
$$Z_1 = \exp(10.657) + \exp(-3.314) \approx 42799.7 + 0.036 = 42799.736$$

$$\mathcal{L}_1 = -\log\left(\frac{\exp(10.657)}{Z_1}\right) = -\log\left(\frac{42799.7}{42799.736}\right) \approx -\log(0.99999916) \approx 8.4\times10^{-7}$$

$$\text{sim}_{2,2}/\tau = \frac{0.710}{0.07} \approx 10.143, \quad \text{sim}_{2,1}/\tau = \frac{-0.250}{0.07} \approx -3.571$$

$$Z_2 = \exp(10.143) + \exp(-3.571) \approx 25478.5 + 0.028 = 25478.528$$

$$\mathcal{L}_2 = -\log\left(\frac{\exp(10.143)}{Z_2}\right) = -\log\left(\frac{25478.5}{25478.528}\right) \approx -\log(0.9999989) \approx 1.1 \times 10^{-6}$$

Ovviamente, batch reali sono generalmente più rumorosi e di dimensioni maggiori, ma questo esempio illustra chiaramente il principio di funzionamento della loss.

Classification and Linear Models

Dato un insieme di coppie embedding-etichetta (z_i, y_i), consideriamo un modello lineare del tipo:

$$f(z) = z^\top W + b,$$

dove $W \in \mathbb{R}^{d\times C}$ è la matrice dei pesi e $z \in \mathbb{R}^d$ rappresenta una embedding (in particolare, l'output di CLIP). Il classificatore deve quindi apprendere i parametri W e b in modo da adattarsi ai concetti già separati nello spazio delle embeddings. In particolare, ogni colonna W_c di W rappresenta un vettore normale a un iperpiano decisionale associato alla classe c, ovvero la "direzione preferita" per quella classe. Il prodotto scalare $z^\top W_c$ rappresenta invece quanto l'embedding z sia compatibile con la classe c.

Evaluation Metrics

Per poter comprendere il comportamento del modello nei diversi esperimenti che verranno condotti, introduciamo ora alcune metriche di valutazione.

Classification Accuracy. È la metrica principale utilizzata nel corso degli esperimenti, soprattutto per valutare le prestazioni del classificatore. Data una serie di etichette predette $\hat{y}_i$ e le rispettive etichette reali y_i, per N campioni, l'accuratezza è definita come:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i)$$

dove $\mathbb{1}(\cdot)$ è la funzione indicatrice che restituisce 1 se la predizione è corretta, 0 altrimenti.

Entropy of Logits. Per analizzare il grado di confidenza e la sparsità dei logits agnostici alla classe, calcoliamo l'entropia dei vettori di logits normalizzati con softmax. Dato un vettore di logits $y_i \in \mathbb{R}^C$ per un'immagine di input (dove C è il numero di prompts testuali agnostici alla classe), si applica prima la *funzione softmax*, che trasforma i punteggi grezzi di similarità in una distribuzione di probabilità:

$$p_i = \frac{\exp(y_i/\tau)}{\sum_{k=1}^{C} \exp(y_k/\tau)}$$

Successivamente, si calcola l'entropia di Shannon del vettore di probabilità risultante p:

$$H(p) = -\sum_{i=1}^{C} p_i \log p_i$$

L'entropia quantifica l'incertezza della distribuzione. Un'entropia bassa indica una predizione più sicura o sparsa (ovvero, il modello favorisce fortemente poche classi), mentre un'entropia elevata corrisponde a maggiore incertezza, con i logit distribuiti più uniformemente tra le classi. È importante sottolineare che l'entropia misura solamente la "concentrazione" della distribuzione di probabilità, e non la correttezza delle predizioni. In altre parole, da sola l'entropia non è sufficiente per valutare l'efficacia del modello: si tratta di una metrica descrittiva, non predittiva.

È inoltre fondamentale ricordare che la funzione softmax ha il solo scopo di normalizzare i logits in un semplice di probabilità, ma non implica alcun processo di apprendimento.

Confusion Matrix. Per un'analisi più dettagliata a livello di classe, è possibile utilizzare matrici di confusione, che permettono di visualizzare le etichette predette rispetto a quelle reali. Questo tipo di analisi risulta particolarmente utile per identificare le classi con cui il modello ha maggiori difficoltà. Un'analisi accurata di questo tipo può evidenziare quanto efficacemente le embeddings di CLIP riescano a catturare la semantica di alcune coppie immagine-testo rispetto ad altre.

1.3 Tools and Framework

Questo progetto fa uso di diversi strumenti e librerie open-source per condurre esperimenti, analizzare i risultati e implementare pipeline di classificazione. I componenti principali includono:

- **CLIP (Contrastive Language–Image Pre-training):** È il modello centrale utilizzato nell'intera tesi. Le varianti pre-addestrate di CLIP sono

accessibili tramite la libreria `transformers` di Hugging Face, che fornisce sia encoder per immagini che per testo, permettendo una rappresentazione multimodale.

- **Dataset:** Gli esperimenti principali sono condotti su CIFAR-10 e CIFAR-100, che contengono immagini naturali a bassa risoluzione suddivise in diverse categorie. Ulteriori test sono eseguiti su Tiny ImageNet, un sottoinsieme ad alta risoluzione di ImageNet, per valutare l'impatto della qualità dell'immagine sulla rappresentazione.
- **PyTorch:** Il framework principale di deep learning utilizzato per tutte le operazioni sui tensori, l'inferenza dei modelli e l'estrazione delle feature. La flessibilità di PyTorch consente un'integrazione agevole con l'architettura CLIP e con pipeline personalizzate.
- **Hugging Face Transformers e Datasets:** Offrono un accesso semplificato a modelli CLIP pre-addestrati e a loader per i dataset.
- **Scikit-learn:** Utilizzato per l'addestramento di classificatori tradizionali (come le Support Vector Machines) sulle embeddings generate da CLIP, nonché per il calcolo di metriche di valutazione quali accuratezza e matrici di confusione.
- **NumPy e Matplotlib:** Impiegati per il calcolo numerico, l'aggregazione dei risultati e la visualizzazione di feature, distribuzioni di similarità ed entropia.

1.4 Expected Outcomes

Seguendo la metodologia descritta finora in questa tesi, ci si propone di comprendere come un modello multimodale come CLIP generi, grazie alla conoscenza acquisita durante il pre-addestramento, embeddings dotate di potere discriminativo. Se tale potere, in contesti agnostici alla classe, non dovesse risultare sufficiente per raggiungere buone performance negli esperimenti di riferimento, si prevede l'applicazione di tecniche di raffinamento volte ad aumentare l'accuratezza dei risultati.

Conclusion

In conclusione, questo capitolo ha fornito una panoramica introduttiva del progetto, delineandone gli obiettivi e l'ambito di applicazione. Stabilito questo

contesto, i capitoli successivi approfondiranno le metodologie, i risultati e le analisi che contribuiranno al raggiungimento degli obiettivi prefissati.

Chapter 2

CLIP Benchmarks

2.1 Introduction

Questo capitolo presenta una prima indagine empirica sulle prestazioni del modello CLIP in tasks standard di classificazione delle immagini. In particolare, utilizziamo la versione `openai/clip-vit-base-patch32` di CLIP come encoder d'immagini e ne valutiamo le rappresentazioni estratte sul dataset CIFAR-100.

L'obiettivo principale di questa fase è stabilire una baseline prestazionale che rifletta la qualità delle features visive apprese da CLIP. A tal fine, estraiamo embeddings di immagini di dimensione 512 da CLIP e addestriamo un classificatore lineare di tipo Support Vector Machine (SVM) su tali features.

Questa configurazione ci permette di verificare quanto efficacemente CLIP sia in grado di catturare semantiche visive significative e generalizzabili tra differenti compiti. Se l'SVM dovesse ottenere buone prestazioni, nonostante la complessità limitata del modello e l'assenza di ottimizzazione delle features, ciò indicherebbe che le rappresentazioni apprese da CLIP durante il pre-addestramento sono altamente informative - una proprietà desiderabile per scenari di tipo zero-shot.

