Sound Event Detection con la tecnica del "few-shot learning"

Matteo Orlandini e Jacopo Pagliuca

Università Politecnica delle Marche

21 luglio 2021

Contenuti

Sound Event Detection

Obiettivo

Individuazione di eventi sonori percettivamente simili all'interno di una registrazione.

Esempi di applicazioni:

- rilevazione di particolari suoni nella musica
- rimozione di parole di riempimento nei podcast

Approccio proposto

Modelli classici di deep-learning:

- grande mole di dati per effettuare il training
- la rete è capace di riconoscere solo eventi sonori su cui è stata allenata.

Approccio *few-shot*

- la rete assumerà la capacità di riconoscere la somiglianza fra due suoni che sta analizzando
- la rete ha la capacità di riconoscere una parola non vista in fase di training

Few-shot learning

L'algoritmo di few-shot learning non necessita di un dataset numeroso per il training.

Meta-learning

L'apprendimento supervisionato tradizionale chiede al modello di riconoscere i dati con cui la rete è stata allenata. Diversamente, l'obiettivo del meta apprendimento è *imparare come imparare* a classificare i dati durante il training e generalizzare quanto imparato durante il test.

In ogni episodio di training della rete, le parole da riconoscere sono diverse, generalizzando il più possibile l'evento e allenando la capacità della rete di riconoscere la somiglianza tra le parole, ma non le parole stesse.

Classificazione C-way K-shot

I modelli di few-shot learning sono allenati per risolvere il compito di classificazione *C-way K-shot*, dove C è il numero fisso di classi tra cui discriminare e K è il numero di esempi forniti per classe.

La figura è un esempio di un task di classificazione 2-way 5-shot, in cui ogni colore rappresenta una classe.

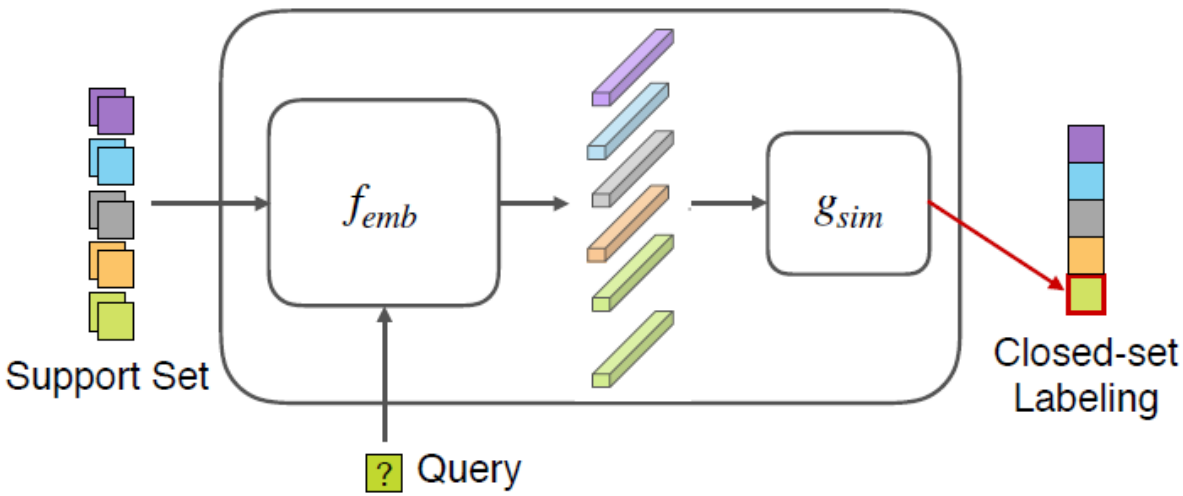

Figura: Modello few-shot learning nel caso 5-*way* 2-*shot*

C-way K-shot

Training

Per la classificazione C-way K-shot, ogni episodio di training usa C classi con K esempi ciascuna. Questi sono chiamati *support set* e vengono utilizzati per apprendere come risolvere il task. Inoltre, esistono ulteriori esempi delle stesse classi, note come *query set*, utilizzate per valutare le prestazioni dell'episodio corrente.

Durante il training, la funzione costo valuta le prestazioni sul query set, dato il rispettivo support set.

Test

Durante il test, utilizziamo un insieme di immagini completamente diverse e valutiamo le prestazioni sul query set, dato il support set.

