

# Machine Learning Project

## Computing Distances From Images

June 1, 2021

**Luca Leone**

## Introduzione

L'obiettivo di questo progetto é di implementare un algoritmo che a partire da un'immagine riconosca il cartello che indica una sorgente di alta tensione e ne calcoli la distanza.

Tale programma potrebbe essere utilizzato per analizzare i video di telecamere indossate da lavoratori esposti a sorgenti di pericolo e dare un'allarme quando questi dovessero venire a trovarsi al di sotto di una distanza critica. La presente applicazione ai cartelli di alta tensione potrebbe essere facilmente estesa ad altre sorgenti di pericolo.

Valutare la distanza di un oggetto da una singola immagine é un problema non banale e sono necessarie delle assunzioni per poter ottenere una soluzione univoca. Per realizzare il programma qui presentato sono state fatte le seguenti assunzioni:

- Si utilizza un unico modello di cartello che indica la sorgente di pericolo di alta tensione, cioé un cartello caratterizzato da dimensioni e simboli prefissati.
- Il campo di vista delle ottiche della telecamera e l'aspect ratio delle immagini sono fissate.

Assumere che il cartello abbia sempre le stesse dimensioni é necessario in quanto da una singola immagine un cartello di lato  $x_1$  alla distanza  $r_1$  é indistinguibile da uno cartello visto dalla distanza  $r_2$  di lato  $x_2 = x_1 \frac{r_2}{r_1}$  in quanto le loro immagini sul piano focale della telecamera sono identiche.

Invece assumere che il campo di vista della telecamera che cattura l'immagine e l'aspect ratio dell'immagine siano costanti non é strettamente necessario, questi parametri potrebbero essere dati in input al programma, ma il problema risulterebbe piú complicato e sarebbe necessario realizzare un dataset molto piú grande contenente foto scattate con specifiche diverse che per ragioni di tempo non é stato possibile realizzare.

Il problema da risolvere puó essere separato in due parti: 1) data l'immagine riconoscere il cartello e 2) valutarne la distanza.

Per la parte del riconoscimento del cartello che indica la sorgente di pericolo ho fatto uso delle librerie 'Detectron2' realizzate da 'Facebook AI Research' (<https://github.com/facebookresearch/detectron2>) per realizzare una rete neurale, ci riferiremo a questa rete con il nome di 'predictor'.

Per valutare la distanza ho realizzato una rete neurale che prendere in input l'output del predictor e dà in output la distanza dei cartelli riconosciuti dal predictor, ci riferiremo a questa rete con il nome di 'distanzinator'.

La relazione é organizzata come segue:

Nella sezione 1 é descritto il dataset realizzato.

Nella sezione 2 é descritto il metodo utilizzato per risolvere il problema del riconoscimento del cartello, com'é strutturato l'output del predictor e il preprocessing che viene fatto a tale output prima di darlo in input al distanzinator. Mentre nella sezione 3 viene descritto problema della stima della distanza, i diversi metodi con cui si é cercato di affrontare il problema e con quali criteri sono stati valutati i modelli realizzati.

Nella sezione 4 vengono riportati alcuni degli esperimenti effettuati e i parametri statistici con cui vengono valutati, questi risultati vengono poi analizzati nella



Figure 1: Tre esempi di immagini contenute nel dataset realizzato.

sezione 5.

Nella sezione 6 é descritto il codice realizzato.

E in fine si discutono le conclusioni di questo lavoro.

## 1 La realizzazione del dataset

Per allenare le reti neurali é stato realizzato un dataset di 1822 foto di scenari contenenti il cartello che indica la sorgente del pericolo. Queste foto sono state scattate dalla distanza di 50, 80, 100, 120, 150, 170, 200, 220 e 250 centimetri (circa 200 foto per ogni distanza).

Si é cercato di ottenere la piú ampia varietá possibile in termini di sfondo, di angoli nell'inquadratura del cartello e possibili rotazioni dell'immagine al fine di simulare il posizionamento non ottimale della fotocamera "indossata" per effetto del movimento dell'operatore.

Per le distanze che sono state misurate con un metro a nastro, si è assunta un'incertezza di 10 centimetri. Per questo motivo si é deciso di andare a passi di 20-30 centimetri nella realizzazione del dataset. Le foto sono state organizzate in cartelle salvate nella cartella 'Dataset', all'interno della quale sono contenute le sotto-cartelle chiamate 'X\_cm' ognuna delle quali contiene le foto scattate alla distanza X in modo da poterlo leggere ed eventualmente ampliarlo con facilitá.

Questo stesso dataset é stato utilizzato sia per allenare il predictor a riconoscere il cartello che indica la sorgente di pericolo che per allenare il distanzinatore a valutare la distanza.

In figura 1 sono riportate tre immagini contenute nel dataset realizzato.

Per allenare il predictor é stato realizzato un dataset per il training in formato COCO (Common Objects in Context) con 676 elementi del dataset costruito all'inizio e un dataset per la validation di 227 elementi. Le immagini sono state etichettate con il software labelme (disponibile al seguente indirizzo: <https://github.com/wkentaro/labelme>) e poi il dataset in formato COCO é stato realizzato con labelme2coco (disponibile al seguente indirizzo: <https://github.com/fcakyon/labelme2coco>).

Per allenare il distanzinatore al fine risparmiare risorse e tempo il dataset é stato preprocessato per realizzare un nuovo dataset costituito da due tensori: uno di dimensione 2 avente per righe il risultato del preprocessing di tutti gli elementi del dataset realizzato all'inizio. Il secondo tensore di dimensione 1 avente per righe le distanze corrispondenti, ogni volta viene estratto dal primo tensore la combinazione di features che si vogliono dare in input alla rete.

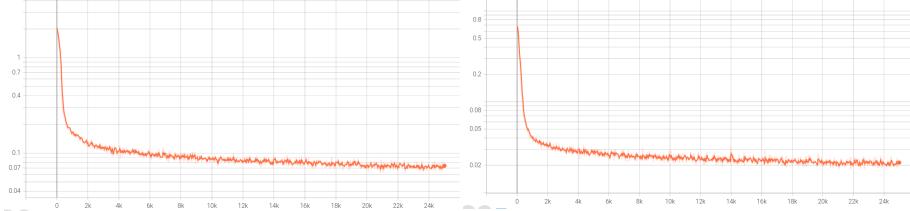


Figure 2: Andamento delle funzioni di loss del predictor. A sinistra la loss totale e a destra quella sul posizionamento delle maschere.

## 2 L'object detection

Come detto prima, per risolvere il problema del riconoscimento del cartello che indica la sorgente di pericolo si é fatto uso delle librerie 'Detectron2' rese disponibili da 'Facebook AI Research'. Per realizzare il predictor ho utilizzato il modello pre-allenato 'mask rcnn R50FPN 3x' dal model zoo di Detectron2 che é stato allenato a riconoscere il cartello.

Il predictor é stato allenato fino alla convergenza della funzione di loss di default, prestando attenzione anche alla loss sul posizionamento delle maschere. In figura 2 é riportato l'andamento delle due funzioni di loss.

Il predictor cosí ottenuto é stato valutato su un dataset di 227 foto usando l'evaluator di default di Detectron2. Ha ottenuto un'average precision del 97.803% per i bounding boxes ed una average precision del 99.380% per la segmentazione.

In modalità d'inferenza il predictor prende in input un'immagine (o una lista di immagini) e restituisce in output un dizionario (o una lista di dizionari), il contenuto di questi dizionari dipende dal modello di base usato per il predictor, informazioni dettagliate sull'output del predictor possono essere trovate nella documentazione ufficiale di Detectron2 alla pagina web <https://detectron2.readthedocs.io/en/latest/tutorials/models.html>, per l'utilizzo che ne é stato fatto é sufficiente sapere che il dizionario dato in output contiene un campo chiamato 'instances', che a sua volta ha i seguenti campi:

- 'pred\_boxes': un oggetto Boxes che contiene N boxes, una per ogni istanza riconosciuta. Le Boxes sono oggetti definiti nelle librerie Detectron2, sono tensori  $N \times 4$  che contengono le bounding boxes  $(x_1, y_1, x_2, y_2)$  delle N istanze trovate, sono tensori di pytorch e supportano diversi metodi come il calcolo dell'area.
- 'scores': un tensore che contiene i confidence scores delle istanze trovate.
- 'pred\_classes': un tensore che contiene i label delle N istanze trovate, nel nostro caso questo campo é irrilevante perché abbiamo un solo label possibile: il cartello che indica la sorgente del pericolo.
- 'pred\_masks': un tensore dal formato  $(N, H, W)$  che contiene le maschere trovate, N é il numero dell'istanza e H,W sono le coordinate del pixel che é stato riconosciuto appartenere alla maschera dell'istanza N.