Oltre a questa baseline supervisionata, valutiamo anche CLIP nella sua modalità nativa di classificazione zero-shot, utilizzando prompts testuali per associare le immagini alle etichette di classe senza alcun tipo di addestramento. Questo confronto mette in evidenza la differenza tra le capacità di ragionamento cross-modale di CLIP e la struttura rappresentazionale inter-modale, ponendo le basi per analisi più approfondite nei capitoli successivi.

2.2 Baseline Image Feature Classification

In questa sezione, definiamo una baseline sfruttando CLIP come estrattore di features. Applichiamo il modello ai tasks di classificazione delle immagini

su CIFAR-10 e CIFAR-100, addestrando una Support Vector Machine (SVM) lineare sui vettori delle embeddings estratte. L'obiettivo è valutare la qualità delle rappresentazioni visive di CLIP in contesti di classificazione intra-modale, senza modificare i pesi del modello.

2.2.1 Model and Dataset Setup

Utilizziamo il modello CLIP preaddestrato e il relativo processor forniti da Hugging Face:

```
clip_model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
clip_processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
clip_model = clip_model.to(device).eval()
```

I dataset CIFAR-10/CIFAR-100 vengono caricati tramite `torchvision` e incapsulati in una classe personalizzata che assicura il preprocessing corretto di ciascuna immagine (dimensione $3 \times 224 \times 224$), utilizzando il processor di CLIP:

```
class CLIPImageDataset(Dataset):
    def __init__(self, cifar_dataset):
        self.dataset = cifar_dataset

    def __getitem__(self, idx):
        image, label = self.dataset[idx]
        processed = clip_processor(images=image,
           return_tensors="pt")
        return processed["pixel_values"].squeeze(0), label
```

2.2.2 Embedding Extraction

Le embeddings vengono ottenute propagando le immagini preprocessate attraverso il modello CLIP congelato (cioè senza alcun fine-tuning). Ogni immagine viene mappata in un vettore di embeddings 512-dimensionale:

```
def extract_embeddings(dataloader):
    all_embeddings, all_labels = [], []
    with torch.no_grad():
        for images, labels in dataloader:
            images = images.to(device)
            embeddings = clip_model.get_image_features(
               images)
            all_embeddings.append(embeddings.cpu().numpy())
```

```
            all_labels.extend(labels.numpy())
    return np.vstack(all_embeddings), np.array(all_labels)
```

2.2.3 Classification and Evaluation

Le features estratte vengono standardizzate con `StandardScaler` per garantire una distribuzione uniforme tra le dimensioni (obiettivo: media = 0 e deviazione standard = 1 per ogni dimensione), e successivamente viene addestrata una SVM lineare:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(train_features)
X_test_scaled = scaler.transform(test_features)

svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
svm_clf.fit(X_train_scaled, train_labels)
y_pred = svm_clf.predict(X_test_scaled)
```

L'accuratezza della classificazione viene infine calcolata come segue:

