Progetto CVandDL: metodi per l' attribute-based ZSL

Mattia Alesi and Marco Proietti

Contributing authors: s1114418@studenti.univpm.it;
s1114163@studenti.univpm.it;

Abstract

In questo progetto proponiamo idee di vario tipo per raggiungere risultati rilevanti nell'ambito dello Zero Shot Learning, basandoci su letteratura degli anni passati e concentrandoci principalmente sul Conventional ZSL e spostandoci eventualmente anche sul Generalized ZSL con un occhio di riguardo sui metodi attribute-based.
Verrà tenuto conto anche dell'impatto ambientale generato da tali esperimenti grazie al tool Code Carbon[1].

1 Introduzione

I modelli basati sul deep learning hanno raggiunto livelli assimilabili all'essere umano sulla classificazione di immagini complesse. Sorgono però alcuni problemi:

- il successo dipende molto dalla **disponibilità di dati** etichettati in fase di training;
- le classi che il modello può riconoscere sono **limitate** a quelle su cui è stato fatto il training;
- in **scenari realistici** questi modelli sono meno utili perché le classi etichettate potrebbero non essere abbastanza.

Dato che non è possibile fare training su tutti le immagini di ogni possibile oggetto si è pensato di far sì che i modelli riconoscano immagini con pochi o nessun esempio disponibile nella fase di training; è proprio questo il concetto di *Zero-Shot Learning* (ZSL).
Si ambisce ad eliminare il limite di avere **dati etichettati** nei sistemi di intelligenza artificiale. Si intende riconoscere oggetti da classi non viste durante la fase di training.

I dati trattati sono categorizzati in:

- Classi viste: si hanno immagini etichettate durante la fase di training (*feature vector*);
- Classi non viste: cioè per le quali non sono disponibili immagini etichettate durante la fase di training.
- Informazioni ausiliarie: descrizioni/attributi semantici/parole per entrambe le classi viste non viste che fanno da ponte tra queste classi.

Nel nostro specifico caso faremo riferimento alla modalità di ZSL basata sui dati di test[1], abbiamo infatti:

- *Conventional ZSL*: le immagini da riconoscere appartengono esclusivamente a classi non viste.
- *Generalized ZSL*: le immagini da riconoscere appartengono sia a classi viste che non viste.

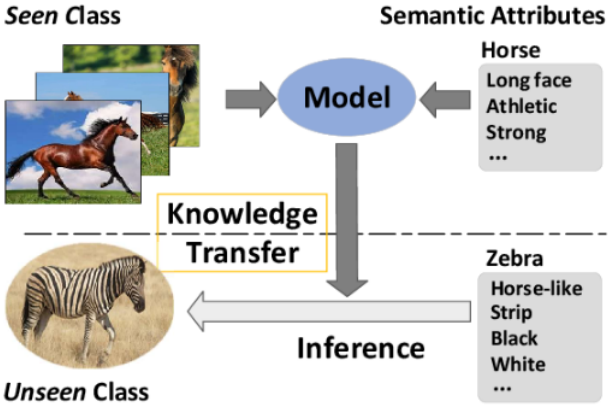

Fig. 1 Schema del funzionamento della ZSL

In più per quanto riguarda gli esperimenti trattati si parla di metodo di classificazione ZSL *attribute-based*:
si va a mappare tra lo spazio degli attributi e le classi e tale mappatura è realizzata manualmente. Esempio: si ha un'immagine x, l'etichetta y pari a "gatto" e ho informazione ausiliari che si trovano in un vettore che consiste di 5 elementi quali *tail, beak, feather, whiskers, furry*. Ogni elemento del vettore è un cosiddetto **attributo** nella forma *one-hot encoded*[2]

[1] L'alternativa è ZSL basato su dati di training (*inductive/transductive*)
[2] Vale 1 se l'attributo è presente nella classe e 0 al contrario.