## 2.1 Preprocessing dell'output del predictor

In teoria si potrebbe usare il campo instances dell'output del predictor come input del modello di distanzinatore, ma questo non risulta conveniente: il predictor restituisce in output un numero fisso di maschere, di cui la maggior parte saranno vuote, quindi quest'output ha un'elevata dimensione e di tutte le informazione che contiene solo poche sono rilevanti. Inoltre se si volesse dare l'output del predictor come input del distanzinatore, allora il distanzinatore dovrebbe essere una rete neurale profonda per poter imparare come rappresentare quest'input oltre a come stimare la distanza.

Per poter stimare la distanza con una rete neurale 'piccola' (una o due hidden layers con poche hidden units), l'output del predictor viene preprocessato prima di essere dato in input al distanzinatore.

Per ogni istanza trovata nell'immagine si estraggono le seguenti features:

- Le coordinate del pixel piú in alto della maschera.
- Le coordinate del pixel piú in basso della maschera.
- Le coordinate del pixel piú a destra della maschera.
- Le coordinate del pixel piú a sinistra della maschera.
- Le coordinate del pixel piú in alto della maschera ruotata di 45 gradi.
- Le coordinate del pixel piú in basso della maschera ruotata di 45 gradi.
- Le coordinate del pixel piú a destra della maschera ruotata di 45 gradi.
- Le coordinate del pixel piú a sinistra della maschera ruotata di 45 gradi.
- Il rapporto tra l'area coperta della maschera e l'area totale della foto.

L'idea alla base di questa scelta é che, con le assunzioni effettuate all'inizio, per risolvere il problema é sufficiente conoscere le posizioni dei vertici del cartello, se la maschera fosse perfettamente rettangolare allora la prima quaterna di punti estratta in fase di preprocessing darebbe una buona approssimazione delle posizioni dei vertici. Poiché la maschera in generale non é perfettamente rettangolare allora estraggo in fase di preprocessing anche la seconda quaterna di punti con l'idea che almeno una delle due quaterne dia una buona approssimazione della posizione dei vertici. In figura 3 é riportato un esempio in cui la prima quaterna di punti non fornisce una buona rappresentazione dei vertici del cartello mentre la seconda quaterna si.

In preprocessing viene estratta anche l'area coperta dalla maschera dell'istanza riconosciuta in rapporto all'area totale dell'immagine perché vista la natura del problema, dare in input questo parametro potrebbe semplificare il problema.

## 3 La stima della distanza

Per stimare la distanza ho sperimentato varie reti neurali con architetture diverse, provando combinazioni diverse delle features estratte dall'output del predictor in fase di preprocessing. Questo perché le features estratte sono state scelte sulla base del fatto che sono sufficienti a risolvere il problema, ma non é

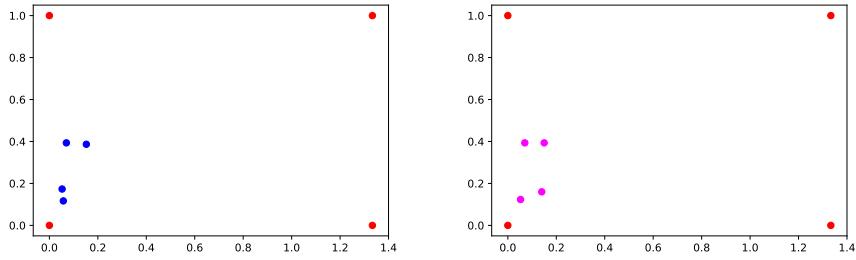


Figure 3: Per la maschera riconosciuta nella foto in alto, plot della posizione dei primi quattro punti (sinistra) e dei secondi quattro punti (destra). Vediamo chiaramente che in questo caso la seconda quaterna di punti fornisce una rappresentazione migliore dei vertici.

detto che siano tutte necessarie. Sono state provate le seguenti combinazioni di features in input:

- Le coordinate della prima quaterna di punti.
- Le coordinate della prima quaterna di punti e il rapporto tra le aree.
- Le coordinate di tutti i punti.
- Le coordinate di tutti i punti e il rapporto tra le aree.

Da ora in poi assumeremo che le reti con 8 features in input utilizzino la prima combinazione di features, le reti con 9 features in input la seconda, le reti con 16 features in input la terza e le reti con 17 features in input la quarta.

Il modello distanzinator è stato realizzato in modo tale che possa operare sia prendendo in input l'immagine che il risultato del preprocessing.

Per allenare i regressori il dataset è stato separato in maniera pseudocausale in dataset di training e di testing con un rapporto di 4:1, in tutti i casi ho usato come funzione di loss la funzione L1Loss (che misura la media dei valori assoluti degli errori) e SGD (Stochastic Gradient Descent) come optimizer.

Per valutare la qualità di un regressore valuto l'andamento delle funzioni di loss sui dataset di training e di testing, il valore a cui convergono le funzioni di loss, l'andamento dell'errore con la distanza e le curve di REC (Regression Error

Characteristic) su entrambi di dataset.

Di seguito vengono riportati i modelli che hanno prodotto i risultati piú interessanti, per tutti i modelli é stata usata la SiLU (Sigmoid Linear Unit) come funzione di attivazione poiché si é osservato produrre i risultati migliori.

Sono stati sperimentati altri modelli che non sono riportati in questa relazione ma possono essere trovati nel Colab allegato.

## 4 Risultati dei modelli provati

Ho fatto piú esperimenti utilizzando reti diverse, di seguito sono riportate delle tabelle che illustrano i risultati ottenuti, il nome del modello é dato in modo da spiegare la sua architettura, il primo numero é il numero di features in input, a seguire ci sono i numeri delle hidden units in ciascun hidden layer e il numero di features in output (che nel nostro caso é sempre 1), infine il nome della funzione di attivazione usata. Ad esempio il modello `[[8,8,1], 'SiLU']` prende 8 features in input (quindi le coordinate dei primi quattro punti estratti in fase di preprocessing), ha un singolo hidden layer con 8 hidden units e usa la SiLU come funzione di attivazione. Invece il modello `[[9, 4, 4, 1], 'SiLU']` prede in input la prima quaterna di punti e il rapporto tra le aree, ha due hidden layer, entrambe con 4 hidden units e usa la SiLU come funzione di attivazione.

## Modello [[8, 4, 1], 'SiLU']

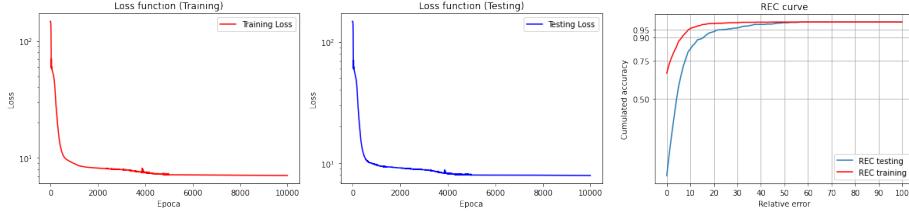


Figure 4: Funzioni di loss e curve di REC del modello [[8, 4, 1], 'SiLU']

Training set	
Errore medio:	7.21 cm
Errori maggiori di 5 cm:	54.72% dei casi
Errori maggiori di 10 cm:	27.21% dei casi
Errore relativo medio:	6.13%
Errori relativi maggiori del 10%:	15.8% dei casi
Errore medio a 50 cm:	8.3 cm (16.6%)
Errore medio a 80 cm:	2.8 cm (3.5%)
Errore medio a 100 cm:	7.6 cm (7.6%)
Errore medio a 120 cm:	8.5 cm (7.1%)
Errore medio a 150 cm:	7.7 cm (5.1%)
Errore medio a 170 cm:	7.0 cm (4.1%)
Errore medio a 200 cm:	6.3 cm (3.2%)
Errore medio a 220 cm:	6.9 cm (3.1%)
Errore medio a 250 cm:	9.2 cm (3.7%)
Testing set	
Errore medio:	7.71 cm
Errori maggiori di 5 cm:	54.73% dei casi
Errori maggiori di 10 cm:	30.11% dei casi
Errore relativo medio:	6.72%
Errori relativi maggiori del 10%:	17.36% dei casi
Errore medio a 50 cm:	10.0 cm (20.0%)
Errore medio a 80 cm:	2.7 cm (3.3%)
Errore medio a 100 cm:	9.0 cm (9.0%)
Errore medio a 120 cm:	8.5 cm (7.1%)
Errore medio a 150 cm:	7.5 cm (5.0%)
Errore medio a 170 cm:	8.3 cm (4.9%)
Errore medio a 200 cm:	8.5 cm (4.3%)
Errore medio a 220 cm:	6.4 cm (2.9%)
Errore medio a 250 cm:	9.1 cm (3.6%)

