PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

Alhamdoosh, Fatemah
Università degli studi di Firenze
fatemah.alhamdoosh@stud.unifi.it

Febraio 2021

1 Riconoscimento di identità, espressione e unicode

In questo progetto presentiamo un'archetittura Gerarchica di una rete neurale si chiama pointNet++, che l'abbiamo usato per risolvere tre compiti: riconoscimento di identità , riconoscimento di espressione faciale e riconoscimento di unicode definendoli come un problema di classificazione supervisionato.
Il nostro obiettivo è quello di costeruire un classificatore in grado di riconoscere la classe a cui appartiene un certo dato.

la figura (1) mostra esempi di alcuni dati con le classi ad esse asscociati.

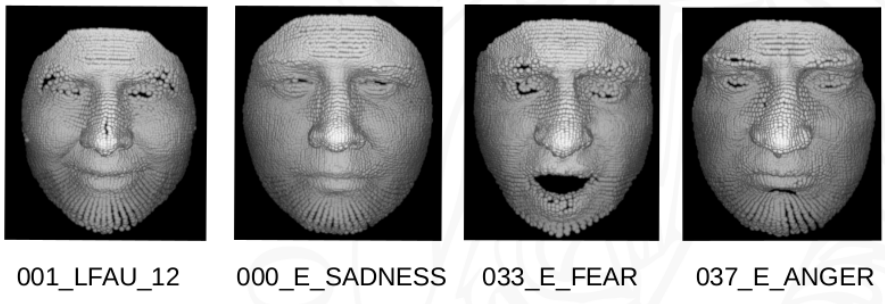

Figure 1: Esempi di nuvole di punti con i corrispondenti classi associati

2 Data: 3D Point Cloud

I dati a nostro dispoizione sono nuvoli di punti tre dimensionale. Una nuvola di punti 3D è un insieme di punti 3D Pi | i = 1, ..., n, dove ogni punto Pi è un vettore delle sue coordinate (x, y, z) a cui si può concatenere canali di

funzionalità extra come colore, normale ecc.

Nel nostro dataset abbiamo 2889 nuvola di punti, ognuna è costituita da 6704 vettore nello spazio tre dimensionale.

Ad ogni nuvola di punti viene associato tre classi: un id, un espressione faciale e un'unicod.
Per esempio un nome di file : "bs011_E_HAPPY_0.pt" viene interpretato nel seguento modo:
011 è id , E è unicod , HAPPY è un espressione.

In totale abbaimo 105 diversi classi di id, 35 classi di espressione ma dobbiamo considirare solo 7 classi, e 5 classi di unicod.

Proprietà di una nuvola di punti:

1-Insieme non ordinato di vettori: A differenza degli array di pixel nelle immagini o matirici di voxel , la nuvola di punti è un insieme di punti senza ordine specifico. In altre parole, una rete che consuma una nuvola di punti di N punto deve essere invariante a N! permutazioni dei punti della nuvola.

2-Interazione tra punti: I punti provengono da uno spazio con una metrica di distanza il che Significa che i punti non sono isolati, e i vicini formano un sottoinsieme significativo .

3-Invarianza sotto trasformazioni: Come un oggetto geometrico, la rappresentazione appresa dell'insieme di punti dovrebbe essere invariante a certe trasformazioni, come rotazione e traslazione.

Caratteristiche di Point Cloud:
sono tipicamente classificati come intrinseci o estrinseci. Possono anche essere classificati come caratteristiche locali e caratteristiche globali. Per un compito specifico, non è banale trovare la combinazione ottimale di caratteristiche.

3 PointNet

Pointnet è un'architettura di reti neurali mostrata in figura (2) ,che è stata modellata per consumare nuvole di punti direttamente come un'insieme di vettori non ordinati, quindi senza un proccesso di preprocessing per trasformarli in una struttura dati standarizzata come per esempio un'immagine o un voxel.

L'idea di base è piuttosto semplice ma è stata lo stato dell'arte nel 2017. Si basa sulla funzione simmetrica per aggregare l'informazione da tutti i punti,è la funzione così detta max pooling, che essendo simmetrica vuol dire che è uguale per tutte le permutazioni del input.