Nel nostro studio ci focalizziamo sul discorso relativo agli attributi poiché è la parte più impattante nello sviluppo di un modello di ZSL efficace.

Il workflow base dei modelli trattati consiste in: una prima fase di input dei dati, poi la generazione di dati sintetici tramite GAN[3] e infine si fa training di un classificatore con il dataset completo (originale & sintetico).
E' quindi possibile manipolare varie componenti all'interno dei modelli utilizzati ma in particolare, nel nostro lavoro ci siamo concentrati sull'architettura della rete di classificazione e sulla manipolazione degli attributi utilizzati per generare feature sintetiche.

2 State of the art

La ZSL e la GZSL sono relativamente recenti come tematiche, infatti i primi studi risalgono al 2008, con la coniatura del termine nel 2009.
Nel primo esperimento riguardante la materia si fa uso della rappresentazione vettoriale di tipo ESA (*Explicit Semantic Analysis*) ma nei successivi lavori sono stati usati altri tipi di rappresentazione.
Dalla seconda metà del 2010 ci sono stati sviluppi sempre più repentini dato che in generale tutto ciò che riguarda la Computer Vision e il Deep Learning ha subito miglioramenti grazie all'espansione delle memorie, tecniche di gestione dei dati e potenza computazionale.

Come ben spiegato in un articolo del 2020 sulla ricerca e i progressi della zero-shot learning[2] il riconoscimento di classi di oggetti rimane un ostacolo, soprattutto quando le classi hanno pochi o nessun campione per il training.
Viene dimostrato infatti che meno campioni ci sono e più è difficile il riconoscimento.

Molti degli esistenti metodi di ZSL usano apprendimento con recupero cross-modale, in cui l'idea è quella di trasferire conoscenza da classi viste a non viste tramite degli attributi.
Si utilizzano inoltre rappresentazioni semantiche intermedie che sono condivise tramite *dataset* ausiliari che non devono essere adattati al *dataset* obiettivo.

Ci sono diverse tecniche chiave per quanto riguarda la ZSL, divise in:

- *Visual feature extraction*
  Riguarda l'**estrazione di feature visuali** come colori, texture, forme.
  In anni recenti gli algoritmi per questa estrazione sono migliorati ma rimane il problema del *semantic gap*: le feature estratte dalle immagini sono di basso livello e non possono essere collegate a informazioni semantiche di alto livello se non con alto onere computazionale.

[3] Rete Generativa Avversaria

- *Semantic representation* Si articola nelle sottocategorie:
  - *attribute-based*: gli attributi (che possono avere un nome) si rifericono alle caratteristiche specifiche di un oggetto o immagine.
    Un attributo semantico è il vettore di tag di classi annotate manualmente; è infatti il metodo più comune ed efficace per la costruzione di feature semantiche.
    Questo metodo mette le basi per la *attribute-based zero-shot classification* cioè la tecnica su cui si concentra il nostro studio;
  - *word vector-based*: costruisce automaticamente vettori processando documenti di testo non supervisionati;
  - knowledge graph-based: utilizza similitudini tra classi per costruire vettori semantici.

- *Visual-semantic mapping*
  Giunzione tra le feature di immagini e i vettori. Una volta che una mappatura visuale-semantica è stabilita, la somiglianza tra i dati di ogni classe non vista e il prototipo di quella vista può essere calcolata. In questo modo la classe non vista può essere predetta in base alla somiglianza con qualcosa che "già si conosce".

Un modo ulteriore per provare a dare risultati efficaci della ZSL è usare le GAN(*Generative Adversarial Network*), un articolo[3] del 2018 prova a introdurre un nuovo modello di GAN che sintetizza le feature della CNN condizionate su informazioni semantiche a livello classe, offrendo una scorciatoia da un descrittore semantico ad una distribuzione di feature.
Ciò si realizza unendo una Wasserstein GAN a una *classification-loss* per generare abbastanza CNN feature per addestrare classificatori softmax. In questo modo sono stati riscontrati miglioramenti rispetto allo stato dell'arte dell'epoca in tutti i dataset allora disponibili.