## Modello [[8, 8, 1], 'SiLU']

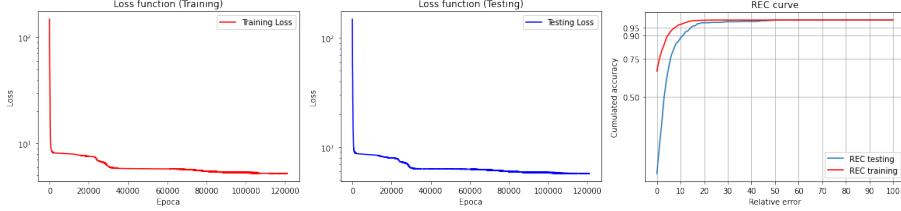


Figure 5: Funzioni di loss e curve di REC del modello [[8, 8, 1], 'SiLU']

Training set	
Errore medio:	5.21 cm
Errori maggiori di 5 cm:	41.62% dei casi
Errori maggiori di 10 cm:	14.12% dei casi
Errore relativo medio:	4.34%
Errori relativi maggiori del 10%:	10.9% dei casi
Errore medio a 50 cm:	3.4 cm (6.9%)
Errore medio a 80 cm:	4.9 cm (6.1%)
Errore medio a 100 cm:	7.1 cm (7.1%)
Errore medio a 120 cm:	6.4 cm (5.3%)
Errore medio a 150 cm:	7.4 cm (5.0%)
Errore medio a 170 cm:	2.7 cm (1.6%)
Errore medio a 200 cm:	5.2 cm (2.6%)
Errore medio a 220 cm:	5.4 cm (2.5%)
Errore medio a 250 cm:	4.2 cm (1.7%)
Testing set	
Errore medio:	5.73 cm
Errori maggiori di 5 cm:	45.71% dei casi
Errori maggiori di 10 cm:	15.6% dei casi
Errore relativo medio:	4.82%
Errori relativi maggiori del 10%:	11.43% dei casi
Errore medio a 50 cm:	4.5 cm (9.0%)
Errore medio a 80 cm:	4.7 cm (5.9%)
Errore medio a 100 cm:	8.1 cm (8.1%)
Errore medio a 120 cm:	6.4 cm (5.3%)
Errore medio a 150 cm:	8.2 cm (5.5%)
Errore medio a 170 cm:	4.3 cm (2.5%)
Errore medio a 200 cm:	6.2 cm (3.1%)
Errore medio a 220 cm:	4.8 cm (2.2%)
Errore medio a 250 cm:	5.2 cm (2.1%)

### Modello modello [[8, 16, 1], 'SiLU']

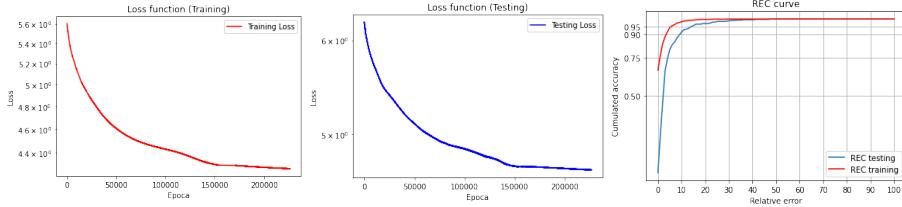


Figure 6: Funzioni di loss e curve di REC del modello [[8, 16, 1], 'SiLU']

Training set	
Errore medio	4.44cm
Errori maggiori di 5cm	32.26% dei casi
Errori maggiori di 10 cm	10.83% dei casi
Errore relativo medio	3.49%
Errori relativi maggiori del 10%	6.88% dei casi
Errore medio a 50 cm:	1.7 cm (3.4%)
Errore medio a 80 cm:	3.2 cm (4.0%)
Errore medio a 100 cm:	7.9 cm (7.9%)
Errore medio a 120 cm:	5.2 cm (4.3%)
Errore medio a 150 cm:	5.6 cm (3.7%)
Errore medio a 170 cm:	2.4 cm (1.4%)
Errore medio a 200 cm:	4.1 cm (2.1%)
Errore medio a 220 cm:	6.3 cm (2.9%)
Errore medio a 250 cm:	3.4 cm (1.4%)
Testing set	
Errore medio	4.86 cm
Errori maggiori di 5 cm	37.8% dei casi
Errori maggiori di 10 cm	11.87% dei casi
Errore relativo medio	3.96%
Errori relativi maggiori del 10%	8.35% dei casi
Errore medio a 50 cm:	2.9 cm (5.9%)
Errore medio a 80 cm:	3.6 cm (4.5%)
Errore medio a 100 cm:	9.2 cm (9.2%)
Errore medio a 120 cm:	4.9 cm (4.1%)
Errore medio a 150 cm:	6.3 cm (4.2%)
Errore medio a 170 cm:	3.1 cm (1.8%)
Errore medio a 200 cm:	4.7 cm (2.4%)
Errore medio a 220 cm:	5.5 cm (2.5%)
Errore medio a 250 cm:	4.8 cm (1.9%)

### Modello [[8, 4, 4, 1], 'SiLU']

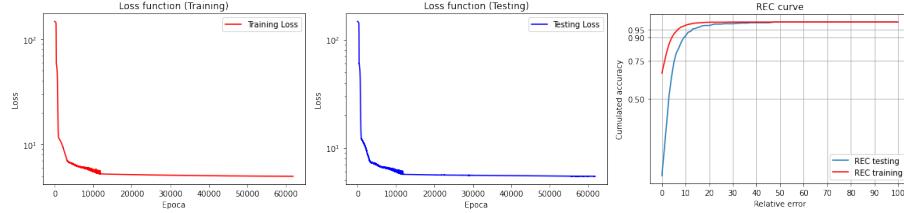


Figure 7: Funzioni di loss e curve di REC del modello [[8, 4, 4, 1], 'SiLU']

Training set	
Errore medio:	4.97 cm
Errori maggiori di 5 cm:	38.92% dei casi
Errori maggiori di 10 cm:	13.83% dei casi
Errore relativo medio:	3.91%
Errori relativi maggiori del 10%:	7.97% dei casi
Errore medio a 50 cm:	2.0 cm (4.0%)
Errore medio a 80 cm:	4.0 cm (5.0%)
Errore medio a 100 cm:	8.6 cm (8.6%)
Errore medio a 120 cm:	5.1 cm (4.2%)
Errore medio a 150 cm:	5.8 cm (3.9%)
Errore medio a 170 cm:	3.1 cm (1.8%)
Errore medio a 200 cm:	4.6 cm (2.3%)
Errore medio a 220 cm:	6.3 cm (2.8%)
Errore medio a 250 cm:	5.0 cm (2.0%)
Testing set	
Errore medio:	5.46 cm
Errori maggiori di 5 cm:	41.32% dei casi
Errori maggiori di 10 cm:	16.26% dei casi
Errore relativo medio:	4.38%
Errori relativi maggiori del 10%:	8.79% dei casi
Errore medio a 50 cm:	3.2 cm (6.4%)
Errore medio a 80 cm:	4.2 cm (5.2%)
Errore medio a 100 cm:	9.6 cm (9.6%)
Errore medio a 120 cm:	4.9 cm (4.1%)
Errore medio a 150 cm:	6.4 cm (4.2%)
Errore medio a 170 cm:	3.9 cm (2.3%)
Errore medio a 200 cm:	6.0 cm (3.0%)
Errore medio a 220 cm:	6.2 cm (2.8%)
Errore medio a 250 cm:	5.7 cm (2.3%)