Una funzione f è simmetrica se f(x1,x2)=f(x2,x1).
Quindi l'idea è quella di approssimare una funzione generale definita su un insieme di punti applicando una funzione simmetrica sul insieme di punti trasformati.

Questa funzione dimostra la sua efficenza sulle altre idee come:
1- Ordinare i punti delle nuvole prima del addestramento del modello.
2- creare un insieme di permutazioni dei punti della nuvola e passarle a una RNN.

L'altra chiave è quella di allineare le nuvole di punti per garantire l'invarianza dei punti alla translazione e la rotazione. Questo è realizzabile attravesro l'appreso di una trasformazione comune che non è altro che una piccola rete neurale si chiama T_net.

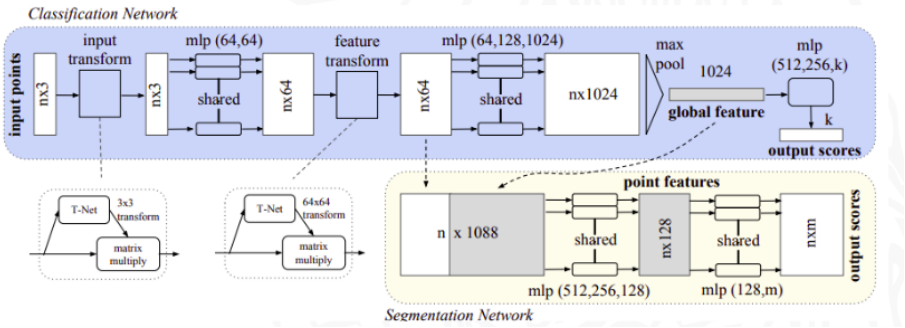

Figure 2: Pointnet

4 PointNet++

La limitazione più grossa di PointNet è che pointnet non cattura le caratteristiche locali indotte dallo spazio metrico in cui vivono i punti, il che limità la possibilità e la capacità di riconoscere schemi a grana fine e la generalizzabilità a scene complesse.

a superare questa limitazione è stata l'architettura PointNet++ mostrata in figura (3), che è una versione gerarchica recursiva che applica PointNet non più direttemente sull'intero input, ma a partizioni dell'input, il che permette di catturare caratteristiche locali a livillo più fine.
La prima domanda che si pone è come possiamo partitzionare una nuvola di punti?
La risposta seconda l'articolo è utilizzando due algoritmi, Farthest Point sample(FPS), che ha il compito di campionare dalla nuvola di punti un insieme di punti detti centroidi, uniformamente distribuiti. la figura (4) mostra alcuni passagi dell'algoritmo FPS.
I centroidi ottenuti da FPS vengono utilizzati dall'algoritmo Query Ball, che