Interessante ai fini della nostra ricerca è uno studio[4] del 2019 che tratta specificamente di dare un peso agli attributi usati. Si applica poi una rete CNN per codificare l'informazione dell'immagine; i due layer fully connected sono usati per codificare gli attributi semantici e un modello semantico-visuale viene costruito combinando il ramo visuale. A questo punto i parametri pesati di ogni attributo vengono appresi dall'*objective weightning-method*[4].
Viene fatta quindi la ricerca del *nearest neighbor* per predire il label della categoria di test grazie ai pesi degli attributi.

Arrivando a tempi più recenti, un articolo[5] del 2022 parla della ZSL basata sugli attributi specificamente per la *encrypted traffic classification*.
Viene proposto un nuovo framework per la ZSL attributed-based, composto da:

- un modello GAN-based per migliorare la generalizzazione del classificatore per le classi;
- un modello per imparare a mappare tra il flusso di feature e gli attributi delle classi viste.

[4] I pesi vengono scelti con metodi matematici e non decisi dal programmatore. Si contrappone al *subjective weightning-method*

I risultati dimostrano che il metodo restituisce prestazioni decenti nell'identificazione di classi non viste e all'attuale livello dello stato dell'arte.

Altri studi interessanti riguardano i vari modelli utilizzati per l'ambito e che possono stare a rappresentare lo stato dell'arte:

- [6] che parla in dettaglio della tf-vaegan, che dimostra risultati favorevoli nei sei dataset testati;
- [3] già citato prima, che tratta della CLSWGAN e tratta la WGAN come miglioramento assicurato di GAN
- [7] che parla del modello FREE che porta risultati competitivi soprattutto in ambito GZSL

3 Materials and methods

Per quanto riguarda il nostro progetto il materiale fornitoci consiste in un *repository*(https://github.com/luca-rossi/analogy-based-zsl) contenente un codice suddiviso in più moduli "flessibili", applicabili alla creazione di esperimenti.
Si verificano le prestazioni sia della ZSL che della GZSL su vari tipi di modelli e con molteplici dataset. I modelli utilizzati sono:

- **clswgan**: per provare a dare risultati soddisfacenti alla challenge della ZSL è stata proposta una GAN che sintetizza feature CNN condizionata in base a informazione semantiche di tipo class-level.
- **tfvaegan**: introduce un framework che unisca un modulo *Generative Adversial Network (GAN)* più un *Variational Autoencoders (VAE)*. Il modello è spiegato nel dettaglio in[6]
- **free**: nuovo metodo per la GZSL chiamato *feature refinement* for GZSL.
  Implementa un mapping da semantico a visuale unitamente ad un modello generativo per rifinire le feature visuali delle classi viste e non viste.

Ogni modello raggiunge performance comparabili con lo stato dell'arte ed è stato testato su quattro possibili dataset:
CUB, **AWA2**, **FLO**, **SUN**, ognuno con un numero di attributi e classi diverse.
Si utilizza inoltre il tool Code Carbon per ogni esperimento, in modo da verificare, in aggiunta, l'impatto ambientale di ogni esecuzione.

Gli esperimenti realizzati possono essere riassunti in 3 categorie:

- **Variazione parametri** dei modelli, come per esempio la *hidden size*, *batch size*, opzioni di pooling[5] o metodi di concatenazione diversi.
  Ciò ha l'obiettivo di affinare la configurazione del modello, in modo da ottenere un training più efficace che si rifletta in un accuracy complessiva più elevata.
- Influenzare il modello prendendo la **seconda classe più simile** (e non la prima) per testarne il suo comportamento.

[5] mean, max, first

- **Enfatizzare gli attributi** della classe più simile in modo da "avvicinare" ad essa la classe da predire.

Illustreremo nel prossimo capitolo i risultati.