## Modello [[9, 4, 1], 'SiLU']

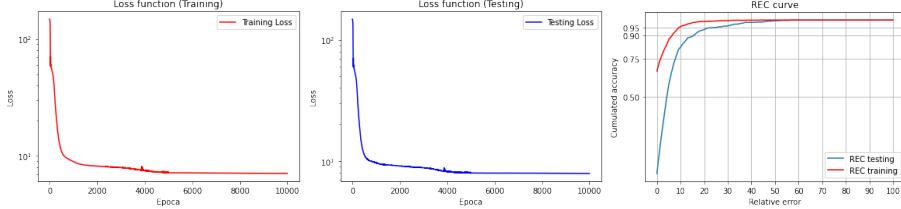


Figure 8: Funzioni di loss e curve di REC del modello [[9, 4, 1], 'SiLU']

Training set	
Errore medio:	6.88 cm
Errori maggiori di 5 cm:	53.77% dei casi
Errori maggiori di 10 cm:	25.9% dei casi
Errore relativo medio:	4.95%
Errori relativi maggiori del 10%:	11.63% dei casi
Errore medio a 50 cm:	1.4 cm (2.7%)
Errore medio a 80 cm:	5.4 cm (6.7%)
Errore medio a 100 cm:	8.7 cm (8.7%)
Errore medio a 120 cm:	5.9 cm (4.9%)
Errore medio a 150 cm:	9.8 cm (6.5%)
Errore medio a 170 cm:	6.6 cm (3.9%)
Errore medio a 200 cm:	7.2 cm (3.6%)
Errore medio a 220 cm:	8.9 cm (4.0%)
Errore medio a 250 cm:	8.3 cm (3.3%)
Testing set	
Errore medio:	7.69 cm
Errori maggiori di 5 cm:	57.8% dei casi
Errori maggiori di 10 cm:	30.77% dei casi
Errore relativo medio:	5.52%
Errori relativi maggiori del 10%:	15.82% dei casi
Errore medio a 50 cm:	2.4 cm (4.8%)
Errore medio a 80 cm:	5.4 cm (6.7%)
Errore medio a 100 cm:	10.1 cm (10.1%)
Errore medio a 120 cm:	5.0 cm (4.1%)
Errore medio a 150 cm:	9.6 cm (6.4%)
Errore medio a 170 cm:	8.2 cm (4.8%)
Errore medio a 200 cm:	10.4 cm (5.2%)
Errore medio a 220 cm:	9.2 cm (4.2%)
Errore medio a 250 cm:	9.8 cm (3.9%)

## Modello [[9, 8, 1], 'SiLU']

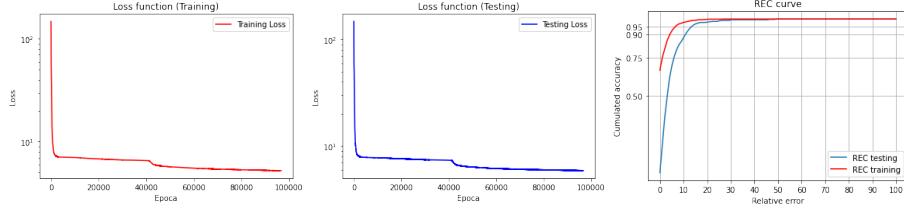


Figure 9: Funzioni di loss e curve di REC del modello [[9, 8, 1], 'SiLU']

Training set	
Errore medio:	5.22 cm
Errori maggiori di 5 cm:	41.99% dei casi
Errori maggiori di 10 cm:	14.85% dei casi
Errore relativo medio:	4.05%
Errori relativi maggiori del 10%:	8.56% dei casi
Errore medio a 50 cm:	1.9 cm (3.8%)
Errore medio a 80 cm:	4.4 cm (5.5%)
Errore medio a 100 cm:	8.9 cm (8.9%)
Errore medio a 120 cm:	4.5 cm (3.8%)
Errore medio a 150 cm:	6.8 cm (4.5%)
Errore medio a 170 cm:	3.4 cm (2.0%)
Errore medio a 200 cm:	5.6 cm (2.8%)
Errore medio a 220 cm:	6.6 cm (3.0%)
Errore medio a 250 cm:	4.8 cm (1.9%)
Testing set	
Errore medio:	5.88 cm
Errori maggiori di 5 cm:	45.71% dei casi
Errori maggiori di 10 cm:	18.24% dei casi
Errore relativo medio:	4.66%
Errori relativi maggiori del 10%:	12.09% dei casi
Errore medio a 50 cm:	2.9 cm (5.7%)
Errore medio a 80 cm:	4.8 cm (5.9%)
Errore medio a 100 cm:	10.6 cm (10.6%)
Errore medio a 120 cm:	4.8 cm (4.0%)
Errore medio a 150 cm:	7.5 cm (5.0%)
Errore medio a 170 cm:	5.0 cm (2.9%)
Errore medio a 200 cm:	6.6 cm (3.3%)
Errore medio a 220 cm:	6.1 cm (2.8%)
Errore medio a 250 cm:	5.8 cm (2.3%)

### Modello modello [[9, 16, 1], 'SiLU']

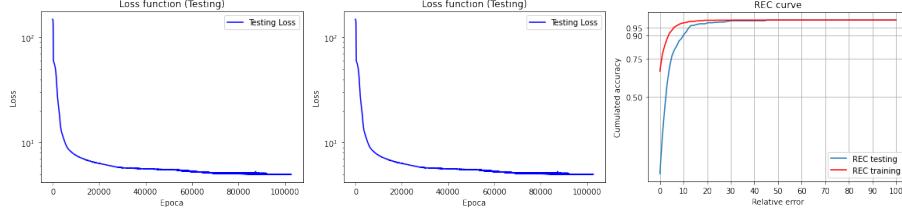


Figure 10: Funzioni di loss e curve di REC del modello [[9, 16, 1], 'SiLU']

Training set	
Errore medio	4.51 cm
Errori maggiori di 5 cm	34.82% dei casi
Errori maggiori di 10 cm	11.05% dei casi
Errore relativo medio	3.56%
Errori relativi maggiori del 10% (training set)	7.24% dei casi
Errore medio a 50 cm:	1.5 cm (3.0%)
Errore medio a 80 cm:	3.9 cm (4.9%)
Errore medio a 100 cm:	8.4 cm (8.4%)
Errore medio a 120 cm:	4.4 cm (3.7%)
Errore medio a 150 cm:	6.0 cm (4.0%)
Errore medio a 170 cm:	2.4 cm (1.4%)
Errore medio a 200 cm:	4.4 cm (2.2%)
Errore medio a 220 cm:	5.9 cm (2.7%)
Errore medio a 250 cm:	3.5 cm (1.4%)
Testing set	
Errore medio	4.99 cm
Errori maggiori di 5 cm	36.92% dei casi
Errori maggiori di 10 cm	12.09% dei casi
Errore relativo medio	4.06%
Errori relativi maggiori del 10%	10.11% dei casi
Errore medio a 50 cm:	2.6 cm (5.1%)
Errore medio a 80 cm:	4.3 cm (5.4%)
Errore medio a 100 cm:	9.8 cm (9.8%)
Errore medio a 120 cm:	4.6 cm (3.9%)
Errore medio a 150 cm:	6.9 cm (4.6%)
Errore medio a 170 cm:	3.1 cm (1.8%)
Errore medio a 200 cm:	5.3 cm (2.6%)
Errore medio a 220 cm:	5.3 cm (2.4%)
Errore medio a 250 cm:	4.4 cm (1.7%)

### Modello [[9, 4, 4, 1], 'SiLU']

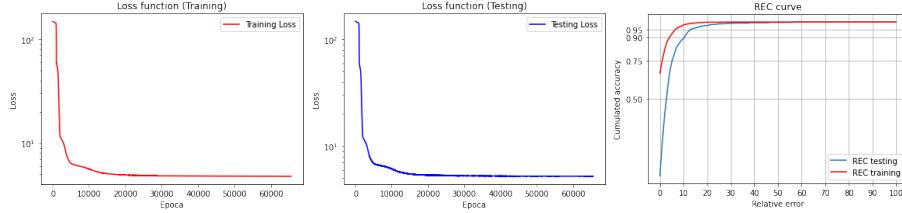


Figure 11: Funzioni di loss e curve di REC del modello [[9, 4, 4, 1], 'SiLU']