```
accuracy = accuracy_score(test_labels, y_pred)
print(f"Baseline CLIP image classification: {accuracy *
    100:.2f}%")
```

2.2.4 Final Insights

Il dataset CIFAR-100 rappresenta una sfida maggiore rispetto a CIFAR-10 a causa delle sue 100 categorie "a grana fine", che spesso includono sottotipi visivamente simili (ad esempio, diverse specie animali o varianti di oggetti). Il calo osservato dell'accuratezza di classificazione, dal 93.24% su CIFAR-10 al 78.01% su CIFAR-100, riflette un fattore chiave: l'aumento della variabilità semantica e visiva tra le classi rende la discriminazione più difficile se si utilizzano le embeddings senza alcun fine-tuning.

È importante ricordare inoltre che il dataset CIFAR utilizza immagini di dimensione $[3 \times 32 \times 32]$ (canali RGB, altezza, larghezza), il che rende complesso, anche per un modello avanzato come CLIP, mantenere informazioni visive di dettaglio durante la trasformazione $[3 \times 32 \times 32] \longrightarrow [3 \times 224 \times 224] \longrightarrow$ embedding del modello CLIP.

Un'analisi dettagliata dei risultati di classificazione mette in evidenza le seguenti tendenze:

- **Classi ad alte prestazioni** (F1-score $\geq 0{,}90$): tipicamente comprendono oggetti con forme o aspetti distintivi, come *pickup truck*, *wardrobe*, *keyboard* e *chair*.
- **Classi a basse prestazioni** (F1-score $\leq 0{,}60$): spesso coinvolgono categorie naturali visivamente simili come *beaver*, *shrew*, *seal* e *crocodile*, difficili da distinguere a bassa risoluzione (spesso anche un umano sbaglierebbe!).

Complessivamente, la bassa performance riscontrata in molte classi di tipo animale, rispetto a oggetti o veicoli, è coerente con l'ipotesi secondo cui l'addestramento di CLIP su coppie immagine-testo estratte dal web favorisca entità distintive e ricche di contesto rispetto a distinzioni biologiche più sottili. Questo evidenzia l'importanza di considerare la risoluzione del dataset come fattore determinante.

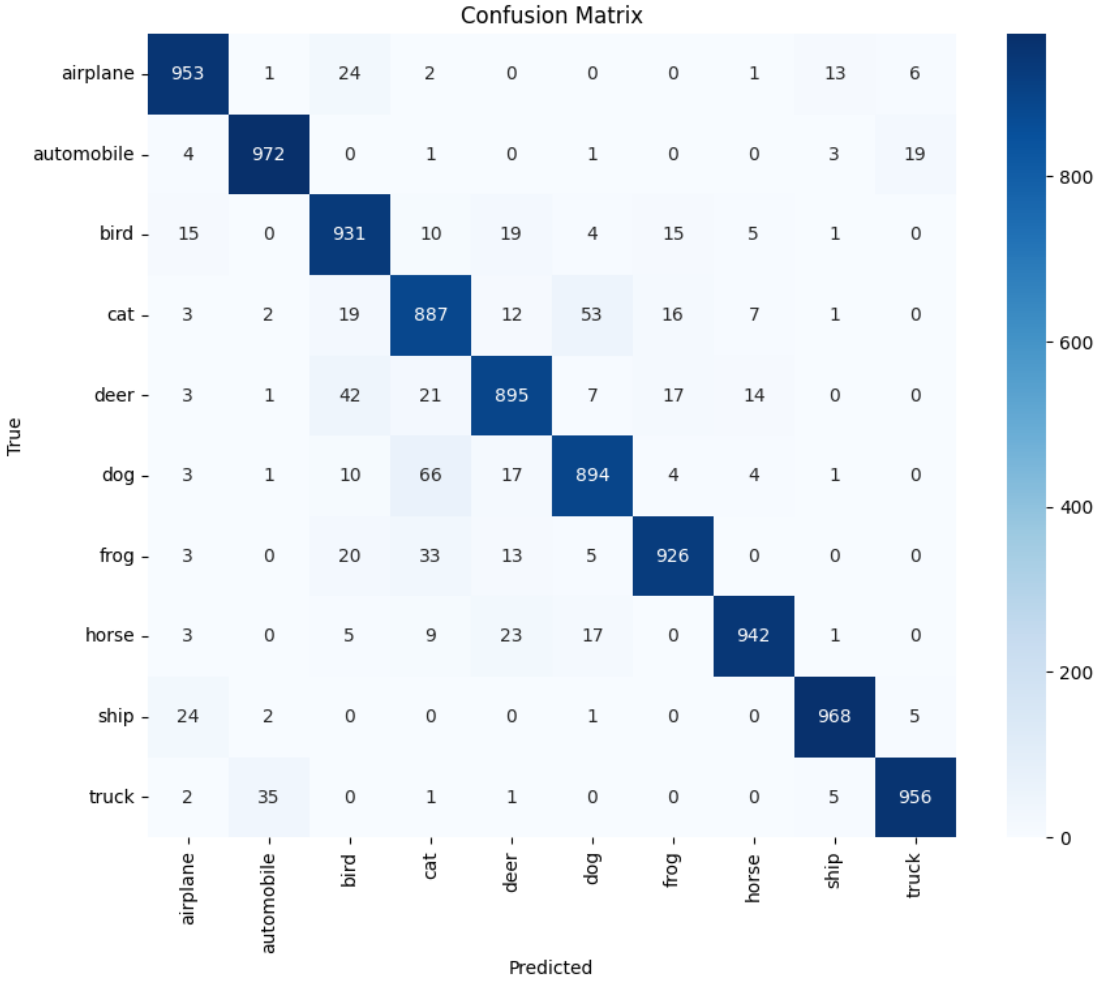

Figure 2.1: Matrice di confusione su CIFAR-10 — Classificazione Baseline.

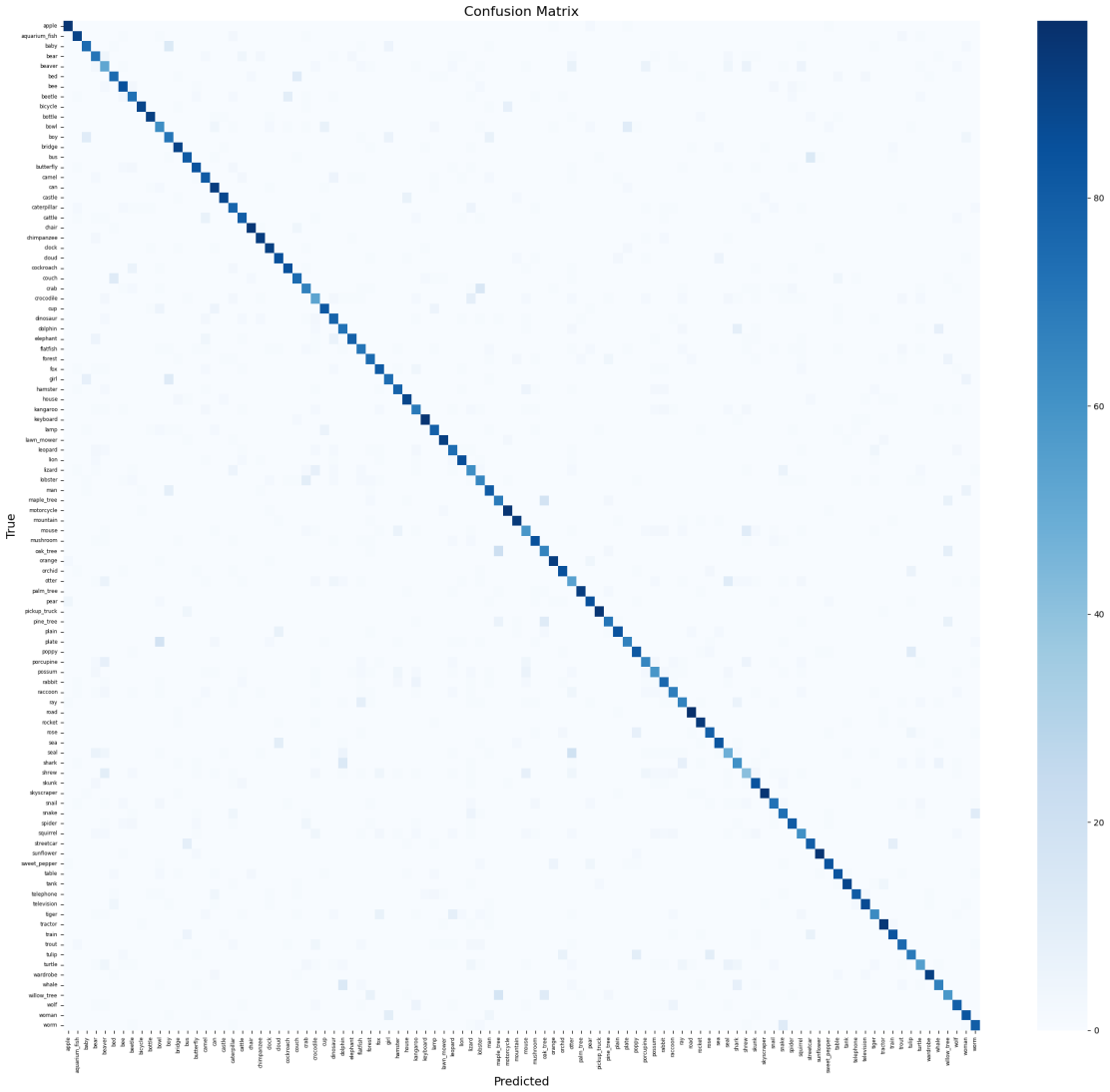

Figure 2.2: Matrice di confusione su CIFAR-100 — Classificazione Baseline.

2.3 Zero-Shot Classification

È noto che CLIP è un modello pre-addestrato su milioni di coppie immagine-testo con l'obiettivo di massimizzare la similarità tra tali coppie. Una delle caratteristiche più interessanti del modello risiede nella capacità di eseguire compiti di classificazione in modalità *zero-shot*.

In altre parole, utilizzando CIFAR come dataset di riferimento, vengono estratte le caratteristiche associate alle descrizioni testuali tipiche delle etichette del dataset

(es. "*a photo of ...*"). Successivamente, tali rappresentazioni testuali vengono confrontate con quelle ottenute dalle immagini dello stesso dataset, al fine di valutare la similarità tramite distanza coseno.

Questo ci permette di analizzare come CLIP riesca a predire correttamente le classi in un contesto di classificazione *inter-modale*, nonostante il modello non abbia mai visto elementi del dataset CIFAR durante la fase di addestramento.

Il concetto di predizione zero-shot si basa sul fatto che non è necessario addestrare alcun classificatore: si forniscono semplicemente a CLIP un'immagine e una lista di descrizioni testuali possibili, e si sfrutta la similarità tra le rispettive embeddings per effettuare la predizione. Naturalmente, quanto più lo spazio delle embeddings prodotto da CLIP è informativo e discriminante, tanto più veritiera risulterà la misura di similarità tra immagine e testo.

2.3.1 Experiment Setup

A seguito dell'esperimento di classificazione di baseline, si procede con la valutazione delle capacità *zero-shot* di CLIP sui dataset CIFAR-10 e CIFAR-100.

CIFAR è composto da categorie semantiche che possono essere espresse naturalmente in linguaggio umano. Per allinearci al paradigma di pre-addestramento di CLIP, costruiamo dei prompts testuali nella forma `"a photo of a {class}"` per ciascuna etichetta. Queste descrizioni in linguaggio naturale vengono tokenizzate (direttamente tramite il `clip processor`) e codificate in embedding testuali di 512 dimensioni utilizzando l'encoder testuale di CLIP:

```
text_prompts = [f"a photo of a {label}" for label in
    class_names]

# tokenizzazione dei prompt
text_inputs = clip_processor(text=text_prompts,
    return_tensors="pt", padding=True).to(device)

text_features = clip_model.get_text_features(**text_inputs)
text_features = text_features / text_features.norm(dim=1,
    keepdim=True)
```

Successivamente, ogni immagine di test viene elaborata attraverso l'encoder visivo di CLIP, e viene calcolata la similarità coseno tra il vettore di embeddings dell'immagine e tutte le embeddings testuali associati alle classi:

```
inputs = clip_processor(images=image, return_tensors="pt").
    to(device)
image_features = clip_model.get_image_features(**inputs)
```

```
image_features = image_features / image_features.norm(dim=1,
    keepdim=True)