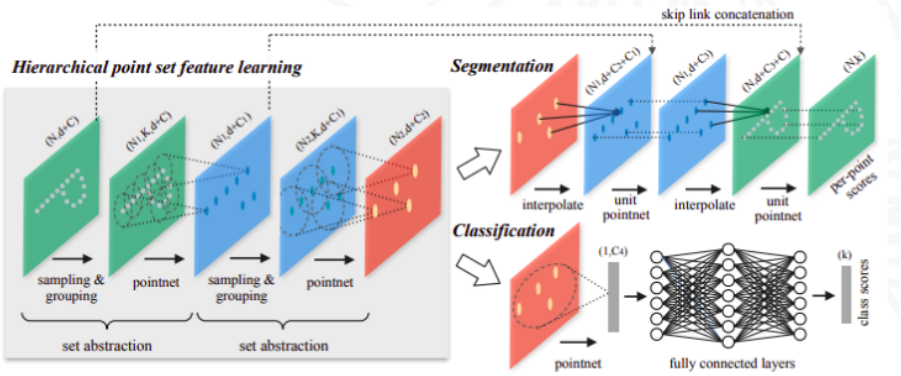

Figure 3: PointNet++

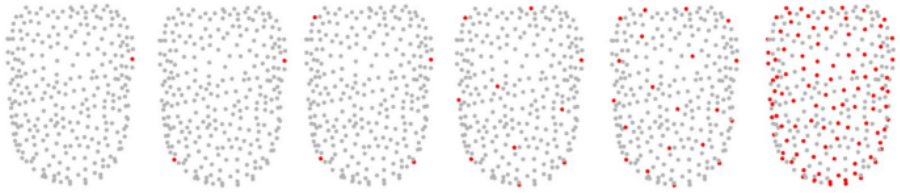

Figure 4: Esempio su un dato 2D,che visualizza i punti campionati da FBS in rosso in alcune iterazioni dell'algoritmo.

riesce a trovare i punti che distano da un centroido al più una distanza r. Così facendo otteniamo un insieme di gruppi di punti sovrapposti uniformamente partizionati.

Quest'architettura in figura (4)viene chiamata single scale grouping(SSG), perchè ad ogni livillo della gerarchia le zone locali ovvero i partizioni sono definiti ad una scala unica, per esemipo al primo livillo il raggio di ogni zona locale è 0.2, mentre al secondo livello è 0.4.

4.1 Multi Scale Grouping MSG:

Limitazione di SSG:
la nuvola di pounti possa avere densità variabile in aree diverse , il che è abbastanza comune nei dati reali come la scansione del sensore della struttura riportata in figura(6). Questa densità variabile rende la scelta della scala appropriata di una partitzione locale più proplematica e sfidabile.

Perciò viene suggerito il modello multi scale grouping, che è semplicemente ragruppa i punti ad ogni livello a diverse scale, per esempio al primo livello li

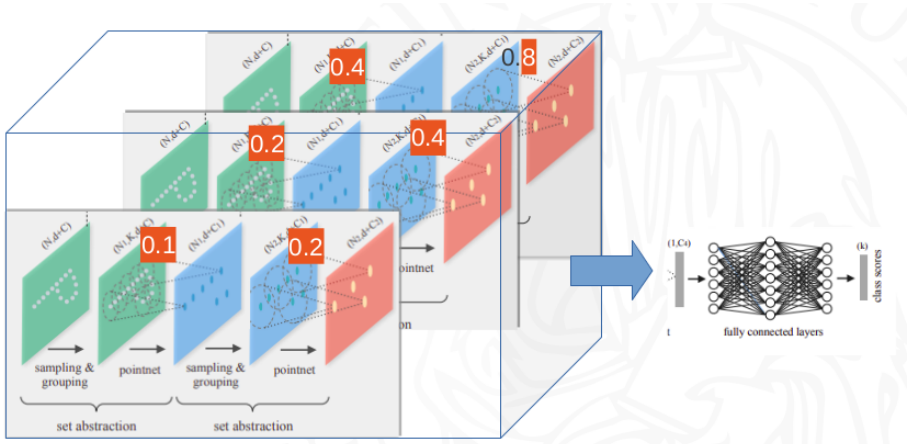

Figure 5: Multi Scale Grouping

Figure 6: Scansione da un sensore di una struttura, a sinistra l'immagine RGB, a destra la nuvola di punti corrispondente.

raggruppa ai raggi 0.1,0.2,0.4, al secondo livello li ragruppa ai raggi 0.2,0.4,0.8 , ad ogni livello, vengono concatenati insieme i vettori di caratteristiche estratti da tutte le scale per dare una rappresentazione più completa che supera il limite della densità variabile.

5 Esperimenti:

5.1 Riconoscimento d'identita:

Ho diviso il dataset in 70% per il training set , e 30% per il test set. Questa rete ha un insieme di heperparametri impostati, riporto alcuni di loro con il valore defualt ad essi assosciato e che l'ho utilizzato nelle esperimenti se non specificato diversamente.