Training e test task nel caso 3-way 2-shot

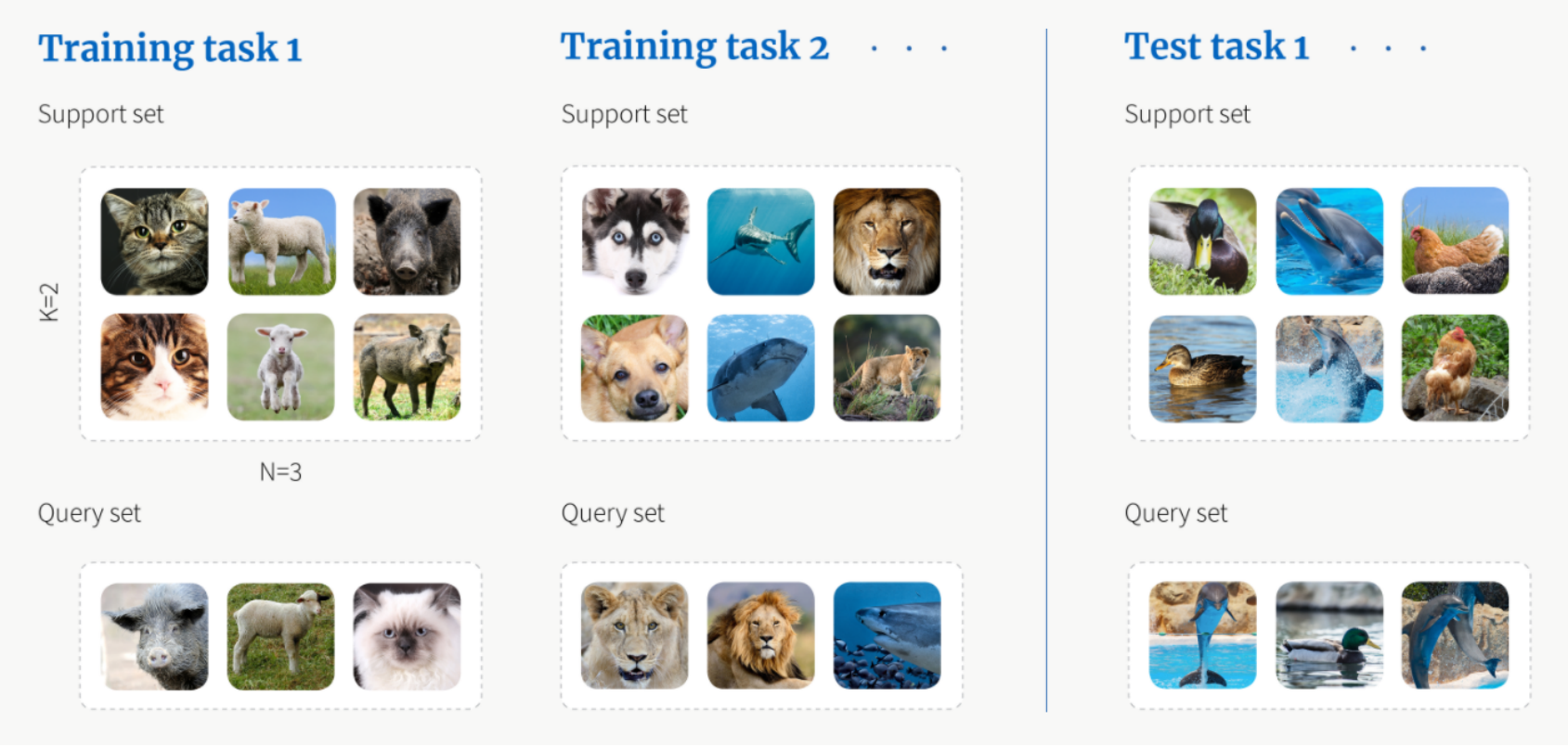

Figura: Esempio 3-way-2-shot.

Metodo proposto per il few-shot sound event detection

Il modello few-shot viene applicato ad un insieme aperto. Inoltre, viene costruito un set di esempi negativi e positivi per incrementare la precisione senza bisogno di ulteriore sforzo umano nell'etichettatura dei dati.

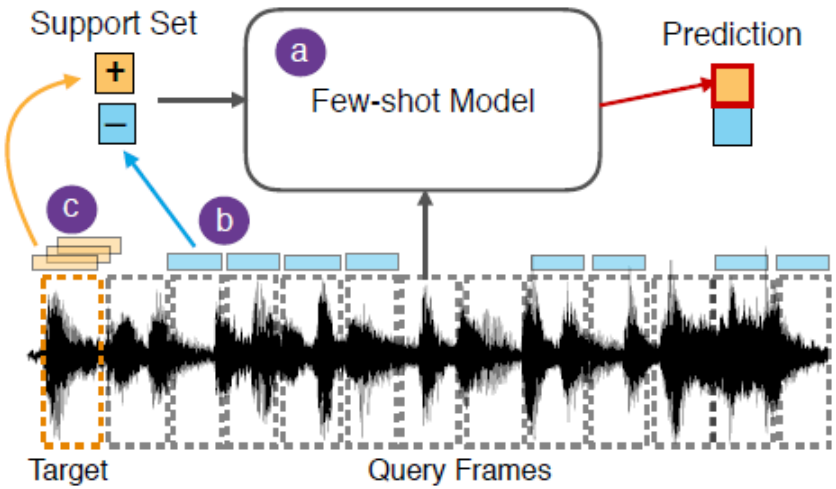

Figura: Metodo proposto per il few-shot sound event detection. (a) Applicazione del modello few-shot, (b) costruzione del set di esempi negativi, in blu, e (c) data augmentation per la generazioni di più esempi positivi, in arancione.

Dataset Spoken Wikipedia Corpora

Spoken Wikipedia Corpora

- nato da lettori volontari di articoli di Wikipedia
- formato da articoli in inglese, tedesco e olandese per gli utenti che non sono in grado o non vogliono leggere la versione scritta dell'articolo.
- i file audio sono raccolti in un corpus, cioè una raccolta ordinata e completa di opere o di autori, allineato nel tempo, rendendolo accessibile per la ricerca.

Ogni cartella del dataset corrisponde ad un articolo di Wikipedia e in ognuna di esse è contenuto un file "aligned.swc" in cui sono salvate tutte le informazioni sull'audio come il nome del lettore, le parole e i relativi timestamp.