similarity = (image_features @ text_features.T).unsqueeze(0)
pred_label = similarity.argmax().item()
```

In questo modo, CLIP effettua una classificazione inter-modale, associando a ciascuna immagine la classe testuale con embedding più simile. Questo processo non richiede alcuna fase di addestramento ulteriore, caratterizzando pienamente il paradigma *zero-shot*.

2.3.2 Final Insights

Rispetto all'esperimento condotto su CIFAR-10, si osserva un chiaro degrado delle prestazioni su CIFAR-100:

- Su CIFAR-10, CLIP ottiene un'accuratezza dell'88.8% utilizzando la classificazione *zero-shot* con prompts in linguaggio naturale.
- Su CIFAR-100, CLIP continua a comportarsi in maniera competitiva, raggiungendo un'accuratezza del 61.71%, nonostante la maggiore difficoltà dovuta alla presenza di 100 classi, molte delle quali sono finemente suddivise e visivamente simili, oltre alla bassa risoluzione delle immagini (32×32).

Questi risultati mettono in evidenza sia i punti di forza che i limiti di CLIP: alla luce di quanto osservato, possiamo affermare che CLIP è piuttosto efficace nel creare rappresentazioni semantiche in grado di discriminare tra categorie generali. Tuttavia, tende a generare confusione quando queste rappresentazioni vengono estratte da dataset a bassa risoluzione e valutate mediante classificatori leggeri, come una SVM lineare.

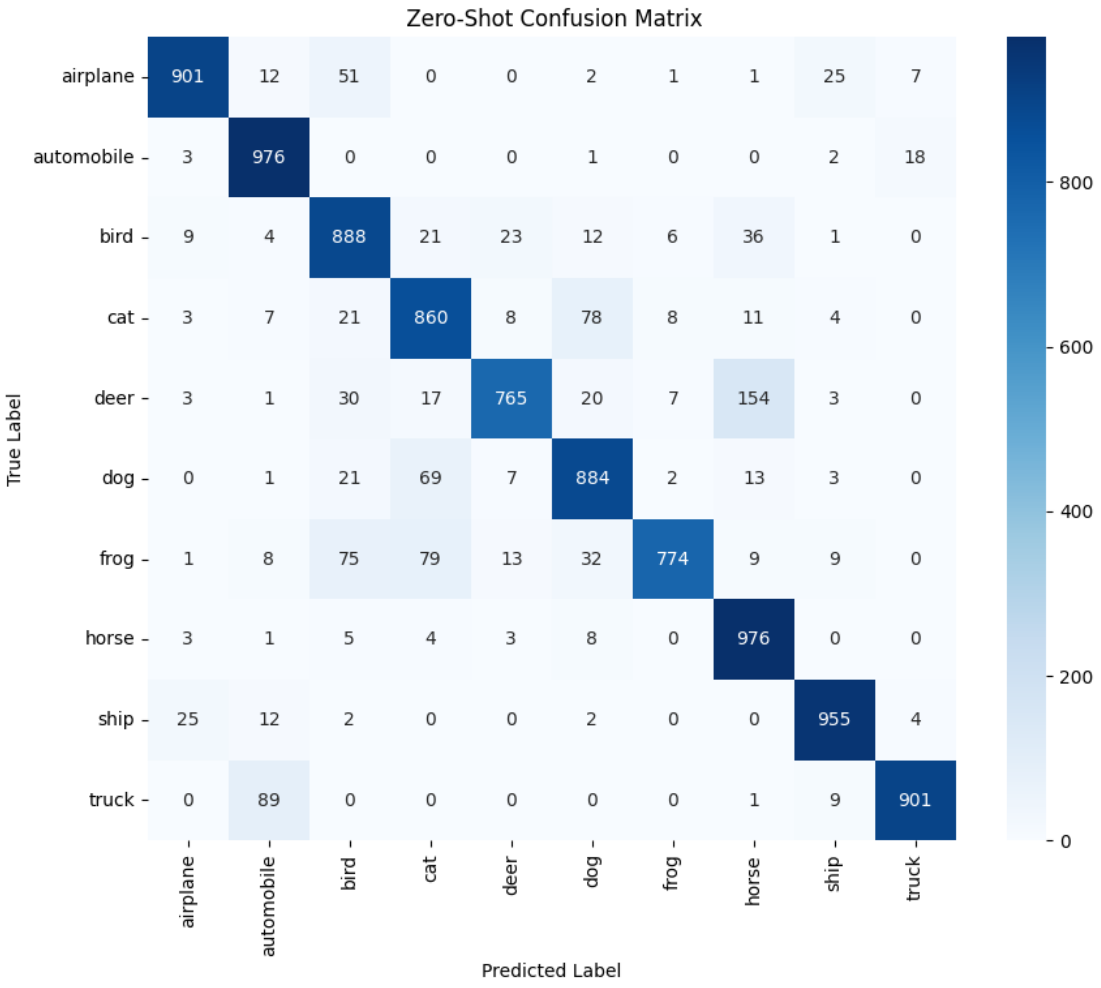

Figure 2.3: Matrice di confusione su CIFAR-10 - Classificazione Zero-Shot.

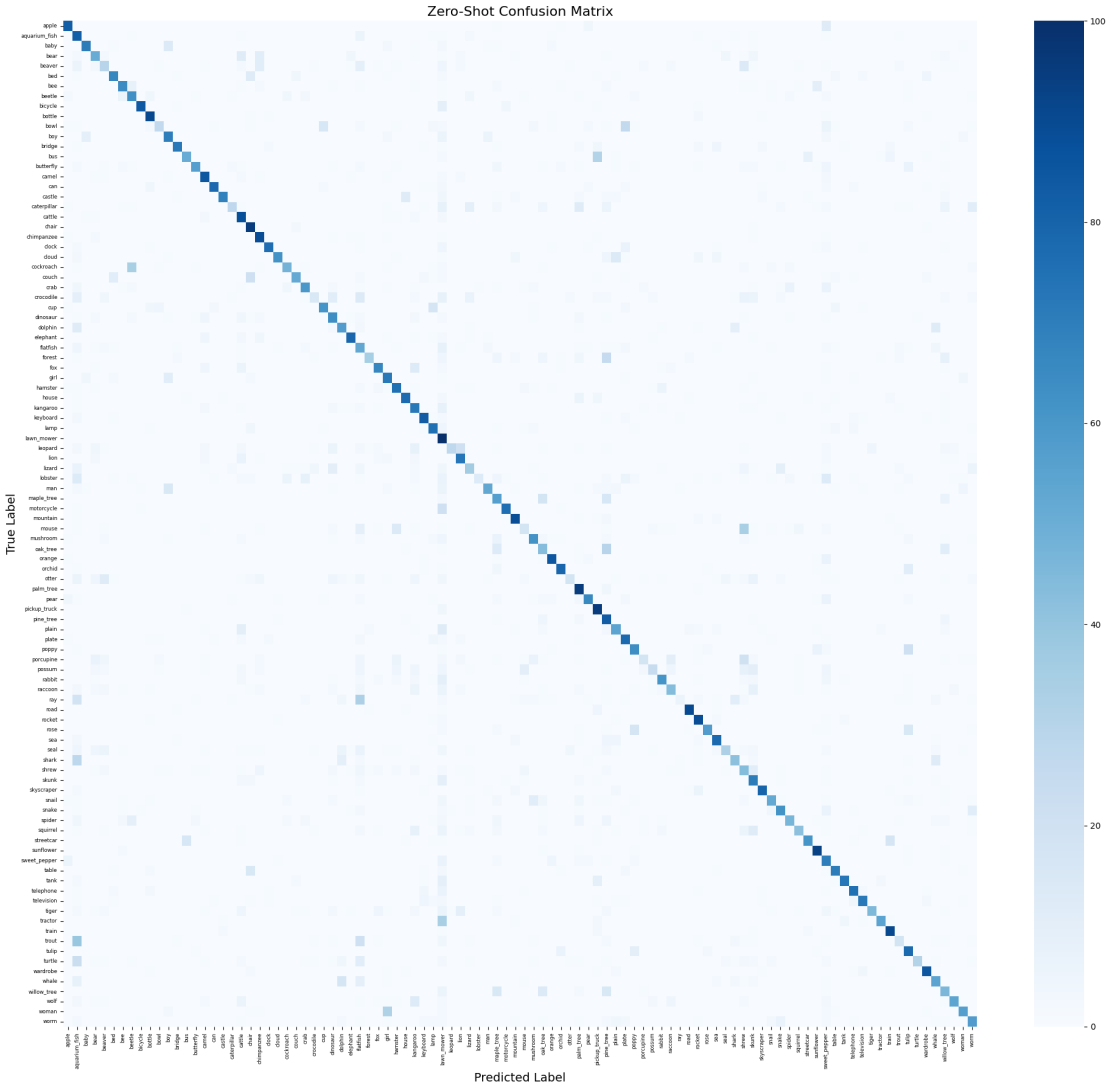

Figure 2.4: Matrice di confusione su CIFAR-100 - Classificazione Zero-Shot.

Chapter 3

Class-Agnostic Techniques

Introduction

In questo capitolo esploriamo il comportamento del modello CLIP in un contesto *class-agnostic*. Si tratta di un approccio molto utile per comprendere la struttura rappresentazionale delle embeddings. Il cuore dell'esperimento consiste infatti nell'estrarre le similarità tra le immagini e un insieme di prompts generici, le cui descrizioni sono semanticamente simili alle classi reali del dataset CIFAR, per poi addestrare un classificatore SVM lineare su queste caratteristiche. Questo approccio si contrappone agli esperimenti di baseline presentati nel Capitolo 2, dove la classificazione veniva effettuata direttamente sulle embeddings delle immagini generate da CLIP.

Nel contesto di modelli come CLIP, il termine *logits* si riferisce ai punteggi grezzi di similarità coseno. Un'impostazione *class-agnostic logit representation*, come quella proposta, tenta di utilizzare questi punteggi come caratteristiche senza vincolare il modello alla conoscenza delle vere classi del dataset di riferimento.

La nostra analisi è strutturata attorno alle seguenti domande guida:

- I *logits* di similarità di CLIP sono semanticamente ricchi al punto da supportare la classificazione di oggetti anche in assenza di etichette durante la fase di estrazione delle caratteristiche?
- La struttura interna delle similarità apprese da CLIP è in grado di generalizzare a nuovi compiti o dataset senza ulteriore fine-tuning?
- Dove risiede l'informazione semantica più utile in CLIP - nelle embeddings delle immagini o nei logits di similarità?

Anche in questo caso, è importante sottolineare che non viene effettuato alcun aggiornamento o fine-tuning del modello CLIP; l'SVM viene utilizzato esclusi-

vamente come strumento diagnostico - una *probe* - per valutare l'efficacia e la generalità delle rappresentazioni pre-addestrate.

Le sezioni successive illustrano la metodologia adottata, gli esperimenti condotti e le osservazioni emerse da questa analisi class-agnostic delle rappresentazioni interne di CLIP.

3.1 Class-Agnostic Logits for CLIP Classification

In questa sezione, analizziamo se i *similarity logits* - calcolati tra le immagini del dataset CIFAR-100 e un insieme di prompts generici, non specifici per classe - possano costituire caratteristiche efficaci per compiti di classificazione downstream. A differenza degli approcci tradizionali, che si basano su embeddings delle immagini o etichette specifiche, il nostro metodo sfrutta la capacità di CLIP di ragionare su concetti semantici di alto livello attraverso la similarità tra testo e immagine.

3.1.1 Prompt Design

Abbiamo costruito una lista di 32 prompts progettati per riflettere concetti visivi generali, piuttosto che categorie specifiche del dataset. Questi includono attributi semantici come *"a flying thing"*, *"a photo taken outside"* oppure *"a blurry photo"*. Questo approccio ci permette di verificare se CLIP sia in grado di associare le immagini a concetti astratti che vanno oltre i confini di classe specifici:

```
prompts = [
    "a photo of an object",
    "a photo of something natural",
    "a photo of something man-made",
    "a blurry photo",
    "a close-up photo",
    "a photo taken during the day",
    "a photo taken outdoors",
    "a photo taken indoors",
    "a small object",
    "a large object",
    "an animal",
    "a machine",
    "a thing that moves",
    "a stationary object",
    "a photo of a living thing",
    "a photo of a synthetic thing",
    "a colorful object",
    "a grayscale image",
```

```
    "a smooth surface",
    "a textured object",
    "a plastic object",
    "a metallic object",
    "a flying thing",
    "a photo of something round",
    "a rectangular object",
    "a noisy image",
    "an object with wheels",
    "an object with wings",
    "an underwater object",
    "a photo from a low angle",
    "a photo from a top view",
    "a centered object",
]
```

I prompts vengono tokenizzati e codificati tramite il codificatore di testo CLIP. Le embeddings risultanti vengono normalizzati per consentire confronti di similarità del coseno:

```
# prompt tokenizing
text_inputs = clip_processor(prompts, return_tensors="pt",
    padding=True).to(device)

with torch.no_grad():
    text_features = clip_model.get_text_features(**
       text_inputs)

text_features = text_features / text_features.norm(dim=-1,
    keepdim=True)
```

3.1.2 Feature Extraction via Class-Agnostic Logits

Per estrarre le caratteristiche per la classificazione, calcoliamo le similarità del coseno (*logits*) tra ciascuna immagine e l'intero set di prompts. Questi logits vengono quindi utilizzati come vettori di features per un classificatore downstream:

```
def extract_logit_features(data_loader, text_features):
    image_features_list = []
    labels_list = []

    for inputs, labels in tqdm(data_loader):
        with torch.no_grad():
```

```
            image_features = model.get_image_features(**
                inputs)
            image_features = image_features / image_features
                .norm(dim=1, keepdim=True)

            # evaluating cosine similarity
            logits = image_features @ text_features.T

        image_features_list.append(logits.cpu().numpy())
        labels_list.append(labels.cpu().numpy())

    features = np.concatenate(image_features_list, axis=0)
    labels = np.concatenate(labels_list, axis=0)
    return features, labels
```

3.1.3 Training and Evaluation

Una Support Vector Machine (SVM) lineare viene addestrata sulle features logit estratte dal set di addestramento CIFAR-100. Il modello viene quindi valutato sul set di test per valutare la qualità di queste rappresentazioni class-agnostic:

```
svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
svm_clf.fit(train_features, train_labels)

test_preds = svm_clf.predict(test_features)
test_acc = accuracy_score(test_labels, test_preds)
print(f"\nTest accuracy using Linear SVM: {test_acc * 100:.2
    f}%")
```

3.1.4 Final Insights

L'esperimento class-agnostic ha prodotto un'accuratezza sul test del 29.30%, che - sebbene significativamente superiore al caso casuale (1% per CIFAR-100) - risulta comunque nettamente inferiore sia rispetto alla classificazione baseline (~78%) sia rispetto alla performance zero-shot di CLIP (~61.71%).

Questo risultato offre una prospettiva non troppo nitida circa l'efficacia dei logits di similarità grezzi come rappresentazioni:

- Il modello riesce a estrarre *una certa* informazione discriminativa tra classi anche da prompts generici e semanticamente astratti, pur in assenza di indicazioni specifiche sulle classi.

- Tuttavia, il divario prestazionale suggerisce che i logits class-agnostic potrebbero non avere una risoluzione o specificità sufficienti per distinguere in modo robusto tra classi fini come quelle presenti in CIFAR-100.

Questo esperimento mette in evidenza una tensione critica nell'uso di modelli visione-linguaggio pre-addestrati per compiti di classificazione fine-grained, anche se CLIP riesce comunque ad allineare, almeno in parte, la semantica tra immagini e descrizioni anche in contesti non supervisionati.

Guardando avanti, questa performance motiva la prossima sezione, in cui vengono esplorati potenziali miglioramenti nella strategia class-agnostic.

Queste indagini mirano a diagnosticare e mitigare le limitazioni emerse in questo contesto, offrendo miglioramenti che conducano verso una generalizzazione più robusta a partire da features pre-addestrate.

3.2 Analyzing the Limitations of Class-Agnostic Logits Representations

Abbiamo visto che l'approccio class-agnostic ha prodotto un'accuratezza del 29.30%; questa percentuale è notevolmente inferiore sia alla classificazione zero-shot (61.71%) che alla baseline supervisionata (78.01%). In questa sezione, esaminiamo le ragioni alla base di questo divario prestazionale e delineamo diverse limitazioni chiave emerse dalla nostra analisi.

Sparse Logit Distributions

Potremmo pensare che i punteggi di similarità del coseno tra le embeddings di immagini e prompts - usati come features per la classificazione - tendano a produrre distribuzioni sparse. Calcolare la sparsità dei logits - attraverso l'analisi dell'entropia come metrica presentata nel primo capitolo - ci aiuta ad avere una visione d'insieme più chiara. Andremo ora ad arricchire la funzione di estrazione delle features, presentata sopra, al fine di implementare un'analisi entropica.