Model	npoint	nsample	DecayStep	Normal	acc	avg-acc	loss
SSG	6704	128	80000	True	**0.94**	**0.90**	**0.26**
SSG	6704	**32**	80000	True	0.01	0.002	3.45
SSG	6704	128	80000	**False**	0.01 1	0.11	6.65
MSG	6704	128	80000	True	0.17	0.16	3.93

Figure 7: Tabella di risultati dell'esperimento sul identità.

1- npoint: il numero totale di punti in input, in default 1024, ma io ho usato sempre 6704.
2- nsample:che il numbero di punti in ogni partizione locale, defualt è 32. ci produce un'overfitting sin dalla prima epoca, invece aumentatndolo a 128 punto otteniamo acc 94,avg e loss0.26.
3- il numero di centroidi, o il numero di partizioni, è 512.
4- il rate di dropOut: 0.5 per ssg. 0.6 per MSG.
5- il learning rate 0.001, sembra proprio ideale.
6- l'optimizer: adham o momentum, Adham non è mai funzionato.
7-il l'uso della normale come extra features: l'effetto della normale è sorprendente in questo specifico esperimento con questi combinazioni di parametri. pur avendo usato 128 nsample e momentum ma senza normale si manifesta l'overfitting, al contrario da quanto riportato nell'articolo che l'uso della normale sul data set standard migliora un pocino le prestazione del modello, ma non che lo rende funzionante o meno.

Esistono altri parametri come il numero di filtri in ogni stratto e il numero di livelli, però qualsiasi aumento di questi parametri aumenta anche l'esigenza di più memoria che al momento non è disponibile. E parlando della memoria posso accenare l'effetto del batch size, che credo che sia il motivo per cui il learning curve oscilla tanto su e giù prima di fare un passo avanti e migliore del precedente. Usando il modello SSG il massimo batche di point cloud che sta in memoria è 16, invece usando MSG il massimo è 8.

SSG vs MSG:
L'uso dell'modellp MSG, anche se è più complicato ma sembra che ha condotto a un'overfitting come èimostrto in figura (9) usando i parametri riportati nella tabella (7).

5.2 Riconoscimento d'espressione:

Abbiamo fatto l'esperimento con 35 classi e con 7 classi. Come mostra la distribuzione in figura (10), questo dataset non è bilanciata, in quale la distribuzione

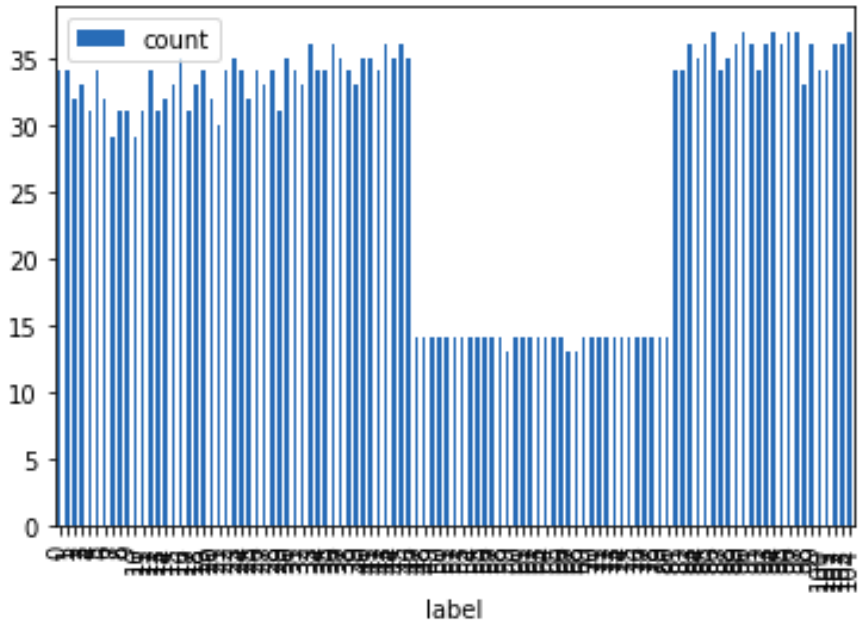

Figure 8: La distribuzione di classi di identità sul dataset

delle classi non è uniforme.
usando le 35 classi, abbiamo una classe con solo 0.3% del dataset, e una classe con 10% del data set mentre i classe restanti da 1% a 4%.