Training set	
Errore medio:	4.79 cm
Errori maggiori di 5 cm:	37.31% dei casi
Errori maggiori di 10 cm:	12.73% dei casi
Errore relativo medio:	3.87%
Errori relativi maggiori del 10%:	8.12% dei casi
Errore medio a 50 cm:	2.5 cm (4.9%)
Errore medio a 80 cm:	3.9 cm (4.9%)
Errore medio a 100 cm:	8.2 cm (8.2%)
Errore medio a 120 cm:	5.1 cm (4.3%)
Errore medio a 150 cm:	5.8 cm (3.8%)
Errore medio a 170 cm:	2.5 cm (1.5%)
Errore medio a 200 cm:	4.4 cm (2.2%)
Errore medio a 220 cm:	6.3 cm (2.9%)
Errore medio a 250 cm:	4.2 cm (1.7%)
Testing set	
Errore medio:	5.2 cm
Errori maggiori di 5 cm:	40.44% dei casi
Errori maggiori di 10 cm:	13.41% dei casi
Errore relativo medio:	4.32%
Errori relativi maggiori del 10%:	10.99% dei casi
Errore medio a 50 cm:	3.7 cm (7.3%)
Errore medio a 80 cm:	3.9 cm (4.9%)
Errore medio a 100 cm:	9.4 cm (9.4%)
Errore medio a 120 cm:	5.4 cm (4.5%)
Errore medio a 150 cm:	6.4 cm (4.3%)
Errore medio a 170 cm:	3.4 cm (2.0%)
Errore medio a 200 cm:	5.1 cm (2.5%)
Errore medio a 220 cm:	5.6 cm (2.5%)
Errore medio a 250 cm:	5.1 cm (2.1%)

### Modello [[16, 4, 1], 'SiLU']

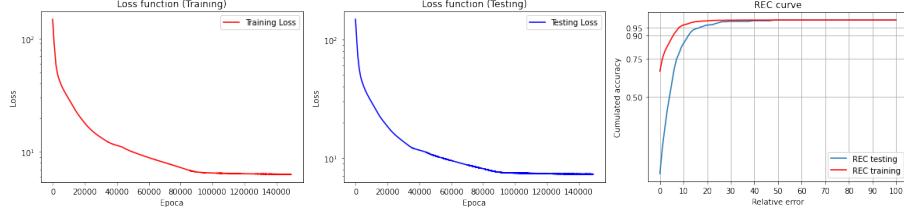


Figure 12: Funzioni di loss e curve di REC del modello [[16, 4, 1], 'SiLU']

Training set	
Errore medio:	6.35 cm
Errori maggiori di 5 cm:	46.67% dei casi
Errori maggiori di 10 cm:	21.65% dei casi
Errore relativo medio:	4.97%
Errori relativi maggiori del 10%:	11.41% dei casi
Errore medio a 50 cm:	3.2 cm (6.5%)
Errore medio a 80 cm:	4.5 cm (5.7%)
Errore medio a 100 cm:	9.2 cm (9.2%)
Errore medio a 120 cm:	6.2 cm (5.1%)
Errore medio a 150 cm:	7.1 cm (4.7%)
Errore medio a 170 cm:	5.4 cm (3.2%)
Errore medio a 200 cm:	7.4 cm (3.7%)
Errore medio a 220 cm:	10.1 cm (4.6%)
Errore medio a 250 cm:	4.0 cm (1.6%)
Testing set	
Errore medio:	7.3 cm
Errori maggiori di 5 cm:	50.11% dei casi
Errori maggiori di 10 cm:	26.37% dei casi
Errore relativo medio:	5.62%
Errori relativi maggiori del 10%:	15.38% dei casi
Errore medio a 50 cm:	4.1 cm (8.3%)
Errore medio a 80 cm:	4.4 cm (5.5%)
Errore medio a 100 cm:	11.3 cm (11.3%)
Errore medio a 120 cm:	6.2 cm (5.2%)
Errore medio a 150 cm:	6.8 cm (4.5%)
Errore medio a 170 cm:	6.2 cm (3.6%)
Errore medio a 200 cm:	11.1 cm (5.6%)
Errore medio a 220 cm:	10.9 cm (4.9%)
Errore medio a 250 cm:	5.4 cm (2.1%)

### Modello modello [[16, 8, 1], 'SiLU']

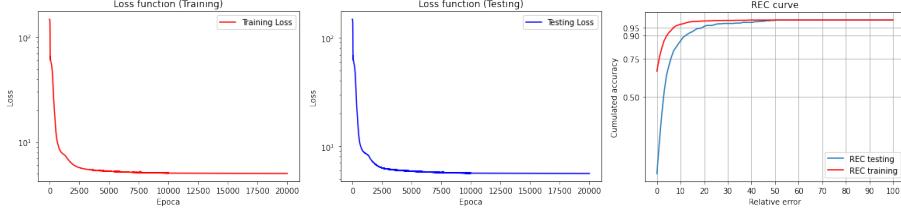


Figure 13: Funzioni di loss e curve di REC del modello [[16, 8, 1], 'SiLU']

Training set	
Errore medio:	5.02 cm
Errori maggiori di 5 cm:	38.26% dei casi
Errori maggiori di 10 cm:	12.73% dei casi
Errore relativo medio:	4.57%
Errori relativi maggiori del 10%:	10.75% dei casi
Errore medio a 50 cm:	6.2 cm (12.4%)
Errore medio a 80 cm:	2.8 cm (3.5%)
Errore medio a 100 cm:	8.4 cm (8.4%)
Errore medio a 120 cm:	5.6 cm (4.7%)
Errore medio a 150 cm:	5.2 cm (3.5%)
Errore medio a 170 cm:	2.2 cm (1.3%)
Errore medio a 200 cm:	5.0 cm (2.5%)
Errore medio a 220 cm:	5.4 cm (2.5%)
Errore medio a 250 cm:	3.8 cm (1.5%)
Testing set	
Errore medio:	5.6 cm
Errori maggiori di 5 cm:	41.98% dei casi
Errori maggiori di 10 cm:	15.82% dei casi
Errore relativo medio:	5.22%
Errori relativi maggiori del 10%:	13.63% dei casi
Errore medio a 50 cm:	7.6 cm (15.3%)
Errore medio a 80 cm:	2.8 cm (3.5%)
Errore medio a 100 cm:	10.8 cm (10.8%)
Errore medio a 120 cm:	5.8 cm (4.8%)
Errore medio a 150 cm:	6.1 cm (4.1%)
Errore medio a 170 cm:	3.0 cm (1.8%)
Errore medio a 200 cm:	5.7 cm (2.8%)
Errore medio a 220 cm:	5.3 cm (2.4%)
Errore medio a 250 cm:	4.4 cm (1.8%)

### Modello [[16, 16, 1], 'SiLU']

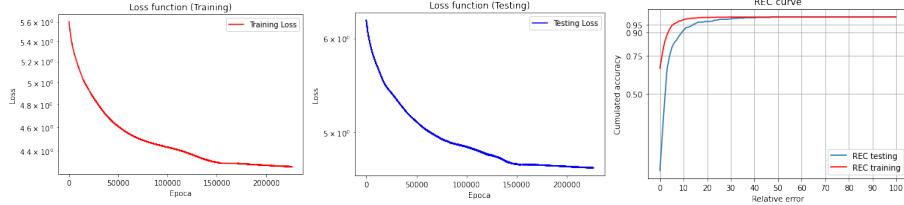


Figure 14: Funzioni di loss e curve di REC del modello [[16, 16, 1], 'SiLU']

Training set	
Errore medio	4.27 cm
Errori maggiori di 5 cm	32.33% dei casi
Errori maggiori di 10 cm istanze	9.66% dei casi
Errore relativo medio istanze	3.41%
Errori relativi maggiori del 10% istanze	6.66% dei casi
Errore medio a 50 cm istanze:	1.7 cm (3.5%)
Errore medio a 80 cm istanze:	3.0 cm (3.7%)
Errore medio a 100 cm istanze:	7.9 cm (7.9%)
Errore medio a 120 cm istanze:	5.3 cm (4.4%)
Errore medio a 150 cm istanze:	4.8 cm (3.2%)
Errore medio a 170 cm istanze:	2.1 cm (1.2%)
Errore medio a 200 cm istanze:	4.2 cm (2.1%)
Errore medio a 220 cm istanze:	5.4 cm (2.4%)
Errore medio a 250 cm istanze:	3.3 cm (1.3%)
Testing set	
Errore medio istanze	4.67 cm
Errori maggiori di 5 cm istanze	33.63% dei casi
Errori maggiori di 10 cm istanze	12.31% dei casi
Errore relativo medio istanze	3.92%
Errori relativi maggiori del 10% istanze	8.79% dei casi
Errore medio a 50 cm istanze:	3.3 cm (6.%)
Errore medio a 80 cm istanze:	3.2 cm (4.0%)
Errore medio a 100 cm istanze:	9.6 cm (9.6%)
Errore medio a 120 cm istanze:	5.3 cm (4.4%)
Errore medio a 150 cm istanze:	5.0 cm (3.3%)
Errore medio a 170 cm istanze:	2.8 cm (1.6%)
Errore medio a 200 cm istanze:	4.8 cm (2.4%)
Errore medio a 220 cm istanze:	4.9 cm (2.2%)
Errore medio a 250 cm istanze:	3.9 cm (1.5%)

