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

Matteo Orlandini e Jacopo Pagliuca

Università Politecnica delle Marche

19 luglio 2021

Contenuti

Bluetooth Low Energy

Concetti fondamentali del Bluetooth Low Energy:

- Banda 2.4 GHz
- 40 canali spaziati da 2 MHz
- 37 canali per i dati e 3 per gli advertising
- FDMA, TDMA
- FHSS

Stack protocollare BLE

Sound Event Detection

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

Metodo proposto per il few-shot sound event detection

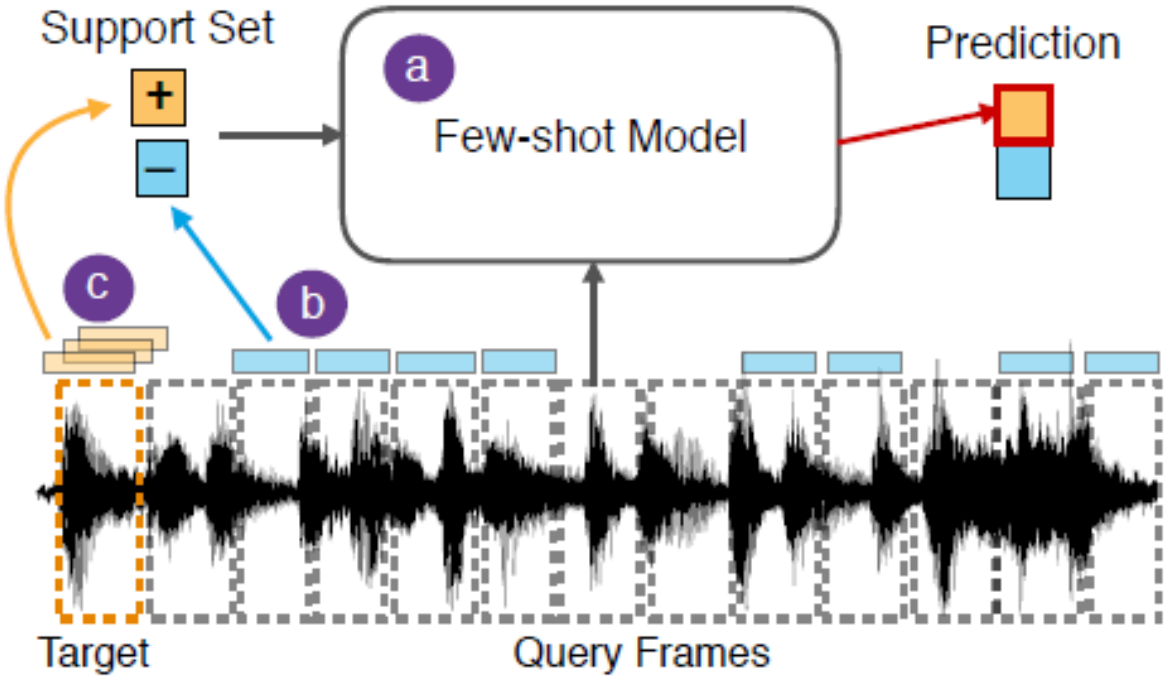

Figura: Metodo proposto per il few-shot sound event detection. (a) Applicazione del modello few-shot, (b) costruzione del set di esempi negativi, in blu, e (c) data augmentation per la generazioni di più esempi positivi, in arancione.