Il dataset Spoken Wikipedia Corpora contiene un totale di 1340 audio di diversi lettori, ma nel progetto vengono presi solo gli audio che contengono annotazioni a livello di parola.

Partizionamento dei lettori

Vengono presi 208 lettori e partizionati con un rapporto 138 : 15 : 30 tra lettori di training, validation e test.

Si effettua un controllo sul nome del lettore perché può capitare che questo venga salvato nel file "aligned.swc" in modi diversi. Ad esempio, in alcuni file si può trovare ":en:user:alexkillby|alexkillby", mentre in altri solo "alexkillby" oppure si può trovare "user:popularoutcast", mentre in altri solo "popularoutcast".

Per ogni cartella del dataset SWC si esegue il parsing del file XML, iterando l'albero fino al tag "n", cioè fino al tag che contiene la normalizzazione della parola. Se è presente la chiave "start" (o, in modo equivalente, "end"), la parola viene salvata.

Target words

Le target words sono le parole che si ripetono almeno 10 volte nel testo.

Vengono salvati due json all'interno di ogni cartella del dataset:

- "word_count.json" contiene tutte le parole pronunciate almeno 10 volte
- "target_words.json" contiene al massimo 10 target words.

Se ci sono più di 10 parole che soddisfano la condizione di target word, se ne prendono solo 10 scelte in modo random. In questo modo, si evita di scegliere parole molto comuni o molto rare.

Calcolo dello spettrogramma delle parole

Per ogni istanza delle parole, prendiamo una finestra di mezzo secondo centrata sulla parola, calcoliamo lo spettrogramma mel da 128 bin e lo portiamo scala logaritmica.

Lo spettrogramma rappresenta l'intensità di un suono in funzione del tempo e della frequenza.
La scala mel è una scala di percezione dell'altezza (pitch) di un suono. La relazione tra la scala mel e quella comunemente usata è rappresentata dall'uguaglianza tra 1000 Mel e 1000 Hz all'intensità di 40 dB.

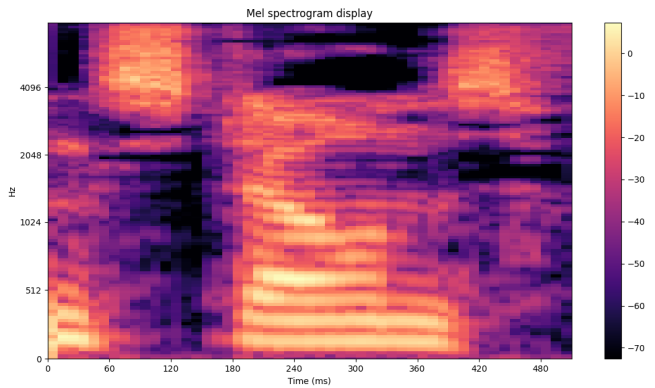

Figura: Spettrogramma di mezzo secondo centrato sulla parola "stones" pronunciata nell'audio "I can't get no satisfaction"

Protypical Network

L'approccio si basa sull'idea che esista un embedding in cui i punti delle istanze di una classe si raggruppano attorno a una singola rappresentazione per ogni classe, chiamata *prototipo*. Nella classificazione few-shot per ogni episodio viene fornito un support set di N esempi etichettati $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ dove $\mathbf{x}_i \in \mathbb{R}^D$ rappresenta il vettore D-dimensionale della feature (nel nostro caso spettrogrammi di dimensione 128×51) e $y_i \in \{1, \ldots, K\}$ la rispettiva label. S_k denota il set di esempi etichettati con la classe k.

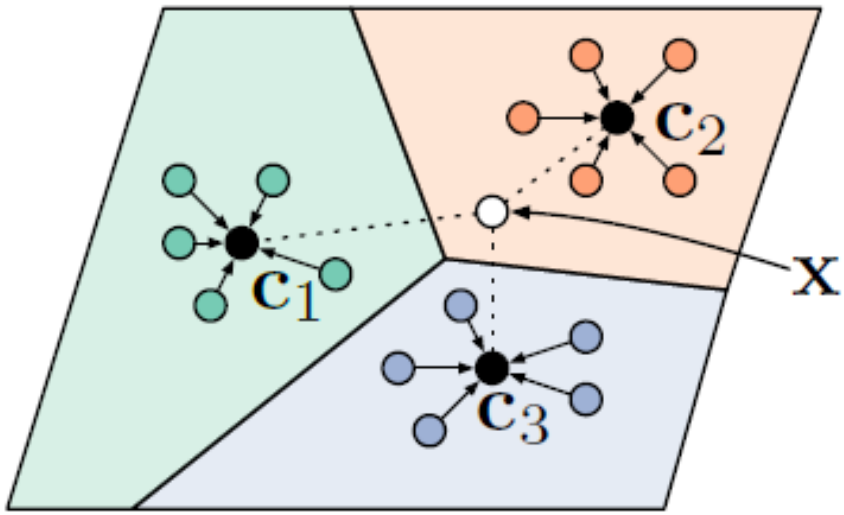

Figura: I prototipi few-shot $\mathbf{c}_k$ sono calcolati come la media degli embedding del support set per ogni classe. I punti degli embedding delle query sono classificati facendo il softmax sulle distanze del prototipo delle classi: $p_\phi(y = k|\mathbf{x}) \propto \exp\left(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)\right)$.

