Image classification basato su reti neurali coinvoluzionali

Lorenzo Meroni

Univerisita' degli studi Milano Bicocca, Milano
l.meroni16@campus.unimib.it

Abstract. Intel image classification `https://www.kaggle.com/datasets/puneet6060/intel-image-classification` è un dataset presente sulla piattaforma Kaggle che raccoglie immagini in formato *jpg* appartenenti a 6 differenti categorie.
Lo scopo dello studio è prevedere, in base a specifici modelli basati sulle reti neurali, addestrati su un training set a quale classe di immagine appartengano le osservazioni presenti nel test set.
Il miglior risultato che coincide con il valore più alto di accuracy è stato ottenuto mediante una rete neurale coinvoluzionale (CNN).

Keywords: Intel image classification · previsione · reti neurali coinvoluzionali

1 Introduzione

Il problema preso esame è un problema di classificazione multiclasse di immagini. Quando si affronta un problema di classificazione multiclasse o multinomiale si tratta di classificare correttamente le istanze in una di tre o più classi, utilizzando come mezzi modelli basati su reti neurali, cuore centrale del corso.
I dati originariamente scelti per essere training set sono stati suddivisi a loro volta in una parte di training set, costituita da 12000 osservazioni mentre la restante parte costituita da 1146 osservazioni è stata utilizzata per validare i risultati ottenuti sul training set ovvero come simulazione di quanto possa accadere sul test set.
Il modello che è stato scelto è una rete neurale coinvoluzionale con uno strato di *input*, 2 strati di *coinvoluzione* seguiti rispettivamente ognuno da un layer di *pooling* ognuno dei quali ha il compito di eseguire un'aggregazione delle informazioni, generando così feature map di dimensione inferiore, un *layer dropout* per evitare l'overfitting e infine uno strato *fully connected*.
Per l'ultimo strato è stata scelta una funzione di attivazione di tipo softmax, infatti come già anticipato si tratta di un problema di classificazione multiclasse dunque l'output deve essere a sua volta multiclasse pertanto la funzione di attivazione softmax è perfetta siccome restituisce la probabilità per ogni classe.
Il risultato ottenuto è un'accuracy sul validation set dello 0.83 circa.

2 Materiali e metodi

2.1 Materiali

Il dataset preso in esame è un dataset di immagini in formato *jpg* presente sulla piattaforma Kaggle chiamato Intel image classification.
Il dataset è già stato diviso in sede di creazione del dataset in traininig set e test set. Il training set è suddiviso al suo interno in altre 6 cartelle per ogni categoria di immagine *buildings, forest, glacier, mountain, sea, street*. Per l'addestramento della rete si è deciso di utilizzare 2000 immagini per categoria, di conseguenza 12000 immagini mentre 190 per categoria per la validazione. Qui sotto vengono riportate 6 immagini esemplificative di quello che troviamo in ognuna delle 6 cartelle.

Fig. 1: Esempi di immagini appartenenti alle cartelle di training

Nel dataset in esame sono stati riscontrati alcuni problemi ispezionando le immagini presenti nelle cartelle di training. Sono state infatti trovate immagini non apparteneni alla categoria attesa all'interno di quella cartella, un esempio viene qui sotto riportato:

Fig. 2: Immagine macchina presente nella cartella glacier

L'immagine mostrata è stata trovata nella cartella di training destinata ai ghiacciai, risulta evidente come questa immagine non rappresenti un ghiacciaio.
Abbiamo quindi optato per eliminare le immagini che non rispettavano l'etichetta posta a nome della cartella, queste immagini infatti costituivano un ostacolo nell'apprendimento della nostra rete.

2.2 Metodi

L'analisi si è composta di diverse fasi, prima fase abbiamo preso le prime 2000 immagini per ogni classe presente nella cartella di training e abbiamo creato il nostro training set, lo stesso lo abbiamo fatto per il validation set. Abbiamo creato 2 funzioni, la prima che trasforma le immagini in tensori mentre la seconda che da un'etichetta in base alla classe alla quale appartiene la singola immagine. Una volta applicate le due funzioni abbiamo i nostri dati di training, di test e di validation con le rispettive etichette.
Il passaggio successivo è stato quello di applicare il processo che viene detto di augmentation ovvero una tecnica utilizzata per aumentare la quantità dei dati utilizzati al fine di addestrare le rete.
L'augmentation implica l'aumento artificiale delle dimensioni del training set aggiungendo trasformazioni per lo più di carattere geometrico ai campioni di dati esistenti (anche se a volte possono riguardare l'immagine a livello di luminosità o nitidezza).
Sia per il training che per il validation che per il test set abbiamo riscalato le immagini in [0,1] mentre per il training set abbiamo applicato delle trasformazioni che vengono riportate nella tabella qui sotto riportata:

Trasformazione	Valore scelto
width shift range	0.25
height shift range	0.25
shear range	0.25
zoom range	0.35
horizontal flip	true
fill mode	nearest

Table 1: Tabella parametri data augmentation

Vediamo ora cosa significhino ognuno di questi parametri:

1. *Width shift*: indica lo spostamento orizzontale.
2. *Height shift*: indica lo spostamento verticale.
3. *Shear range*: indica lo spostamento di un oggetto ad una distanza costante.
4. *Zoom range*: indica lo zoom sull'immagine.
5. *Horizontal flip*: capovolge le immagini orizzontalmente.
6. *Fill mode*: sceglie come riempire i pixel vuoti, in questo caso si basa sul pixel più vicino.

Una volta fatto ciò abbiamo artificialmente aumentato la quantità delle nostre immagini di training sulla base delle trasformazioni e riscalato train, validation e test set.

Si passa poi alla costruzione della rete, per quanto riguarda il modello si è deciso di utlizzare una rete neurale coinvoluzionale, questo perchè estremamente adeguata all'image processing infatti la rete coinvoluzionale tramite le trasformazioni kernel è in grado di ridurre drasticamente il numero di parametri.
Dopo vari tentativi di rete facendo variare il numero di layer di coinvoluzione, il numero di nodi per ciascun layer e alcuni iperparametri come il rate del layer dropout si è arrivati alla configurazione della rete di seguito presentata:
La rete neurale coinvoluzionale è composta da un *input layer* con funzione di attivazione *Relu* seguito da 2 layer di coinvoluzione il primo con 64 nodi mentre il secondo con 128 nodi, ovviamente per tutti e 2 gli strati è stata predisposta una funzione di attivazione *Relu* come anche un Kernel di dimensione 3x3.
Dopo ogni layer di coinvoluzione è stato posto uno strato di pooling con il compito di ridurre gradualmente la dimensione della matrice e di aumentare il livello di astrazione.
Successivamente è stato inserito un *layer dropout* che è necessario per evitare che il modello finisca nell' overfitting quello che il *dropout* fa è infatti che alcuni dei neuroni negli strati nascosti o visibili vengono rilasciati o omessi casualmente. Nel *layer dropout* va impostato un parametro ovvero il *rate* che indica la frazione delle unità da droppare che abbiamo deciso di impostare pari a 0.43.
Infine è stato aggiunto un livello *fully connected* che poi esegue di fatto la classificazione, siccome il problema è di classificazione multiclasse è stata scelta una funzione di attivazione *softmax*.
Si è deciso di massimizzare l'*accuracy* minimizzando come funzione di perdita la *categorical crossentropy* che è una funzione di perdita utilizzata nelle task di

classificazione multiclasse, la funzione è costituita in maniera tale per cui venga calcolata una distanza tra due distribuzioni di probabilità.
Per ottimizzare i pesi si è utilizzato un ottimizzatore *adam* con learning rate posto pari a 1e-3.
L'allenamento della rete è stato poi effettuato su 30 epoche e per ogni epoca sono stati impostati 100 step.

3 Risultati

Per il modello preso in esame è stata monitorata la *loss* e la *accuracy* sia sul training set che sul validation set, ricordiamo come il nostro obiettivo sia quello di ottenere la migliore classificazione possibile sul test set, privilegiare dunque di avere un valore elevato sul validation set rispetto ad uno sul training set siccome il validation set è una simulazione di quello che può accadere a livello di classificazione nel test set.
Nel grafico di seguito presentato viene mostrato l'andamento *loss* e la *accuracy* sia sul training set che sul validation set del modello in esame:

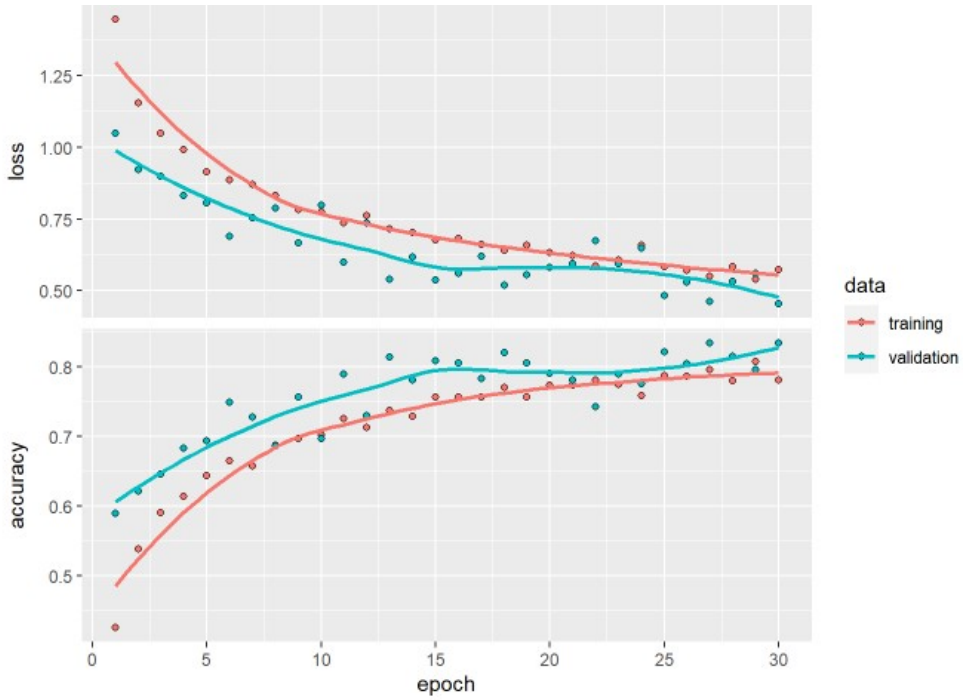

Fig. 3: Loss e Accuracy sul traing e sul validation set

Come possiamo notare dal grafico la classificazione funziona meglio sul validation set rispetto al training set e possiamo affermare anche chiaramente che il modello non va incontro al fenomeno dell'overfitting.

Diamo ora uno sguardo alla matrice di confusione presentata di seguito per verificare dove gli errori avvengano nello specifico.

	buildings	forest	glacier	mountain	sea	street
buildings	333	0	0	0	4	22
forest	5	464	1	3	4	18
glacier	4	2	472	94	44	2
mountain	12	4	39	387	19	3
sea	8	2	23	40	432	2
street	75	2	0	0	6	454

Table 2: Matrice di confusione test set

Risulta abbastanza evidente che la maggior parte degli errori si concentrino nella classificazione delle immagini della classe *mountain*, la confonde maggiormente con la classe *glacier* tuttavia anche con la classe *sea* notiamo che il modello tende spesso a sbagliarsi.
Di seguito viene ripotata un'immagine appartenente al test set della classe *mountain*:

Fig. 4: Immagine test set classe mountain

Il ghiacciaio come mostrato in figura 4 scende da una montagna e spesso da vita a specchi d'acqua simili al mare pertanto la rete fatica a distinguere tra le 3 classi poichè spesso sono presenti nella stessa immagine contemporaneamente oltre al fatto che possono avere colori molto simili (le montagne ricoperte di neve sono facilmente associabili ad un ghiacciaio, il mare invece è azzurro e spesso può essere confuso con un ghiacciaio).

Vengono commessi errori anche nella classificazione della classe *glacier* pressochè per lo stesso motivo.
La rete tende inoltre in alcuni casi a confondere *buildings* con *street* nononstante il modello le distingua abbastanza bene le due classi a livello generale.
Risulta invece ottima la classificazione della classe *forest*.
A livello generale sul test set il modello ha classificato con un' *accuracy* dell'0.85.

4 Discussioni

L'obiettivo dell'analisi era quello di prevedere raggiungendo il più alto valore possibile di accuracy i dati appartenti al test set di Intel Image classification, un dataset contenente immagini appartenenti a 6 classi differenti.
Nello specifico il dataset era già suddiviso in training e test set, il primo sarebbe servito per l'allenamento della rete, mentre la classificazione sul test set era il reale obiettivo dello studio.
Si è proposto un modello di rete neurale coinvoluzionale con 2 strati di coinvoluzione che sul validation set, ottenuto togliendo una frazione di dati al training set ha ottenuto un valore di accuracy superiore allo 0.83 confermato poi sul test set con un valore di *accuracy* pari a 0.85.
Abbiamo poi visto grazie ad una matrice di confusione dove si concentrassero gli errori e abbiamo provato a spiegare cosa stesse alla base di questi errori che la rete continuava a commettere, si trattava infatti di dati molto dubbi su cui la rete faticava a prendere una decisione univoca.
Una soluzione analizzata sarebbe potuta essere quella di introdurre addizionali canali informativi in input alla rete. Si è preso in considerazione per esempio il "brightness" dell' immagine caratteristica che avrebbe consentito una migliore separazione delle classi *sea* e *glacier*, ma purtroppo sono stati riscontrati dei problemi implementativi in R.
Sono stati inotre riscontrati problemi di sistema nell'ampliare il modello oltre i 3 layer di coinvoluzione per mancanza di memoria sul sistema utilizzato.