### Modello modello [[16, 4, 4, 1], 'SiLU']

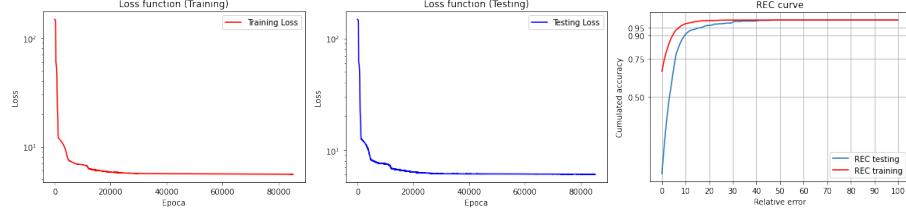


Figure 15: Funzioni di loss e curve di REC del modello [[16, 4, 4, 1], 'SiLU']

Training set	
Errore medio:	5.56 cm
Errori maggiori di 5 cm:	42.72% dei casi
Errori maggiori di 10 cm:	16.61% dei casi
Errore relativo medio:	4.25%
Errori relativi maggiori del 10%:	8.49% dei casi
Errore medio a 50 cm:	2.0 cm (3.9%)
Errore medio a 80 cm:	4.1 cm (5.2%)
Errore medio a 100 cm:	9.1 cm (9.1%)
Errore medio a 120 cm:	5.6 cm (4.7%)
Errore medio a 150 cm:	6.1 cm (4.1%)
Errore medio a 170 cm:	4.2 cm (2.5%)
Errore medio a 200 cm:	5.5 cm (2.8%)
Errore medio a 220 cm:	6.9 cm (3.1%)
Errore medio a 250 cm:	6.3 cm (2.5%)
Testing set	
Errore medio:	6.05 cm
Errori maggiori di 5 cm:	46.15% dei casi
Errori maggiori di 10 cm:	20.88% dei casi
Errore relativo medio:	4.79%
Errori relativi maggiori del 10%:	9.67% dei casi
Errore medio a 50 cm:	3.3 cm (6.7%)
Errore medio a 80 cm:	4.5 cm (5.6%)
Errore medio a 100 cm:	10.6 cm (10.6%)
Errore medio a 120 cm:	5.7 cm (4.8%)
Errore medio a 150 cm:	5.9 cm (4.0%)
Errore medio a 170 cm:	4.8 cm (2.8%)
Errore medio a 200 cm:	7.1 cm (3.5%)
Errore medio a 220 cm:	6.6 cm (3.0%)
Errore medio a 250 cm:	7.1 cm (2.8%)

## Modello modello [[16, 8, 4, 1], 'SiLU']

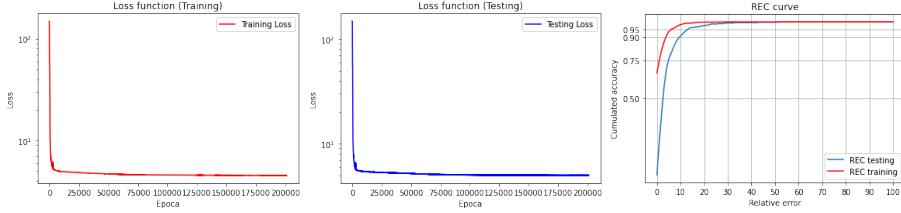


Figure 16: Funzioni di loss e curve di REC del modello [[16, 8, 4, 1], 'SiLU']

Training set	
Errore medio:	4.53 cm
Errori maggiori di 5 cm:	35.55% dei casi
Errori maggiori di 10 cm:	11.34% dei casi
Errore relativo medio:	3.62%
Errori relativi maggiori del 10%:	6.95% dei casi
Errore medio a 50 cm:	1.9 cm (3.7%)
Errore medio a 80 cm:	3.8 cm (4.7%)
Errore medio a 100 cm:	8.4 cm (8.4%)
Errore medio a 120 cm:	4.8 cm (4.0%)
Errore medio a 150 cm:	5.6 cm (3.8%)
Errore medio a 170 cm:	2.4 cm (1.4%)
Errore medio a 200 cm:	4.6 cm (2.3%)
Errore medio a 220 cm:	5.3 cm (2.4%)
Errore medio a 250 cm:	3.8 cm (1.5%)
Testing set	
Errore medio:	4.99 cm
Errori maggiori di 5 cm:	37.36% dei casi
Errori maggiori di 10 cm:	13.19% dei casi
Errore relativo medio:	4.16%
Errori relativi maggiori del 10%:	9.23% dei casi
Errore medio a 50 cm:	3.1 cm (6.2%)
Errore medio a 80 cm:	4.2 cm (5.3%)
Errore medio a 100 cm:	10.0 cm (10.0%)
Errore medio a 120 cm:	5.1 cm (4.2%)
Errore medio a 150 cm:	6.2 cm (4.1%)
Errore medio a 170 cm:	2.9 cm (1.7%)
Errore medio a 200 cm:	5.6 cm (2.8%)
Errore medio a 220 cm:	4.9 cm (2.2%)
Errore medio a 250 cm:	4.1 cm (1.6%)

### Modello modello [[17, 4, 1], 'SiLU']

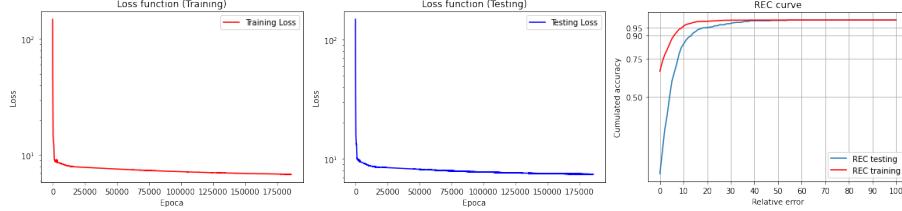


Figure 17: Funzioni di loss e curve di REC del modello [[17, 4, 1], 'SiLU']

Training set	
Errore medio:	6.84 cm
Errori maggiori di 5 cm:	52.08% dei casi
Errori maggiori di 10 cm:	24.58% dei casi
Errore relativo medio:	5.59%
Errori relativi maggiori del 10%:	14.12% dei casi
Errore medio a 50 cm:	6.0 cm (12.1%)
Errore medio a 80 cm:	3.5 cm (4.4%)
Errore medio a 100 cm:	8.3 cm (8.3%)
Errore medio a 120 cm:	7.3 cm (6.1%)
Errore medio a 150 cm:	8.2 cm (5.5%)
Errore medio a 170 cm:	5.9 cm (3.5%)
Errore medio a 200 cm:	6.3 cm (3.1%)
Errore medio a 220 cm:	7.2 cm (3.3%)
Errore medio a 250 cm:	8.6 cm (3.4%)
Testing set	
Errore medio:	7.41 cm
Errori maggiori di 5 cm:	54.29% dei casi
Errori maggiori di 10 cm:	25.71% dei casi
Errore relativo medio:	6.08%
Errori relativi maggiori del 10%:	15.82% dei casi
Errore medio a 50 cm:	6.7 cm (13.5%)
Errore medio a 80 cm:	3.5 cm (4.3%)
Errore medio a 100 cm:	10.3 cm (10.3%)
Errore medio a 120 cm:	7.5 cm (6.2%)
Errore medio a 150 cm:	8.3 cm (5.5%)
Errore medio a 170 cm:	7.2 cm (4.2%)
Errore medio a 200 cm:	8.8 cm (4.4%)
Errore medio a 220 cm:	7.0 cm (3.2%)
Errore medio a 250 cm:	8.4 cm (3.4%)