La rete Prototypical calcola una rappresentazione M-dimensionale $\mathbf{c}_k$, o *prototipo*, di ogni classe tramite una funzione di embedding $f_\phi : \mathbb{R}^D \to \mathbb{R}^M$ con parametri da allenare ϕ. La funzione di embedding è rappresentata da una rete convoluzionale. Ogni prototipo è calcolato come la media tra gli embedding delle istanze della stessa classe.

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i) \tag{1}$$

Data una funzione distanza $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, la rete Prototypical calcola la relazione di una query $\mathbf{x}$ rispetto ai prototipi tramite la funzione softmax delle distanze prese con segno negativo.

$$p_\phi(y = k|\mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum'_k \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}'_k))} \tag{2}$$

Il processo di training avviene minimizzando il negativo del logaritmo della probabilità

$$J(\phi) = -\log\left(p_\phi(y = k|\mathbf{x})\right) \tag{3}$$

considerando la distanza fra query e il prototipo della sua classe. Gli episodi di training sono formati campionando C classi di parole da un lettore e selezionando per ognuna casualmente K istanze e Q query.

Architettura della rete

La CNN che funge da rete di embedding per la Prototypical Network è costituita da 4 strati convoluzionali. Il primo di essi ha come ingresso un singolo canale e come uscita 64 mentre i tre successivi traspongono 64 canali in altri 64.
Al termine dei blocchi viene effettuato il reshape dell'uscita schiacciandola in una sola dimensione, `x.view(x.size(0),-1)`, in modo da avere una matrice le cui righe rappresentano gli embedding vettoriali.

```python
import torch.nn as nn

def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)

def conv_block(in_channels,out_channels):

    return nn.Sequential(
        nn.Conv2d(in_channels,out_channels,3,padding=1),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(),
        nn.MaxPool2d(2)
    )

class Protonet(nn.Module):
    def __init__(self):
        super(Protonet,self).__init__()
        self.encoder = nn.Sequential(
            conv_block(1,64),
            conv_block(64,64),
            conv_block(64,64),
            conv_block(64,64)
        )
    def forward(self,x):
        (num_samples,mel_bins, seq_len) = x.shape
        x = x.view(-1,1,mel_bins,seq_len)
        x = self.encoder(x)
        return x.view(x.size(0),-1)
```

Listing 1: protonet.py

Blocco convoluzionale

Ogni blocco convoluzionale ha 4 fasi:

- Convoluzione bidimensionale con un kernel 3x3 i cui parametri vengono aggiornati ad ogni backward. Viene effettuato un padding di zeri ai bordi dell'ingresso.
- Batch normalization: un metodo utilizzato per rendere le reti neurali artificiali più veloci e stabili attraverso la normalizzazione degli input dei livelli con re-centering and re-scaling.
- ReLu: il rettificatore è una funzione di attivazione definita come la parte positiva del suo argomento. $f(x) = \max(0, x)$
- Max-pooling: metodo per ridurre la dimensione di un'immagine, suddividendola in blocchi e tenendo solo quello col valore più alto.

Training

A partire dalla lista di lettori di training/validation, seleziona quelli che hanno almeno un numero di parole diverse pari a C e li suddivide tra training e validation seguendo lo stesso rapporto usato dal paper. Il numero totale di iterazioni è 60000. Per ogni episodio:

1. Si campiona un lettore casuale e si selezionano C parole pronunciate da esso.
2. Si calcola la loss relativa all'episodio
3. Tramite la backpropagation si aggiornano i parametri della rete in base alla loss.

Per il calcolo della loss le operazioni sono:

1. Entrambi i set vengono concatenati, risultando in un tensore $(C \cdot (Q + K)) \times 128 \times 51$, per poter essere passate al modello che calcolerà gli embedding per ogni istanza.
2. Gli elementi del support set che fanno parte della stessa classe andranno a costituire i prototipi tramite una media dei loro valori.
3. Vengono successivamente calcolate le distanze rispetto ai prototipi degli embedding di ogni classe ottenendo una matrice di dimensioni $(C \cdot Q) \times C$. L'i-esima riga e la j-esima colonna rappresentano la distanza euclidea tra gli embedding della i-esima query $(i = 1, \ldots, C \cdot Q)$ e quelli del j-esimo $(j = 1, \ldots, C)$ prototipo.
4. Per ogni query viene poi usata la funzione di attivazione softmax, i cui argomenti sono le distanze tra la query e i prototipi delle varie classi. La funzione loss da minimizzare è il logaritmo del softmax preso con segno negativo.

Validation

Ogni 5000 episodi del training vengono effettuati 1000 episodi di validation. Come nel training vengono ricavati support set e query set, ma in questo caso dai lettori di validation. Per questi vengono calcolati allo stesso modo loss e accuracy e vengono salvati. In questo caso però i parametri del modello non vengono aggiornati. Questo processo è necessario per verificare l'effettività del modello su ingressi mai visti in fase di training.
Al termine dei 1000 episodi di validation, se la media dell'accuracy è migliore di quella calcolata nel passo precedente il modello viene salvato e sovrascritto, altrimenti viene ignorato.