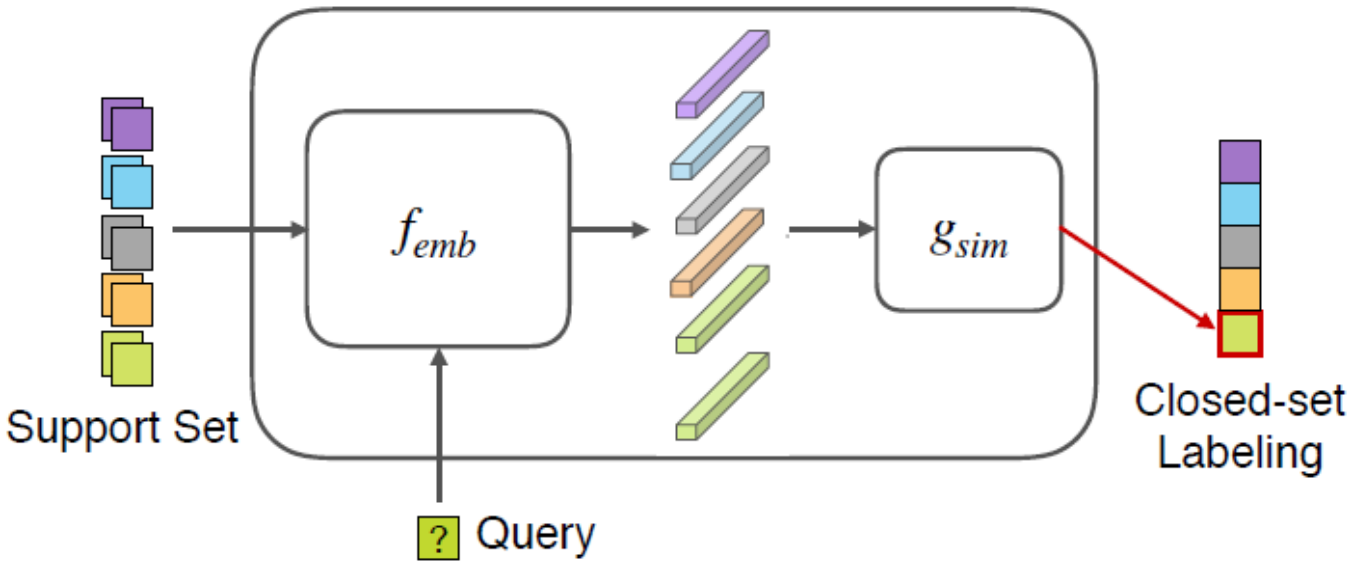

Figura: Modello few-shot learning nel caso 5-*way* 2-*shot*

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 di training e quindi di generalizzare per dati di test non visti in precedenza. Diversamente, l'obiettivo del meta apprendimento è imparare.

Nel quadro del meta-apprendimento, *impariamo come imparare* a classificare in base a una serie di *episodi di training* e valutiamo utilizzando una serie di episodi di test.

C-way K-shot

Ogni episodio per la classificazione C-way K-shot contiene:

- Un support set costituito da K istanze di C classi.
- Un query set con Q istanze per ognuna delle C classi.

Ogni episodio può essere completamente unico; potremmo non vedere mai le classi di un episodio in nessuno degli altri.
Poiché la rete viene sottoposta ad un compito diverso in ogni fase temporale, deve imparare a discriminare le classi di dati in generale, piuttosto che un particolare sottoinsieme di classi.