```
def extract_logit_features(data_loader, text_features):
  image_features_list = []
  labels_list = []

  entropies = []

  for inputs, labels in tqdm(data_loader):
    with torch.no_grad():
```

```
        image_features = model.get_image_features(**inputs)
        image_features = image_features / image_features.norm(
            dim = 1, keepdim = True)

        # cosine similarity
        logits = (image_features @ text_features.T)

        # Softmax Function for transforming logits into
            normalized probabilities
        probs = F.softmax(logits, dim = 1)

        # Entropy Anlysis for Sparsity Metrics
        entropy = -torch.sum(probs * torch.log(probs + 1e-9),
            dim = 1)

        entropies.append(entropy.cpu().numpy())

    image_features_list.append(logits.cpu().numpy())
    labels_list.append(labels.cpu().numpy())

features = np.concatenate(image_features_list, axis = 0)
labels = np.concatenate(labels_list, axis = 0)

all_entropy = np.concatenate(entropies)

print(f"Sparsity analysis through mean entropy: {
    all_entropy.mean():.4f}")

return features, labels
```

Poiché abbiamo 32 prompts generici, l'entropia massima teorica - ovvero quando la distribuzione è perfettamente uniforme - è

$$\log(32) \approx 3.4657.$$

Dall'analisi entropica descritta sopra, otteniamo un'entropia media di circa 3.4656, un valore molto vicino al massimo teorico. Questo implica che i logits non sono affatto sparsi, ma al contrario ampiamente distribuiti. In altre parole, le immagini presentano una similarità (coseno) simile con tutte le 32 classi, e quindi il modello non riesce a distinguere bene tra le classi generiche da noi selezionate.

Un risultato del genere era piuttosto prevedibile, poiché le classi testuali generiche sono forse semanticamente troppo simili tra loro per risultare discriminative rispetto alle immagini di CIFAR-100. Anche un essere umano potrebbe confondersi

tra "*a photo of something round*" e "*a centered object*" quando si ha a che fare con immagini a bassa risoluzione 32×32.

Abbiamo inoltre testato prompts semanticamente mirati con una cardinalità maggiore pari a 99 classi. La distribuzione di probabilità dei logits ottenuti da CLIP mostra un'entropia media che rimane sistematicamente vicina al massimo teorico,

$$\log(99) \approx 4.5951,$$

indicando nuovamente una mancanza di decisione netta tra le classi. Seguendo l'intuizione presentata nel lavoro di Balanya et al. [4], che evidenzia una correlazione positiva tra entropia e necessità di calibrazione tramite temperature scaling, si suggerisce di sperimentare l'uso delle temperature. Infatti, l'utilizzo di temperature basse potrebbe aumentare la nitidezza della distribuzione, ovvero amplificare le differenze discriminative nell'attivazione tra le classi.

Limitations of Linear Classifiers

Una SVM lineare, pur essendo semplice e interpretabile, potrebbe non essere sufficientemente espressiva per sfruttare la struttura non lineare incorporata nello spazio dei logits. Poiché le embeddings di CLIP risiedono in uno spazio ad alta dimensionalità, potenzialmente curvo, un modello lineare potrebbe non riuscire a catturare correttamente i confini tra le classi. Infatti, un classificatore lineare (come una SVM con kernel lineare) si comporta in questo modo:

$$f(z) = z^\top W + b,$$

In altre parole, può solamente tracciare confini decisionali lineari (iperpiani) nello spazio delle features. Tuttavia, nel nostro caso le features class-agnostic sono ad alta dimensionalità ma non linearmente separabili: i logits derivano da similarità coseno con prompts generici che non corrispondono direttamente a nessuna delle 100 etichette di CIFAR-100. Quindi, oggetti appartenenti a classi diverse possono avere vettori di logits molto simili, rendendo inefficaci i confini lineari.

A supporto di questa osservazione possiamo considerare il lavoro presentato in [5]. Gli autori propongono un modello chiamato CASVM, che combina una CNN arricchita con un meccanismo di *coordinate attention* (CA) per l'estrazione di features più efficaci, e una SVM come strato finale. Sperimentando su Fashion-MNIST, CIFAR-10, CIFAR-100 e Animal10, il lavoro mostra che CASVM supera i modelli standard basati su softmax, offrendo migliori accuratezze, maggiore robustezza e perdite finali più basse. Questo conferma che le features non lineari estratte da una CNN spesso non sono separabili linearmente nello spazio ad alta dimensionalità. CASVM affronta la stessa criticità del nostro scenario: rendere le features estratte quanto più discriminative possibile per la classificazione (grazie alla CA).

Detto ciò, anche un semplice modello MLP (*Multilayer Perceptron*) potrebbe già essere in grado di discriminare meglio all'interno dello spazio ad alta dimensionalità degli embedding.

Limitations of Prompts Vocabulary

Quando si utilizza un insieme limitato e poco variegato di prompts testuali, si corre il rischio di non coprire adeguatamente le diverse sfumature semantiche dei concetti visivi contenuti nelle immagini di CIFAR-100. Di conseguenza, le embeddings estratte da queste immagini non trovano facilmente un allineamento con le descrizioni scelte. Aumentare la varietà del vocabolario, ovvero utilizzare più prompts, ma soprattutto più descrizioni per gli stessi concetti, potrebbe aiutare a ottenere una maggiore separazione informativa nelle embeddings prodotte da CLIP.

Low Image Resolution

Le immagini di CIFAR-100 sono solo di 32×32 pixel, il che limita significativamente la quantità di dettagli visivi disponibili per la rappresentazione. In effetti, potrebbe esserci una grande perdita di informazione nella transizione tra l'immagine CIFAR e l'embedding prodotto da CLIP (vettore a 512 dimensioni). Questa riduzione di risoluzione può dunque ostacolare la capacità dell'encoder di CLIP di produrre embeddings semanticamente ricche, specialmente se combinate con prompts astratti. Come dimostrano anche gli autori in [6], la classificazione fine-grained su dataset con immagini a bassa risoluzione subisce un drastico calo delle performance a causa della perdita di dettagli rilevanti. Per cercare di risolvere questo problema, gli autori integrano un modulo di super-risoluzione nella pipeline che ripristina dettaglio e precisione. Il loro modello utilizza infatti una funzione di perdita che incorpora attributi aggiuntivi (*attribute-assisted loss*) per rendere le features più discriminative e meno ambigue tra classi simili. Valutando il modello su risoluzioni di 32×32, 64×64, 128×128 e 224×224, osservano come la qualità della classificazione aumenti al crescere della risoluzione.

Queste osservazioni guidano i nostri prossimi passi nel miglioramento della configurazione class-agnostic. Nella sezione seguente, esploriamo strategie specifiche volte a mitigare tali criticità, migliorare la qualità delle features estratte e, in ultima analisi, colmare il divario rispetto alle performance ottenute in modalità zero-shot.

Chapter 4

CLIP Logits Enhancements

Introduction

In questo capitolo cerchiamo di mitigare le limitazioni della rappresentazione tramite logits calcolata nel Capitolo 3, al fine di ottenere features più discriminative necessarie per una classificazione più precisa. Infatti, dagli esperimenti class-agnostic finora condotti, sembrerebbe che non si stia sfruttando appieno il potenziale semantico di CLIP nei contesti di classificazione. È quindi opportuno intervenire a livello qualitativo sullo spazio delle features, cercando di potenziare le caratteristiche dei logits.

Il capitolo che segue presenta alcune tecniche di calibrazione, come l'utilizzo di fattori di scaling della temperatura, che agiscono direttamente sulla forma dei logits con l'obiettivo di ricalibrare le probabilità naive del modello. Allo stesso tempo, si analizza l'utilizzo di diverse cardinalità di prompts con una maggiore varietà di vocabolario, per aumentare sensibilmente l'efficacia dei dati derivati dai logits. Infine, osserviamo come le features estratte da CLIP, utilizzando dataset contenenti immagini a risoluzione più alta rispetto a quelle finora usate, abbiano un impatto sull'efficacia del potere discriminativo della rappresentazione tramite logits.

4.1 Temperature Scaling

In questa sezione analizziamo l'efficacia dell'integrazione della tecnica di temperature scaling all'interno della pipeline di CLIP per passare da logits grezzi a output più affidabili. I valori di entropia misurati nel Capitolo 3 riportano una produzione di logits poco confidenti e scarsamente calibrati. Applicando la temperature scaling come tecnica di post-calibrazione ai logits otteniamo una distribuzione di probabilità definita da

$$\hat{p}_k = \frac{\exp(z_k/\tau)}{\sum_{j=1}^{K} \exp(z_j/\tau)}.$$

Nel nostro scenario, i logits sono generati tramite similarità coseno tra le embeddings visive e i prompts testuali, semanticamente allineati ai nomi delle classi. Tuttavia, se i prompts non sono perfettamente discriminativi, la distribuzione di probabilità dei logits tende a essere piatta, come già osservato tramite l'entropia. In altre parole, il modello assegna valori decisionali simili a molte classi. L'obiettivo è quindi osservare se, per $\tau < 1$, tale distribuzione venga amplificata, riuscendo così ad accentuare anche le differenze tra classi. Si sperimenta inoltre come l'uso di questa tecnica possa influenzare lo spazio geometrico dei logit features osservando se il classificatore SVM lineare sia maggiormente in grado di trovare iperpiani che discriminino le classi.