Test

Relation Network

La rete Relation ha lo scopo di associare due istanze alla volta per determinare la loro similarità. Questo viene effettuato concatenando gli embedding di più istanze in un unico elemento che sarà dato in ingresso a una rete decisionale i cui parametri saranno aggiornati in modo che una concatenazione di elementi simili restituisca un risultato vicino a 1. La Relation Network è costituita da due moduli: un modulo di *embedding* f_φ (equivalente a quello nella Prototypical) e un modulo di *relation* g_ϕ. Le istanze x_i del query set $\mathcal{Q}$ e quelle x_j del support set $\mathcal{S}$ vengono date in ingresso al modulo di embedding producendo dei vettori (feature maps) $f_\varphi(x_i)$ e $f_\varphi(x_j)$. Questi ultimi vengono poi dati all'operatore $\mathcal{C}(\cdot,\cdot)$ che ne fa la concatenazione: $\mathcal{C}(f_\varphi(x_i), f_\varphi(x_j))$.

Le feature map concatenate passano poi attraverso il modulo di decisione che restituisce uno scalare da 0 a 1, il quale rappresenta la somiglianza tra x_i e x_j. Per il caso C-way one-shot, viene concatenata la query con le istanze delle C classi producendo C punteggi di somiglianza.

$$r_{i,j} = g_\phi(\mathcal{C}(f_\varphi(x_i), f_\varphi(x_j))), \qquad i = 1, 2, \ldots, C \tag{4}$$

Nel caso C-way K-shot invece, la query viene concatenata con la somma elemento per elemento degli embedding di ogni istanza delle classi. Quindi, in entrambi i casi i confronti $r_{i,j}$ sono C per ogni query.
Per allenare il modello viene usato l'errore quadratico medio (MSE) in modo che l'uscita del modulo di decisione produca 1 se i vettori concatenati sono della stessa classe e 0 altrimenti.

Relation Network

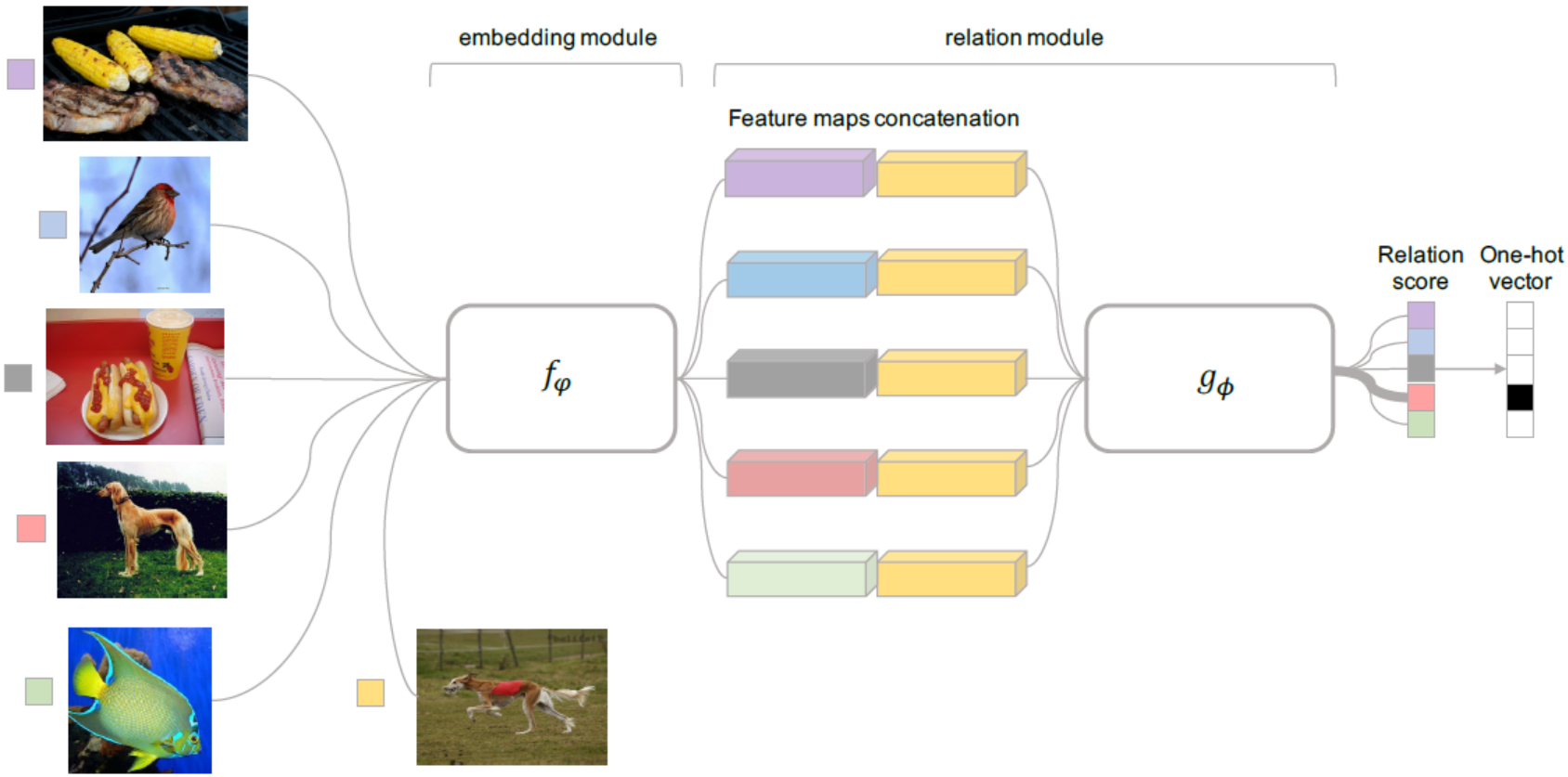

Figura: Architettura della Relation Network nel caso 5-*way* 1-*shot* con un esempio di query.