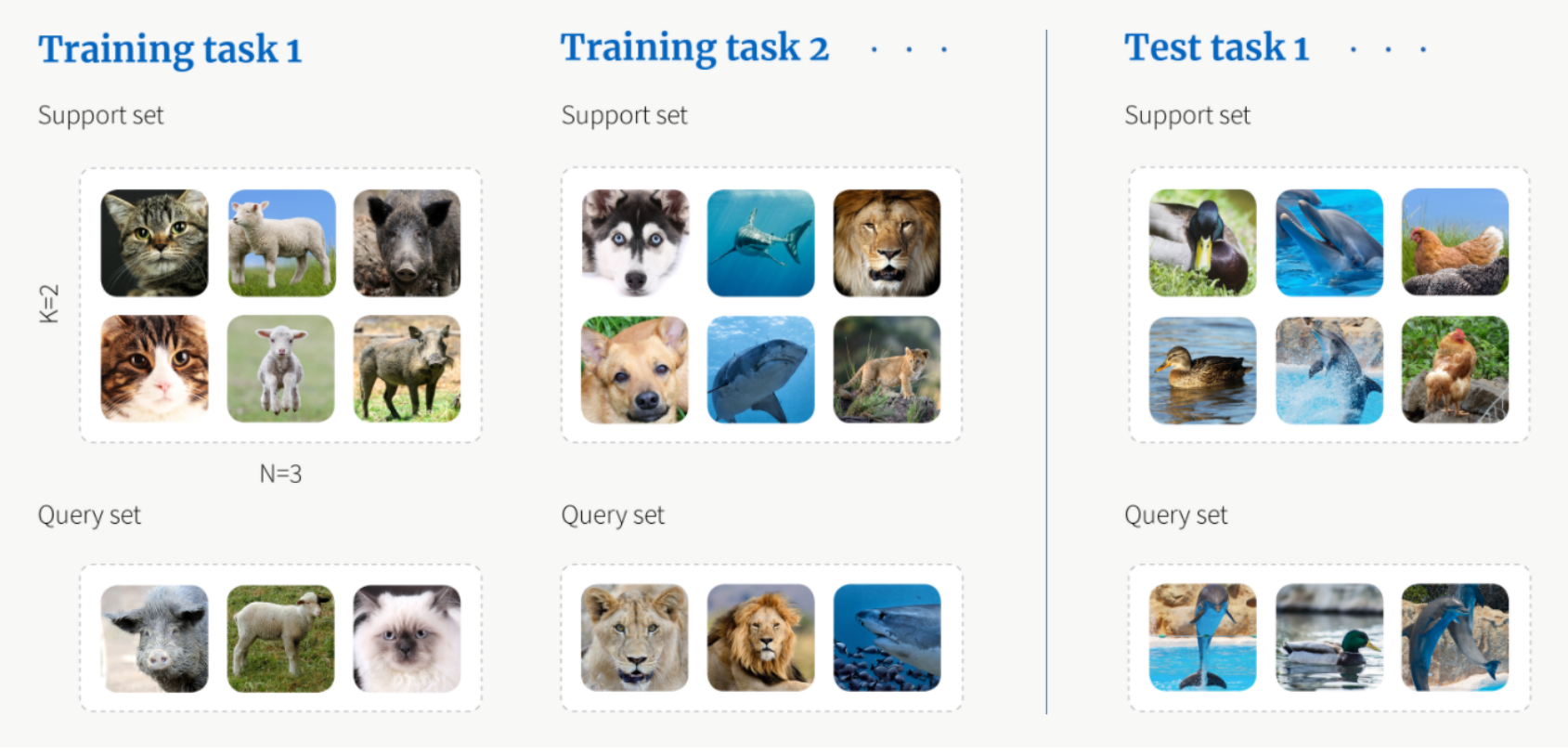

Figura: Esempio episodi

esempio titolo: lettori validi

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.