Di particolare interesse è anche studiare il comportamento del modello al variare del parametro τ. Questa analisi ci consente di individuare una regione di valori di τ che possiamo definire cutoff, ovvero al di sotto della quale scalare i logits con temperature ancora più basse porta a definire uno spazio delle features distorto, in cui le classi appaiono molto diverse anche se semanticamente non lo sono. I risultati sperimentali ottenuti sono riassunti nella seguente tabella e nel relativo grafico.

τ	**Entropia Media**	**Accuracy SVM**
0.50	3.4649	50.62% ↑
0.07	3.4206	62.95% ↑
0.03	3.2099	63.76% ↑
0.02	2.8957	63.45% ↓

Table 4.1: Risultati variando il parametro di temperatura sui logits di CLIP.

Si nota che quando la temperatura scende sotto τ=0.07 l'accuratezza del classificatore lineare sembra convergere verso il limite superiore del 64%. Si osserva persino un'inversione di tendenza tra τ=0.03 e τ=0.01, ovvero il fattore τ diminuisce e l'accuratezza smette di aumentare. Questo è sintomo che la temperature scaling non è più utile al classificatore in termini di discriminazione dei logits. Abbiamo dunque identificato la regione di cutoff.

Quando $\tau < 0.07$, anche piccole differenze nei valori di similarità z_{ij} (ottenuti tramite similarità coseno tra l'embedding visivo i e quello testuale j) vengono esagerate, inducendo una distribuzione di output fortemente appuntita e overconfident. Questo fenomeno si traduce in una distorsione dello spazio delle features. Poiché la discriminazione è guidata da un artefatto numerico (il basso valore di τ),

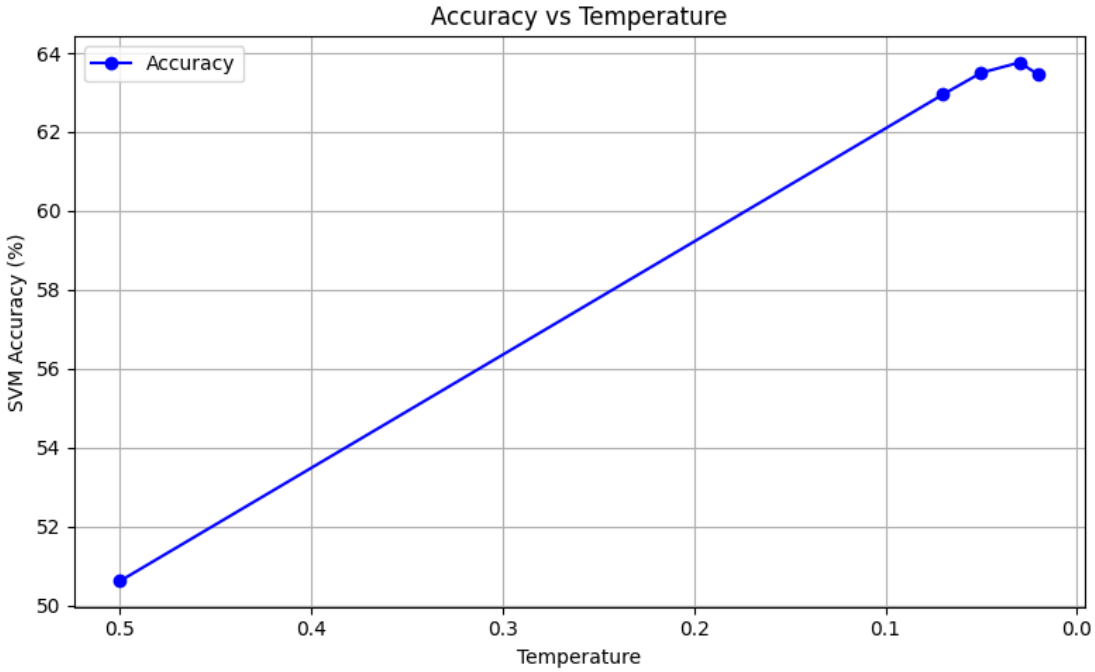

Figure 4.1: Relazione tra temperatura e accuratezza SVM.

le relazioni geometriche tra le embeddings vengono distorte: classi semanticamente vicine (es. "*animal*" vs "*creature*") appaiono artificialmente distanti nello spazio dei logits.

4.2 Increasing Prompts Vocabulary

Sappiamo che CLIP proietta immagini e testi in uno spazio vettoriale condiviso tramite embeddings. Possiamo immaginare che con pochi prompts, lo spazio delle proiezioni testuali sia molto limitato e quindi le immagini vengano descritte lungo poche "direzioni semantiche". Aumentando il numero e la diversità dei prompts, aumentiamo il numero di queste direzioni. E speriamo di ottenere una maggiore capacità di distinguere tra classi differenti. Questo perché, aumentando numero e vocabolario dei prompts, aumenta la probabilità che ogni classe del dataset sia rilevante per almeno alcuni di questi prompts, migliorando quindi la qualità delle features per il classificatore lineare.

Stabilito questo, abbiamo ripetuto l'esperimento di rappresentazione dei logits in modalità class-agnostic con un set progressivo di prompts di cardinalità 99, 205, 415. Ogni prompt ha il vincolo di non corrispondere esattamente alle classi reali di CIFAR-100, mantenendo descrittività e semantica visiva, e diversificandosi per

soggetto, forma, composizione, posizioni, ecc. I risultati sperimentali ottenuti sono descritti nella tabella seguente.

Set di Prompt	**Entropia Media**	**Accuratezza SVM**
32	3.4656	29.30%
99	4.5950	45.45%
205	5.3299	54.59%
415	6.0282	62.86%

Table 4.2: Risultati sperimentali su CIFAR-100 utilizzando logits di similarità immagine-testo ottenuti da CLIP con diversi set di prompt generici.

Osservando il netto aumento di accuratezza (addirittura superiore al risultato zero-shot su CIFAR-100 quando il set di prompt ha cardinalità 415), possiamo pensare che i vettori di logits siano diventati più ricchi e meno ridondanti. Di conseguenza, abbiamo features più discriminative e classi simili come "*truck*" vs "*car*" potrebbero iniziare ad avere proiezioni più distinguibili.

Si osserva poi che il miglioramento tra i diversi set di prompts non è lineare, come atteso in sistemi saturabili. Infatti, tra 99 $\longrightarrow$ 205 prompts c'è un incremento di accuratezza di +9.14%, mentre tra 205 $\longrightarrow$ 415 di +8.2%. Ciò significa che il modello beneficia dell'espansione semantica, ma comincia a saturarsi, ovvero i nuovi prompts aggiungono informazioni sempre più marginali o già correlate a quelle esistenti.

4.3 High-quality image dataset

Abbiamo già evidenziato come uno dei limiti che portano a scarse performance nell'esperimento class-agnostic baseline sia la bassa qualità delle immagini di CIFAR-100. Questo comporta un input molto "compresso" per CLIP, con conseguente perdita della maggior parte delle caratteristiche visive rilevanti.

Per questo motivo abbiamo testato lo stesso esperimento iniziale in modalità class-agnostic, utilizzando dataset a risoluzione più alta. In particolare, ci siamo basati su Tiny ImageNet a risoluzione 64×64 (dal quale abbiamo filtrato 100 delle 200 classi generiche fornite) e su Food101 a risoluzione 512×512 (101 classi alimentari come "*apple_pie*", "*beef_carpaccio*", "*churros*"). Per il dataset Food101 abbiamo ripetuto l'esperimento sia con i 32 prompts generici basati semanticamente sulle classi di CIFAR, sia con 32 nuovi prompt specifici del dominio alimentare, come "*a photo of baked dessert*", "*a photo of a cold drink*", ecc.

I risultati sperimentali ottenuti sono riportati nella seguente tabella.

Prompts	**Dataset**	**Entropia Media**	**Accuratezza SVM**
CIFAR-based	Tiny ImageNet	3.4656	35.62%
CIFAR-based	Food101	3.4656	41.25%
Food-based	Food101	3.4654	62.47%

Table 4.3: Risultati sperimentali class-agnostic utilizzando dataset a risoluzione più alta.

Si nota immediatamente come il passaggio da CIFAR-100 (32×32) a Tiny ImageNet (64×64) porti a embeddings di CLIP più espressive. La tesi secondo cui una qualità migliore delle immagini comporti maggiore informazione nello spazio degli embedding viene rafforzata dal fatto che usando Food101 (512×512) con prompts basati sulle classi di CIFAR si ottiene un'accuratezza superiore al 29.30% (baseline class-agnostic), il che è straordinario! Inoltre, quando si utilizza Food101 con prompts semanticamente allineati, si raggiunge una percentuale del 62.47%, superiore perfino allo zero-shot (61.71%). D'altra parte, sappiamo che CLIP è stato addestrato su un dataset massivo e molto variegato, chiamato *WebImageText* (WIT), che fornisce una qualità media comunque superiore rispetto a quella di CIFAR.

Chapter 5

Clustering

Introduction

Come ultimo esperimento, effettueremo un'analisi di *clustering*, quindi non parliamo più di classificazione supervisionata come nel caso della SVM, ma di analisi non supervisionata. In questo approccio, infatti, non è necessario conoscere a priori quali siano i "cluster", ma è l'algoritmo stesso che deve scoprirli autonomamente. Questa tecnica è utile per individuare pattern naturali nei dati prodotti dalle embeddings di CLIP.

Esistono diversi algoritmi di clustering, basati sulla struttura dei dati e sugli obiettivi dell'analisi. Nel nostro caso lavoriamo con embeddings CLIP, ovvero vettori normalizzati che risiedono in uno spazio a 512-dimensionale, e il nostro obiettivo è ottenere una panoramica su quanto questi siano già semanticamente strutturati, anche senza alcuna supervisione. Abbiamo dunque scelto di utilizzare l'algoritmo partizionale `K-means`, il quale si basa sulla scelta a priori del numero K di cluster da generare. Una volta scelti casualmente K centroidi iniziali, l'algoritmo assegna ogni osservazione al cluster il cui centroide è più vicino e aggiorna i centroidi calcolando la media delle osservazioni assegnate a ciascun cluster.

La metrica di accuratezza del clustering che utilizziamo è la *purity*, definita come

$$\text{Purity} = \frac{1}{N} \sum_{k} \max_{j} |C_k \cap L_j| \in [0, 1]$$

dove C_k rappresenta l'insieme delle immagini nel cluster k, L_j rappresenta l'insieme delle immagini appartenenti alla vera classe j e N è il numero totale di immagini. In altre parole, si tratta di una metrica che misura quanto i cluster trovati siano omogenei rispetto alle etichette reali.

5.1 Baseline and Class-Agnostic Clustering

Partendo dal clustering sulle features estratte dall'esperimento di classificazione *Baseline Image Feature*, raggruppiamo le immagini senza utilizzare alcuna etichetta, valutando successivamente quanto i cluster trovati corrispondano alle classi reali. La pipeline utilizzata prevede l'uso dell'algoritmo `K-Means` sulle features delle immagine già estratte, valutando i cluster ottenuti tramite confronto con le etichette originali, per poi calcolare la *purity*.

La stessa pipeline viene applicata anche al caso *class-agnostic*, dove però ora utilizzeremo i vettori di logits per effettuare il clustering. Testeremo la purity sia quando le features sono estratte utilizzando i 32 prompts generici originali, sia con il set esteso di 415 prompts, dotati di un vocabolario più espressivo e specifico.