Usando le stesse configurzione funzionati con il modello precedente, si ottiene: Loss: 1.11, accuracy: 0.64 e g acc per classe 57%.
Però monitorando l'accuracy per classe,ho notato che l'accuracy sulla classe denominate è molto alta mentre sulla classe di minoranza l'acc è quasi 0.
Perciò ho provato la funzione weighted loss cross entropy, che da un peso più alto all'errore in classe di minoranza , e da peso più passo all'errore nel classe denominante.
I risultati generici diciamo che sono vicini, ma l'accuracy per classe sembra meglio e più bilanciata.anche se l'andamento generale sembra lo stesso.In figura (11) riporto le curve di loss pesato e non.
Cose che ho provato ma non hanno migliorato la prestazione:

1- Cancellando la classe di minoranza o qullo di maggioranza o entrambi
2-Usare nsample=32, o npoint=2084, learning rate 0.0001.

Alla fine l'ultima prova è stata di provare con solo 7 classi, il risultato è riportato in figura (12).

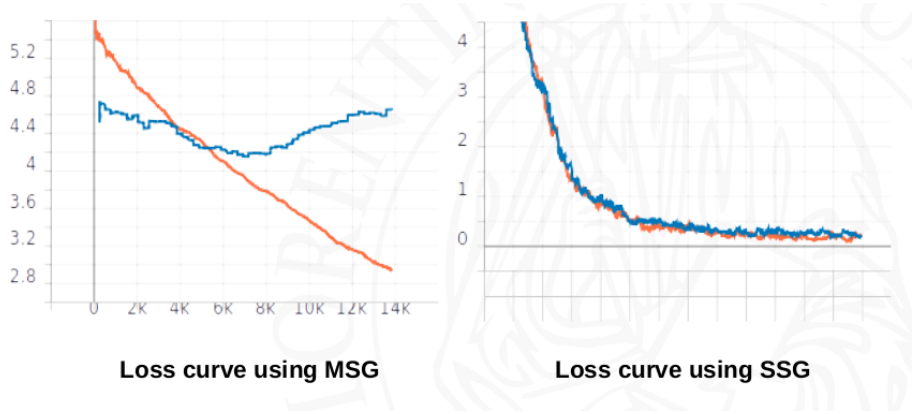

Figure 9: Un confronto tra MSG e SSG. L'andamento dell'errore durante l'addestramento.

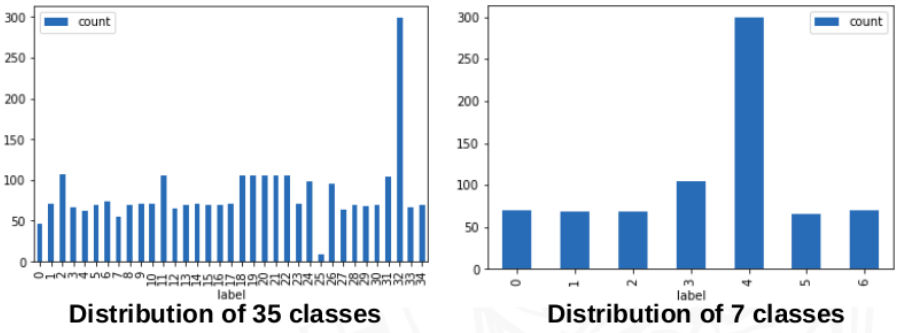

Figure 10: La distribuzioni di classi di espressione sul dataset,usando 35 classi e usando 7 classi

5.3 Riconoscimento di unicode:

In questo esperimento abbiamo 5 classi, e come il precedente non è bilanciata. La classe E costituisce circa il 50% del data set. La figura (13) mostra la distribuzione delle classi e la curva di loss relativa ai parametri usati nella prima riga della tabella in figura (**??**). In pratica ho provato anche il loss pesato ma il risultato è comunque simile.