4 Results

Possiamo affermare che gli esperimenti da noi effettuati raggungono risultati variabili ma complessivamente discreti e presentabili ma che non mostrano miglioramenti rispetto allo stato dell'arte.

Dividiamo i risultati in tre categorie: accuracy nelle classi riconosciute della ZSL, quella della GZSL (divisa per classi viste e non viste).
Nella tabella riportata si possono vedere i vari risultati ottenuti divisi per modelli e dataset. Gli spazi vuoti indicano che non è stato effettuato l'esperimento con tale modello e su tale dataset.

Non potendo fare per ogni esperimento dodici tentavi(quattro dataset per tre modelli), abbiamo puntato a un lavoro sia quantitativo che qualitativo cercando di fare più prove possibili per ogni esperimento ma anche un numero discreto di esperimenti.

Model	Experiment	AWA2			CUB			FLO			SUN		
hline		cZSL	Seen	Unseen	cZSL	Seen	Unseen	cZSL	Seen	Unseen	cZSL	Seen	Unseen
clswgan	Baseline				0.5579	0.5101	0.4077	0.6181	0.7154	0.5088			
	n.1	0.6928	0.6454	0.5003	0.5575	0.4725	0.4317						
	n.2	0.6737	0.6196	0.536	0.5704	0.5024	0.4565	0.6002	0.7381	0.4958	0.5722	0.376	0.4076
	n.3	0.6586	0.6309	0.4345	0.5156	0.4314	0.4052	0.5974	0.7922	0.4559	0.5361	0.3422	0.3868
	n.4	0.6691	0.5623	0.4723									
	n.5	0.6698	0.6347	0.51	0.5514	0.5021	0.4135	0.6142	0.7288	0.5265	0.5542	0.3539	0.4125
free	Baseline	0.6333	0.7711	0.5037									
	n.1												
	n.2	0.5802	0.7226	0.4682	0.582	0.551	0.4864	0.6309	0.7377	0.5937			
	n.3												
	n.4				0.6011	0.5942	0.4727						
	n.5												
tfvaegan	Baseline	0.6144	0.7218	0.4565									
	n.1							0.6602	0.7718	0.5904	0.616	0.3938	0.4472
	n.2	0.5939	0.7678	0.4256	0.5902	0.5844	0.4897	0.6218	0.8052	0.5432	0.6125	0.3922	0.441
	n.3	0.5576	0.7446	0.3934									
	n.4												
	n.5	0.5998	0.7383	0.3596	0.5816	0.5682	0.4765						

Descriviamo nel dettaglio in cosa consiste ogni esperimento:

1. Rappresenta il tentativo di prendere la seconda classe più simile invece che la prima.
2. Il parametro hidden size è stato ridotto da 4096 a 2048
3. Si seleziona un pooling type di tipo *first*
4. similmente all'esperimento precedente ma questa volta con *max*
5. Il batch size è stato raddoppiato a 64

Come anticipato nella sezione 3, nella terza categoria di esperimenti, un risultato particolare è stato trovato andando a moltiplicare gli attributi della classe più simile: le metriche del modello forniscono valori esattamente identici alla versione senza il cambiamento (questo esperimento non è stato riportato nella tabella per ridondanza). Dalla tabella si può notare che i risultati forniscono risultati simili e non troppo distanti tra loro a parte qualche caso particolarmente negativo.
Interessante notare come a seconda del modello e dataset usati per lo stesso esperimento si possano avere differenze sostanziali.

4.1 Code Carbon

Riportiamo a questo punto una comparazione quantitativa dello sforzo computazionale richiesto per l'esecuzione dei modelli, utilizzando il tool *Code Carbon*[1].
Lo strumento è in grado di misurare l'impatto ambientale dovuto all'addestramento della GAN, analizzando le caratteristiche dell'elaboratore che esegue il codice; tema fondamentale in questo momento storico.