Training

Validation

Test

Matrice di confusione

Nella fase di test, la rete elabora una predizione dell'output a partire da un ingresso noto che poi viene confrontata con il valore effettivo. Nel nostro caso è presente un positive set e un negative set, si tratta quindi di classificazione binaria. Il confronto tra predizione e label può produrre quattro risultati:

- Vero Negativo (TN): il valore reale è negativo e il valore predetto è negativo;
- Vero Positivo (TP): il valore reale è positivo e il valore predetto è positivo;
- Falso Negativo (FN): il valore reale è positivo e il valore predetto è negativo;
- Falso Positivo (FP): il valore reale è negativo e il valore predetto è positivo;

Precision e Recall

Precision

Il parametro Precision rappresenta quanti tra i casi predetti come positivi sono realmente positivi.

Recall

Il parametro Recall indica quanti tra i casi realmente positivi è stato predetto in modo corretto.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

AUPRC

Area under precision-recall curve (AUPRC)

Area sottessa dalla curva precision-recall. Questo valore è molto utile quando i dati sono sbilanciati e, in una classificazione binaria, siamo più interessati al riconoscimento di una classe in particolare.

Per calcolare questo parametro sono state usate le funzioni `precision_recall_curve` e `sklearn.metrics.auc`.

Calcolo AUPRC

1. Consideriamo i valori di probabilità delle query di appartenere alla classe positiva.
2. Ordiniamo il vettore in ordine decrescente.
3. Consideriamo come soglia il valore di probabilità presente al primo elemento e calcoliamo precision e recall confrontando con un vettore che indica la classe corretta.
4. Spostiamo la soglia al valore del secondo elemento e calcoliamo un'altra coppia di parametri.
5. Ripetiamo il passo 3 e 4 per tutti gli elementi.
6. Otteniamo una curva considerando alle ascisse i valori di recall e come ordinate i valori di precision.
7. Calcolando l'area sottesa dalla curva ricaviamo il parametro AUPRC.

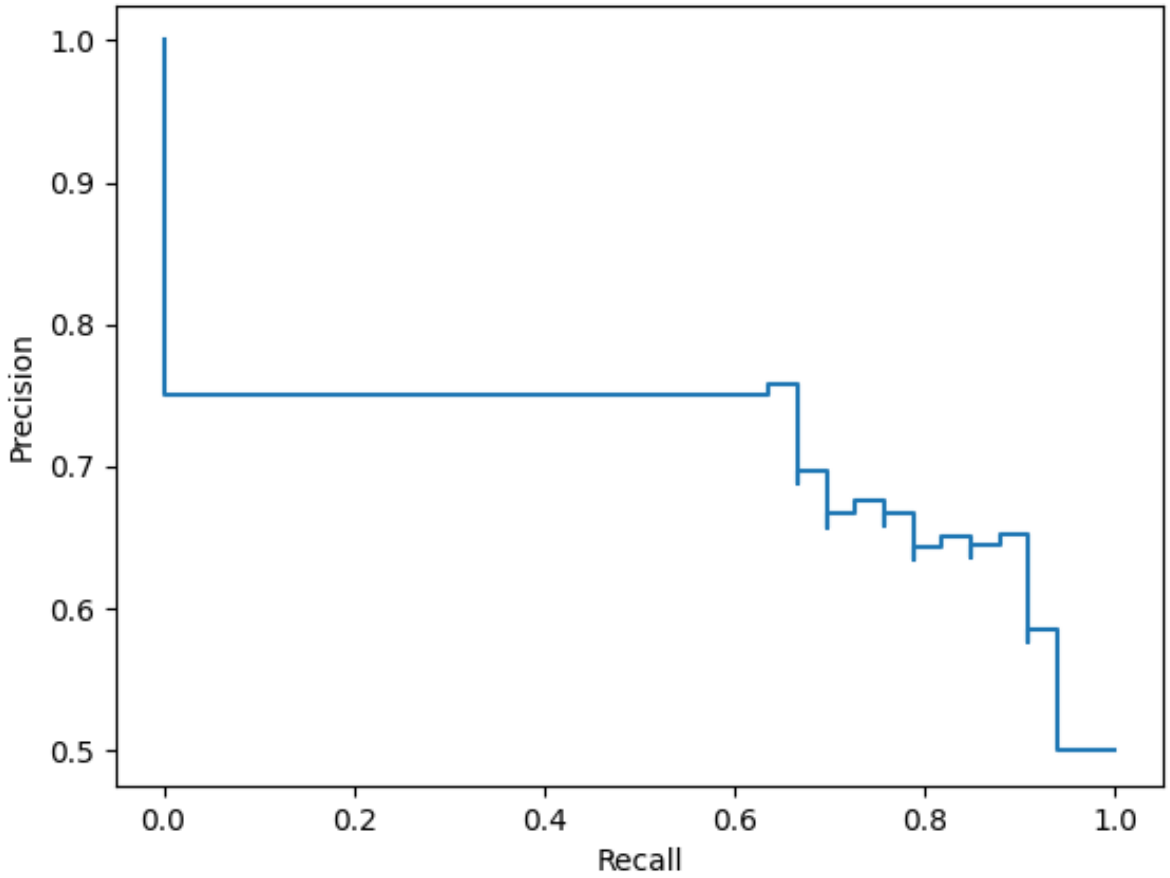

Figura: Curva precision-recall per il test della rete prototypical con $C = 2$, $K = 1$ e $p = 5$.