### Modello [[17, 8, 1], 'SiLU']

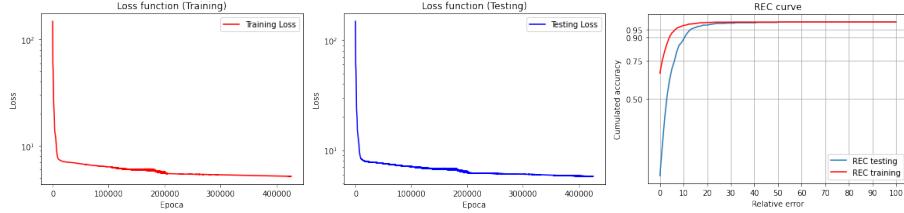


Figure 18: Funzioni di loss e curve di REC del modello [[17, 8, 1], 'SiLU']

Training set	
Errore medio:	5.17 cm
Errori maggiori di 5 cm:	40.23% dei casi
Errori maggiori di 10 cm:	14.56% dei casi
Errore relativo medio:	3.98%
Errori relativi maggiori del 10%:	8.05% dei casi
Errore medio a 50 cm:	1.7 cm (3.4%)
Errore medio a 80 cm:	4.1 cm (5.2%)
Errore medio a 100 cm:	9.0 cm (9.0%)
Errore medio a 120 cm:	4.6 cm (3.8%)
Errore medio a 150 cm:	6.4 cm (4.3%)
Errore medio a 170 cm:	3.2 cm (1.9%)
Errore medio a 200 cm:	5.6 cm (2.8%)
Errore medio a 220 cm:	6.4 cm (2.9%)
Errore medio a 250 cm:	5.2 cm (2.1%)
Testing set	
Errore medio:	5.79 cm
Errori maggiori di 5 cm:	42.64% dei casi
Errori maggiori di 10 cm:	19.12% dei casi
Errore relativo medio:	4.49%
Errori relativi maggiori del 10%:	10.99% dei casi
Errore medio a 50 cm:	2.6 cm (5.2%)
Errore medio a 80 cm:	4.4 cm (5.5%)
Errore medio a 100 cm:	10.5 cm (10.5%)
Errore medio a 120 cm:	4.5 cm (3.8%)
Errore medio a 150 cm:	7.0 cm (4.7%)
Errore medio a 170 cm:	4.7 cm (2.8%)
Errore medio a 200 cm:	7.0 cm (3.5%)
Errore medio a 220 cm:	6.1 cm (2.8%)
Errore medio a 250 cm:	6.4 cm (2.6%)

### Modello [[17, 8, 4, 1], 'SiLU']

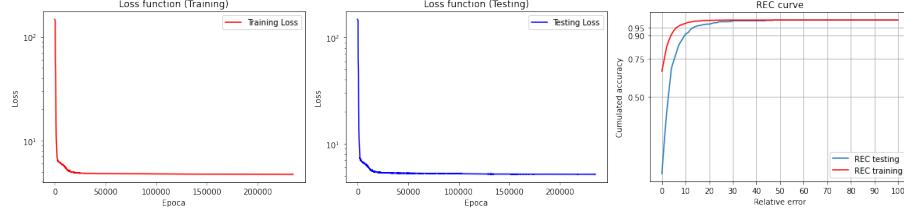


Figure 19: Funzioni di loss e curve di REC del modello [[17, 8, 4, 1], 'SiLU']

Training set	
Errore medio:	4.72 cm
Errori maggiori di 5 cm:	36.36% dei casi
Errori maggiori di 10 cm:	12.07% dei casi
Errore relativo medio:	3.75%
Errori relativi maggiori del 10%:	7.9% dei casi
Errore medio a 50 cm:	1.9 cm (3.8%)
Errore medio a 80 cm:	4.0 cm (5.0%)
Errore medio a 100 cm:	8.5 cm (8.5%)
Errore medio a 120 cm:	5.1 cm (4.2%)
Errore medio a 150 cm:	5.4 cm (3.6%)
Errore medio a 170 cm:	2.5 cm (1.5%)
Errore medio a 200 cm:	4.6 cm (2.3%)
Errore medio a 220 cm:	5.7 cm (2.6%)
Errore medio a 250 cm:	4.5 cm (1.8%)
Testing set	
Errore medio:	5.19 cm
Errori maggiori di 5 cm:	39.34% dei casi
Errori maggiori di 10 cm:	15.38% dei casi
Errore relativo medio:	4.26%
Errori relativi maggiori del 10%:	9.45% dei casi
Errore medio a 50 cm:	3.2 cm (6.5%)
Errore medio a 80 cm:	3.9 cm (4.9%)
Errore medio a 100 cm:	10.1 cm (10.1%)
Errore medio a 120 cm:	5.3 cm (4.4%)
Errore medio a 150 cm:	6.1 cm (4.1%)
Errore medio a 170 cm:	3.3 cm (1.9%)
Errore medio a 200 cm:	5.5 cm (2.7%)
Errore medio a 220 cm:	5.6 cm (2.5%)
Errore medio a 250 cm:	5.0 cm (2.0%)

### Modello modello [[17, 16, 1], 'SiLU']

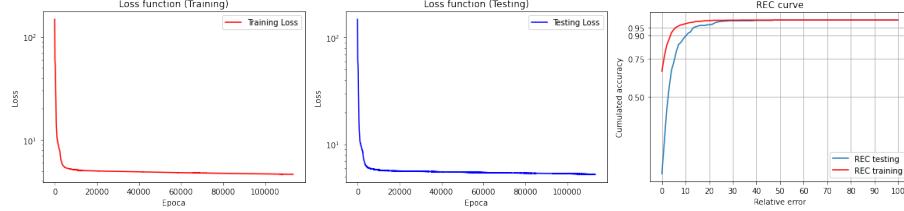


Figure 20: Funzioni di loss e curve di REC del modello [[17, 16, 1], 'SiLU']

Training set	
Errore medio	4.63 cm
Errori maggiori di 5 cm	35.77% dei casi
Errori maggiori di 10 cm	11.7% dei casi
Errore relativo medio	3.76%
Errori relativi maggiori del 10%	8.19% dei casi
Errore medio a 50 cm:	1.6 cm (3.3%)
Errore medio a 80 cm:	4.8 cm (6.0%)
Errore medio a 100 cm:	9.4 cm (9.4%)
Errore medio a 120 cm:	4.2 cm (3.5%)
Errore medio a 150 cm:	5.7 cm (3.8%)
Errore medio a 170 cm:	2.2 cm (1.3%)
Errore medio a 200 cm:	5.2 cm (2.6%)
Errore medio a 220 cm:	5.0 cm (2.3%)
Errore medio a 250 cm:	3.3 cm (1.3%)
Testing set	
Errore medio	5.24 cm
Errori maggiori di 5 cm	40.88% dei casi
Errori maggiori di 10 cm	14.51% dei casi
Errore relativo medio	4.32%
Errori relativi maggiori del 10%	10.55% dei casi
Errore medio a 50 cm:	2.6 cm (5.2%)
Errore medio a 80 cm:	5.0 cm (6.3%)
Errore medio a 100 cm:	11.1 cm (11.1%)
Errore medio a 120 cm:	5.0 cm (4.1%)
Errore medio a 150 cm:	6.5 cm (4.3%)
Errore medio a 170 cm:	3.0 cm (1.7%)
Errore medio a 200 cm:	6.0 cm (3.0%)
Errore medio a 220 cm:	4.9 cm (2.2%)
Errore medio a 250 cm:	4.4 cm (1.7%)

## 5 Analisi dei modelli risultati

Prima di analizzare i risultati dei modelli presentati, per chiarezza é utile fare una breve analisi matematica del problema che vogliamo risolvere.

### 5.1 Una breve analisi matematica del problema

Se il cartello che indica la sorgente del pericolo ha area  $A$ , si trova ad una distanza  $r$  e il vettore che collega il cartello alla fotocamera forma un angolo  $\theta$  con il versore normale al cartello, allora, ignorando le varie forme di aberrazione delle lenti della fotocamera che potrebbero portare ad un certo grado di anamorfismo, il cartello coprirá un angolo solido

$$\Omega = \frac{A \cos(\theta)}{r^2} \quad (1)$$

e quindi la distanza dal cartello sará data dalla relazione

$$r = \sqrt{\frac{A \cos(\theta)}{\Omega}} \quad (2)$$

Il parametro che abbiamo estratto in fase di preprocessing e chiamato rapporto tra le aree é, in realtà, il rapporto tra l'angolo solido coperto dal cartello e l'angolo solido coperto dalla foto, che é costante, denotiamo questo parametro con  $W$ , con le assunzioni che abbiamo fatto all'inizio allora la distanza seguirá la relazione

$$r = K \sqrt{\frac{\cos(\theta)}{W}} \quad (3)$$

dove  $K$  é una costante. Quindi noi vorremmo che il modello impari il valore della costante  $K$  e a valutare  $\sqrt{\cos(\theta)/W}$ . Per questo motivo ci si puó aspettare che i modelli che non ricevono  $W$  in input siano svantaggiati rispetto a quelli che lo ricevono.