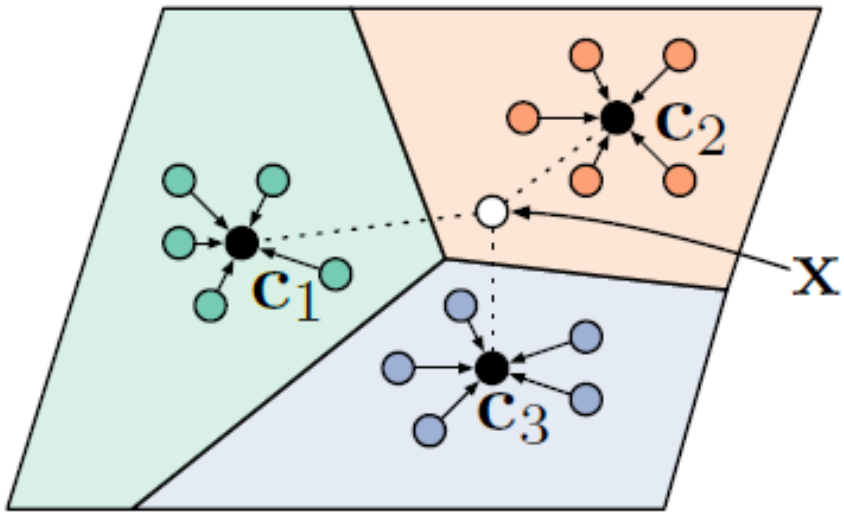

Figura: I prototipi few-shot $\mathbf{c}_k$ sono calcolati come la media degli embedding del support set per ogni classe. I punti degli embedding delle query sono classificati facendo il softmax sulle distanze del prototipo delle classi: $p_\phi(y = k|\mathbf{x}) \propto \exp\left(-d(f_\phi(\mathbf{x}),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

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:

- `nn.Conv2d(in_channels,out_channels,3,padding=1)`: Convoluzione bidimensionale con un kernel 3x3 i cui parametri vengono aggiornati ad ogni backward. Viene effettuato un padding di zeri ai bordi dell'ingresso.
- `nn.BatchNorm2d(out_channels)`: La batch normalization è 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.
- `nn.ReLU()`: il rettificatore è una funzione di attivazione definita come la parte positiva del suo argomento.
  $f(x) = \max(0, x)$
- `nn.MaxPool2d(2)`: Il max-pooling è un metodo per ridurre la dimensione di un'immagine, suddividendola in blocchi e tenendo solo quello col valore più alto.

Training

Il numero totale di iterazioni è 60000; per ogni episodio viene richiamata la funzione `batch_sample`.
Con `loss_out.backward()` viene utilizzata la loss per la retropropagazione che aggiorna i parametri della rete.

La funzione `get_training_validation_readers`, 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 il rapporto usato nel paper di riferimento.

La funzione `batch_sample` campiona a caso uno dei lettori di training e di questo seleziona casualmente C parole. Per ognuna di queste parole si ricava il numero di istanze presenti del dataset e vengono campionate $K + Q$ istanze random della parola. Le $K + Q$ istanze vengono divise in K istanze del support set e Q stanze del query set. La funzione ritorna due tensori `query` e `support`, rispettivamente di dimensioni $C \times Q \times 128 \times 51$ e $C \times K \times 128 \times 51$.

Una volta ottenute, le feature del support set e query set vengono date in ingresso alla funzione `loss`. Entrambi i set vengono ridimensionati, rispettivamente da $C \times K \times 128 \times 51$ a $(C \cdot K) \times 128 \times 51$ e da $C \times Q \times 128 \times 51$ a $(C \cdot Q) \times 128 \times 51$. Vengono poi concatenate, 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. Gli elementi del support set che fanno parte della stessa classe andranno a costituire i prototipi tramite una media dei loro valori.

Vengono successivamente calcolate le distanze rispetto ai prototipi degli embedding di ogni classe tramite la funzione `euclidean_dist` che ritorna 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. 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. Se la query è vicina al prototipo, la funzione softmax tende a 1 e di conseguenza il suo logaritmo a 0, dunque la loss tende a 0, come dovrebbe appunto risultare. Nel caso in cui, invece, la query è molto lontata dal prototipo, la funzione softmax tende a 0 e il suo logaritmo a $-\infty$. In questo caso, la loss, definita come il logaritmo del softmax preso con segno negativo, tende a $+\infty$, come ci si aspetta.

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 con la funzione `loss` 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. In questo modo si salva il modello migliore durante il training, cercando di evitare sia underfitting sia overfitting.