Risultati

C, K	AUPRC	Deviazione standard
2, 1	0.731	0.069
2, 5	0.721	0.074
5, 1	0.709	0.089
5, 5	0.688	0.062
10, 1	0.728	0.067
10, 5	0.727	0.078
10, 10	0.725	0.059

Tabella: Prototypical network $p = 1$, $n = 10$.

C, K	AUPRC	Deviazione standard
2, 1	0.800	0.056
2, 5	0.791	0.062
5, 1	0.774	0.092
5, 5	0.754	0.075
10, 1	0.787	0.057
10, 5	0.794	0.070
10, 10	0.793	0.050

Tabella: Prototypical network $p = 5$, $n = 10$.

C, K	AUPRC	Deviazione standard
2, 1	0.775	0.069
2, 5	0.609	0.046
5, 1	0.889	0.061
5, 5	0.666	0.047
10, 1	0.898	0.053
10, 5	0.710	0.059
10, 10	0.538	0.015

Tabella: Relation network $p = 1$, $n = 10$.

C, K	AUPRC	Deviazione standard
2, 1	0.596	0.042
2, 5	0.813	0.056
5, 1	0.698	0.050
5, 5	0.945	0.037
10, 1	0.691	0.052
10, 5	0.945	0.036
10, 10	0.845	0.061

Tabella: Relation network $p = 5$, $n = 10$.

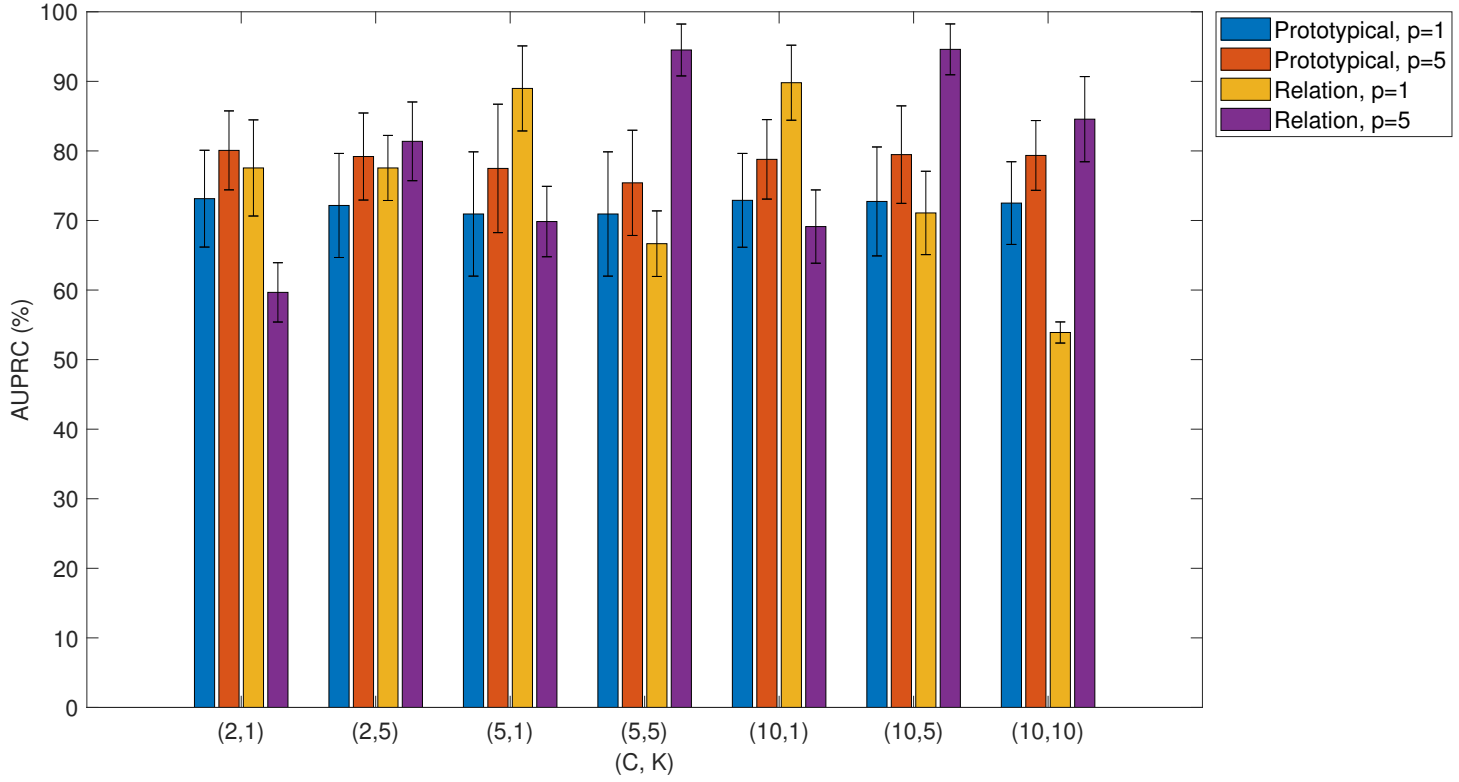

Figura: Risultati ottenuti.