### 5.2 Analisi dei risultati ottenuti

Alle misure delle distanze nel dataset é stato assegnato un errore di 10 centimetri, poiché la L1Loss non é altro che il valore medio dell'errore in valore assoluto, allora ci aspettiamo che se il modello funziona bene la funzione di loss converga ad un valore paragonabile con l'errore, infatti in tutti i modelli presentati la funzione di loss converge ad un valore tra i 4 e gli 8 centimetri. Questo fa pensare che i modelli non siano affetti da overfitting e che l'errore dato sulle misure del dataset sia stato leggermente sovrastimato.

Possiamo vedere come il preprocessing ci ha permesso di risolvere il problema con una rete 'piccola', infatti si osserva una differenza marginale tra i modelli con una e due hidden layers.

Inoltre non si osserva una differenza significativa tra i modelli che usano diversi tipi di input. Possiamo quindi concludere che la prima quaterna di punti in realtà é fornisce una buona rappresentazione dei dati.

## 6 Il codice consegnato

Il codice consegnato é strutturato come segue.

Nella prima sezione del codice vengono installati i pacchetti e importate le librerie necessarie al funzionamento del programma.

Poi é definito il modello del distanzinatore. Il costruttore del modello prevede in input l'architettura della rete: una lista il cui primo elemento é il numero di features in input e gli elementi successivi sono il numero di hidden units dei layer (é possibile mettere un numero arbitrario di hidden layers), l'ultimo numero é il numero di elementi in output. Poiché vogliamo che la rete dia in output la distanza del cartello, allora il numero di elementi in output sarà sempre 1. É possibile dare in input la funzione di attivazione da usare, se non specificata verrá usata la SiLU.

Il modello eredita tutte le proprietá della classe nn.Module e in piú sono definiti i metodi:

- `allena()`: questo metodo allena il modello sui dataset di training e di testing specificati in input usando la funzione di loss L1Loss e l'optimizer SGD, il metodo vuole in input il learning rate, il numero di epoch e il momentum. Il modello salva tutti i valori delle funzioni di loss e alla fine del training produrrá i grafici delle funzioni di loss.
- `clean_losses()`: questo metodo elimina i valori salvati delle funzioni di loss.
- `valuta()`: il metodo prende in input i dataset di training e di testing e stampa le statistiche presentate nella relazione per il modello e la curva di REC
- `image_to_data()`: il metodo vuole in input il path di un'immagine e un modello di predictor e dà in output un array contenente gli output pre-processati del predictor, uno per ogni istanza trovata. Se in input si dà `print_output='y'` allora il metodo produrrá una stampa dell'immagine con le istanze riconosciute.  
Se non vengono riconosciute istanze il metodo restituisce una lista vuota.
- `image_to_distance()`: il metodo vuole in input il path di un'immagine e un modello di predictor e dà in output le distanze di tutti i cartelli riconosciuti nell'immagine. Se in input si dà `print_output='y'` allora il metodo produrrá una stampa dell'immagine con le istanze riconosciute.  
Se non vengono riconosciute istanze il metodo restituisce una lista vuota.

Nella seconda sezione c'è il codice per il training del predictor, prima viene caricato il dataset in formato COCO, poi il predictor viene allenato ed infine viene valutato, il tutto é fatto usando le librerie 'Detectron2'. Il modello allenato viene automaticamente salvato su google drive.

La terza sezione serve a caricare il modello di predictor già allenato in precedenza.

La quarta sezione serve per il preprocessing del dataset per il training del distanzinatore e lo salva su una cartella di google drive.

La quinta sezione serve caricare il dataset preprocessato per il training del distanzinatore.

La sesta sezione contiene i vari esperimenti di predictor, sono organizzati in base

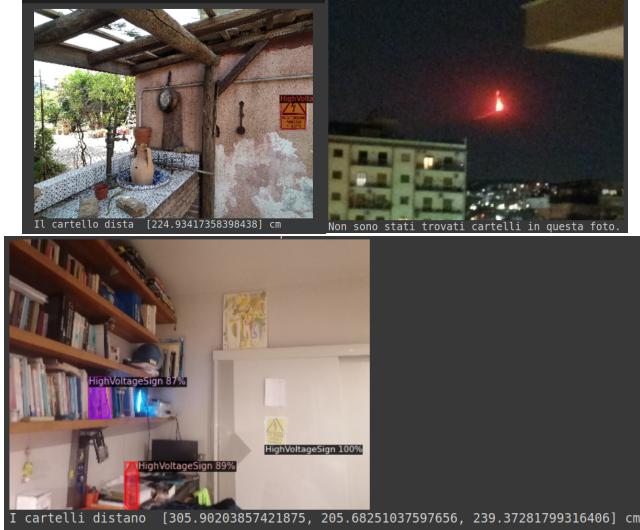


Figure 21: Esempi di output del programma realizzato.

all’input che ricevono, ogni volta il dataset preprocessato viene adattato al tipo di input in questione.

La settima e ultima sezione é la demo di come si usa il programma, la prima cella carica i modelli di predictor e distanzinator salvati su google drive e definisce i metadata per il predictor (le etichette per l’object detection). La seconda cella d un immagine al distanzinator caricato e stampa l’output. Il modello usato nella demo é il modello [[16, 16, 1], ‘SilU’]. Per usare la demo é sufficiente lanciare il primo gruppo di celle con dove vengono installati i pacchetti, importate le librerie e definita la classe distanzinator. Dopo di che si pu andare direttamente alla demo. In figura 21 é riportato un esempio di output del programma. Nella demo é stata volontariamente messa un’immagine in cui il predictor riconosce dei falsi positivi per mostrare come vengono gestiti dal distanzinator.

## 7 Conclusioni

Sono stati realizzati diversi modelli capaci di stimare la distanza del cartello che indica la sorgente di pericolo con un errore medio di circa 5 centimetri sul testing set.

Occasionalmente il predictor riconosce falsi positivi, questo é probabilmente dovuto alla ridotta dimensione del dataset su cui é stato allenato e al fatto che in questo dataset ci sono pochi oggetti che assomigliano al cartello ma non lo sono. Per risolvere il problema occorre realizzare un dataset pi grande su cui allenare il predictor che contenga dei ‘distrattori’, oggetti molto simili al cartello ma che non sono il cartello in modo da forzarlo a guardare dettagli pi fini. Infatti quando il predictor é stato allenato su un dataset troppo piccolo ha imparato a riconoscere come cartello che indica la sorgente del pericolo qualsiasi forma rettangolare.

Nonostante questo problema il programma continua a funzionare se il predictor

riconosce dei falsi positivi in quanto analizza ogni istanza singolarmente, nella peggiore delle ipotesi potrebbe lanciare un allarme quando non é necessario. Non sono stati riscontrati casi di falsi negativi in cui il predictor non é stato capace di riconoscere il cartello.

La threshold del predictor, cioé quanto deve essere sicuro di un'istanza per darla in output puó essere modificata dall'utente all'avvio del programma, una threshold alta ridurrá il numero di falsi positivi, ma aumenta la probabilitá di avere falsi negativi.

Il programma realizzato puó essere usato per analizzare il video di una fotocamera indossata da un utente come descritto nell'introduzione, ma il predictor ha un tempo d'inferenza di circa 6 secondi se eseguita su CPU contro un tempo di inferenza di circa 0.1 secondi se eseguita su CUDA, il che rappresenta una pesante limitazione.

Sarebbe interessante estendere lo studio il comportamento dell'errore per vedere come sia correlato all'angolo  $\theta$ , visto che per piccoli valori di  $\theta$  si ha  $\cos(\theta) \sim 1$  se il dataset fosse sbilanciato con molte foto in cui il cartello é fotografato frontalmente allora la rete potrebbe semplicemente ignorare la dipendenza da  $\theta$  senza essere penalizzata eccessivamente. Purtroppo visto com'è strutturato il dataset non é possibile fare quest'analisi perché i valori dell'angolo  $\theta$  non sono stati riportati.