```
def purity_score(y_true, y_pred):
    # we define a matrix [key, value] where key is the ID of the cluster and value is the
    # list of the associated true lables to the cluster element
    # example: y_true = ["cat", "dog", "cat", "dog", "dog"]
    #          y_pred = [0, 1, 0, 1, 1]

    # truelabel_matrix {
    #   0: ["cat", "cat"],
    #   1: ["dog", "dog", "dog"]
    # }

    truelabel_matrix = {}
    for true, pred in zip(y_true, y_pred):
        truelabel_matrix.setdefault(pred, []).append(true)

    total = 0
    for cluster in truelabel_matrix.values():
        most_common_label = Counter(cluster).most_common(1)[0][1]
        total += most_common_label
    return total / len(y_true)

# clustering
def cluster_and_evaluate(features, true_labels, n_clusters):
    print(f"\nRunning KMeans with {n_clusters} clusters...")

    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
```

```
    pred_labels = kmeans.fit_predict(features)

    purity = purity_score(true_labels, pred_labels)

    print(f"\nClustering evaluation:")
    print(f"Purity: {purity:.4f}")

    return pred_labels, purity
```

5.2 Clustering Insights

Testando le funzioni descritte in precedenza, abbiamo ottenuto i seguenti risultati di clustering riassunti nella tabella:

Prompt Set	Features	Purity
32	Baseline Image	0.4455
32	Class-Agnostic	0.2976
415	Class-Agnostic	0.3321

Table 5.1: Risultati sperimentali di clustering.

Osserviamo che i risultati sembrano indicare che le embeddings puramente visive prodotte da CLIP siano semanticamente strutturate. Possiamo infatti considerare una purity pari a 0.45 su 100 cluster come un risultato più che discreto, tenendo conto che CIFAR-100 contiene 100 classi facilmente sovrapponibili tra loro, come ad esempio "*truck*" e "*car*". Si può quindi affermare che le image features di CLIP contengano già una struttura semantica sufficientemente utile per discriminare tra le classi.

Nel caso invece della modalità *Class-Agnostic* con soli 32 prompts generici, si osserva un crollo della purity. Questo suggerisce che le features estratte in questa configurazione siano più confuse e meno informative per distinguere tra le classi. I prompts generici, quindi, non aiutano il modello a discriminare categorie simili. Tuttavia, aumentando il numero di prompts a 415 e variandone maggiormente il vocabolario, si fornisce a CLIP un contesto più ampio per raffinare le embeddings, con un conseguente miglioramento della purity.

General Conclusion

Dai risultati degli esperimenti condotti, si osserva che CLIP produce embeddings visive semanticamente ricche, che permettono elevate prestazioni sia in contesti di classificazione supervisionata che in zero-shot.

Tuttavia, abbiamo riscontrato che, in un contesto class-agnostic, l'efficacia di tali embeddings dipende fortemente dalla qualità linguistica, dalla quantità e dalla coerenza semantica dei prompts utilizzati, nonchè dalla qualità di risoluzione delle immagini dei dataset. Miglioramenti significativi sono stati inoltre ottenuti tramite tecniche di temperature scaling e con insiemi di prompts ampi e ben progettati.

Anche l'analisi di clustering conferma che le embeddings puramente visive sono intrinsecamente più organizzate semanticamente rispetto alle features ottenute tramite prompts generici. Tutto ciò può far luce su quanto profondamente CLIP abbia effettivamente "compreso" la semantica visiva durante il suo pretraining su milioni di immagini raccolte dal web.

Bibliography

[1] OpenAI, "Clip: Contrastive language–image pre-training," https://github.com/openai/CLIP, 2021.

[2] M. Mistretta, A. Baldrati, L. Agnolucci, M. Bertini, and A. D. Bagdanov, "Cross the gap: Exposing the intra-modal misalignment in clip via modality inversion," *arXiv:2502.04263*, Feb. 2025, accepted for publication at ICLR 2025. [Online]. Available: https://arxiv.org/abs/2502.04263

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[4] D. Balanya *et al.*, "Adaptive temperature scaling for confidence calibration in deep learning," *arXiv preprint arXiv:2206.XXXX*, 2022.

[5] Q. Tan, Z. He, X. Wang, S. Pan, L. Li, and X. Wang, "CASVM: An Efficient Deep Learning Image Classification Method Combined with SVM," *Applied Sciences*, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/22/11690

[6] K. Singh, R. Agarwal, and C. Jawahar, "Enhancing fine-grained classification for low resolution images," *arXiv:2105.00241*, 2021. [Online]. Available: https://arxiv.org/abs/2105.00241