Conclusioni

Si può notare che:

- La rete Prototypical ha un AUPRC più grande con $p = 5$ rispetto a $p = 1$ per ognuno dei modelli allenati.
- Il miglioramento nell'AUPRC è di circa 6-7% passando da $p = 1$ a $p = 5$.
- Per la rete Relation questo comportamento non si presenta.
- Con $p = 1$ si nota che la rete Prototypical ha un risultato migliore per $(C, K) \in \{(10, 5), (10, 10)\}$, mentre nei restanti casi la rete Relation risulta migliore.
- Per $p = 5$ la rete Prototypical ha invece un AUPRC più elevato per $(C, K) \in \{(2, 1), (5, 1), (10, 1)\}$.

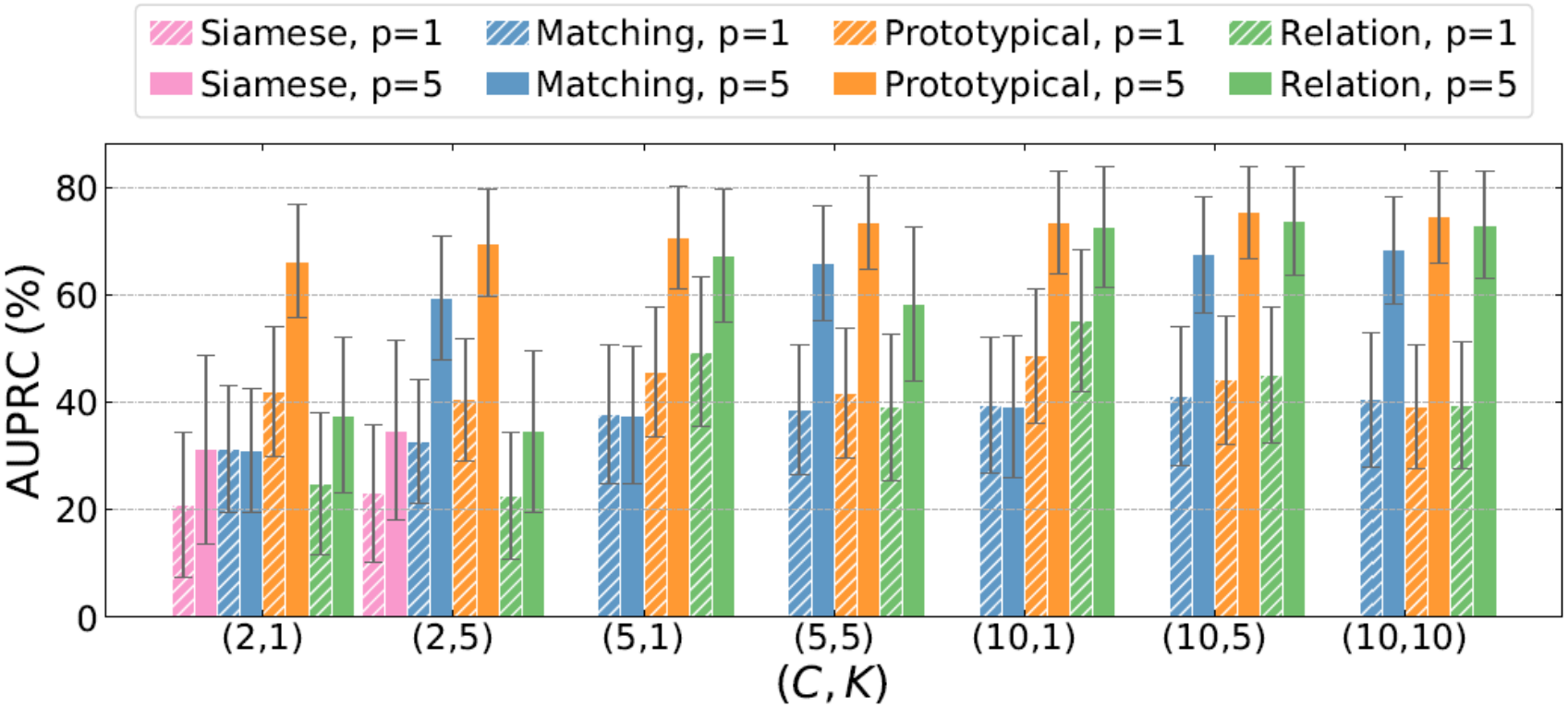

Figura: Risultati proposti nel paper

Confronto

Confrontando i risultati con quelli del paper si può osservare che:

- in entrambi i grafici, per la rete Prototypical con $C = 10$ non si ha un miglioramento significativo dell'AUPRC e che questo tipo di rete si comporta meglio con $p = 5$.
- la rete Relation ha sempre dei risultati peggiori rispetto alla Prototypical, mentre nel nostro caso dipende dai valori di C e K con cui è stata allenata la rete.
- i valori di AUPRC nella nostra implementazione sono generalmente più alti

Possibili cause di divergenza dei risultati:

- il preprocessing del dataset.
- diversa implementazione della rete Relation, la cui funzione g_{sim} non era stata descritta nell'articolo.
- differente criterio per il salvataggio dei modelli durante il training.