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

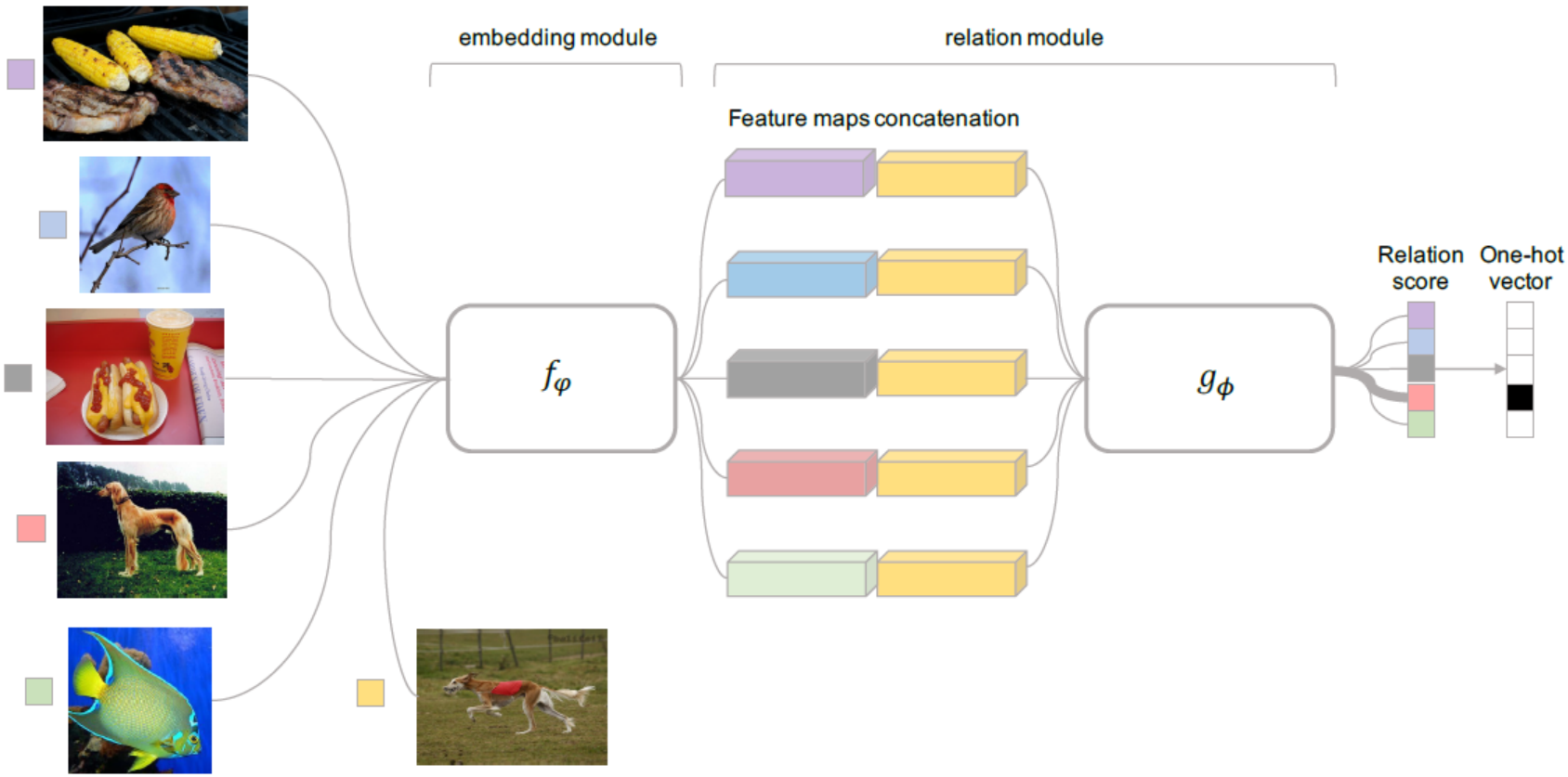

Figura: Architettura della Relation Network nel caso 5-*way* 1-*shot* con un esempio di query.

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.

Risultati

Conclusioni