Table 1 Tabella emissioni code carbon

Modello	Duration [sec]	Emissions [g CO2]	Energy Consumed [Wh]
Clswgan	2077.580232	0.1194004594	50.22695734
Tfvaegan	5157.477629	20.90995862	150.702596
Free	3778.184869	48.39130507	99.04195522

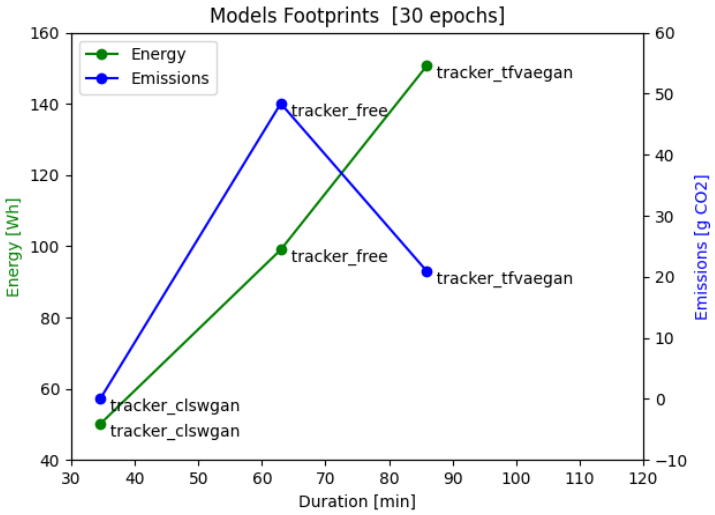

Fig. 2 Consumi/emissioni dei 3 modelli utilizzati

5 Future Developments

Una possibile espansione futura può essere l'applicazione di un feedback al modulo *Generator* contenuto in *modules/models.py*. In particolare, nel generatore si va a manipolare un vettore di rumore e uno di attributi per poi ottenere in uscita un vettore di *feature*.
L'hidden layer può eventualmente prendere un vettore di feedback con i relativi pesi associati in modo da provare a migliorare la qualità dei dati sintetici.
Il tutto coinvolge anche il modulo decoder poiché interagisce con il vettore di feedback. Questo potrebbe incrementare anche la precisione finale del modello e le predizioni in fase di test. Riteniamo questa strategia la più interessante e la tecnica che merita più approfondimenti.

Altro esperimento può essere quello di sfruttare combinazioni lineari per fare condizionamento più accurato: ad esempio si potrebbe riconoscere una zebra condiziono la classe di un cavallo (classe più simile) ma non solo: si può procedere anche per differenza con un'altra classe che ha attributi mancanti nella prima.

Un ulteriore sviluppo è quello di cercare attributi salienti per distinguere una classe dall'altra e poi condizionare il generatore per far riconoscere al modello più classi non viste.
Nel nostro lavoro abbiamo implementato una tecnica per trovare gli attributi più salienti ma le applicazioni richiedono altro lavoro per essere sviluppate.

References

[1] https://codecarbon.io/

[2] Xiaohong Sun, H.S. Jinan Gu: Research progress of zero-shot learning. Springer Nature, 3600–3611 (2020)

[3] Yongqin Xian, B.S. Tobias Lorenz, Akata, Z.: Feature generating networks for zero-shot learning

[4] Wenbai Chen, C.L. Xiangfeng Chem, Wu, H., Li, D.: Zero-shot image classification method based on attribute weighting (2020)

[5] Ying Hu, W.C. Guang Cheng, Jiang, B.: Attribute-based zero-shot learning for encrypted traffic classification (2022)

[6] Sanath Narayan, F.S.K. Akshita Gupta, Snoek, C.G.M., Shao, L.: Latent embedding feedback and discriminative features for zero-shot classification (2020)

[7] Shiming Chen, B.X. Wenjie Wang, Peng, Q., You, X., Zheng, F., Shao, L.: Feature refinements for gzsl (2020)