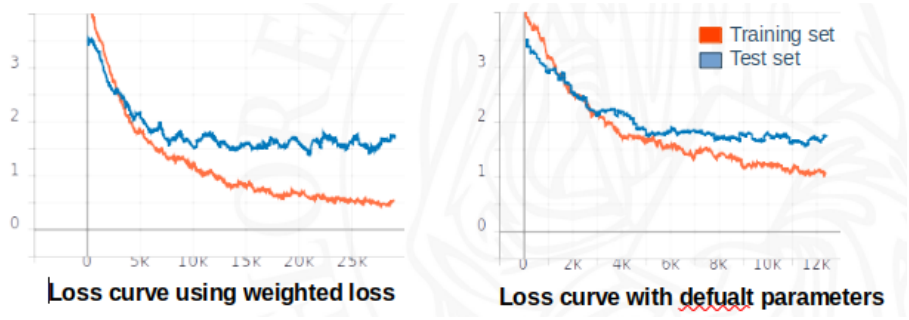

Figure 11: esperimenti di Espressione: Un confronto tra la curva di loss con weighted loss e la curva di loss non pesato.

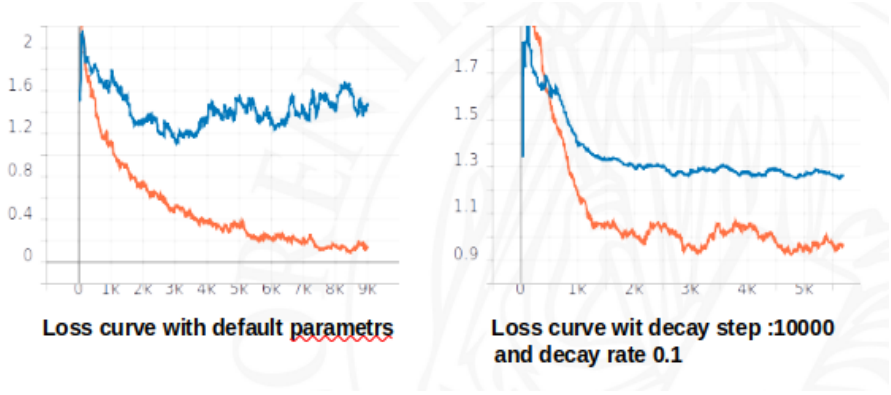

Figure 12: Esperimento di espressione: La curva di loss usando 7 classi.

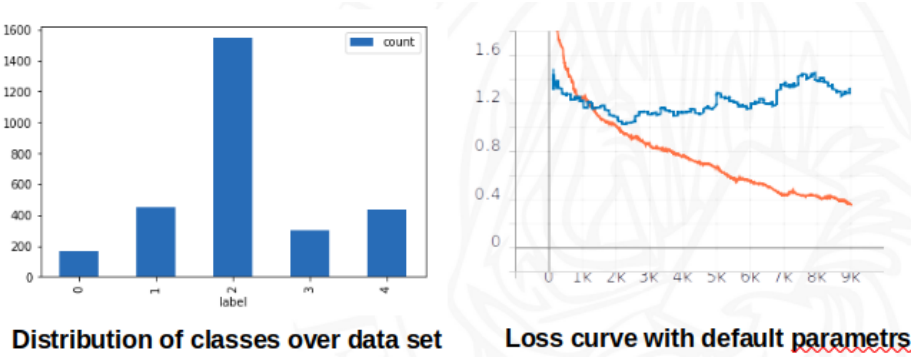

Figure 13: La distribuzione di classi di unicode sul datase, e il loss curve relativo al primo esperimenti della figura 14.

Model	npoint	nsample	DecayStep	Normal	acc	avg-acc	loss
SSG	6704	128	40000	True	**0.64**	**0.57**	**1.12**
SSG	**2048**	**64**	40000	True	0.57	0.34	1.6
SSG	6704	128	40000 Lr=0.0001	**True**	0.58	0.48	1.11

Figure 14: Tabella di esperimenti di unicode.