Università degli Studi di Ferrara

DE: Department of Engineering Ferrara

Progetto di Laboratorio di Intelligenza Artificiale

Michele Fraccaroli - 111140

Indice

- Introduzione
- Il problema
- Modelli computazionali
- Ambiente di lavoro
- Sviluppo
- Risultati

Introduzione

Sviluppo di una rete neurale convoluzionale per il riconoscimento e la classificazione delle cifre dei contatori.

Lo scopo principale è quello di riconoscere e classificare le cifre in situazioni intermedie, ovvero, quando due cifre sono visibili contemporaneamente.

Il problema

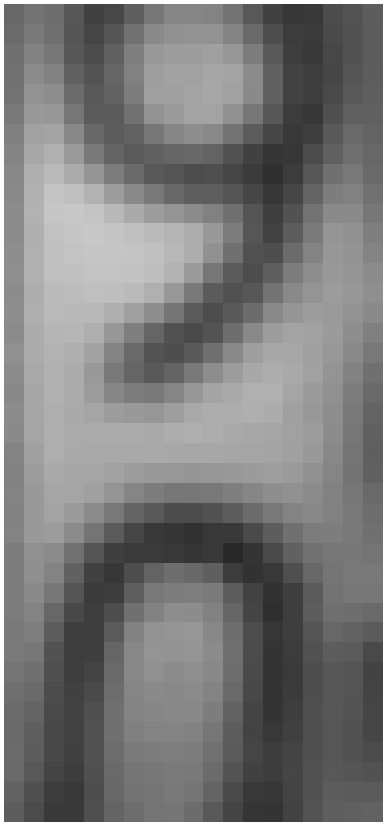


Figura 1 – Cifra in situazione intermedia.

Il problema della classificazione è dovuto alle situazioni intermedie dove la classificazione dipende da quale cifra viene riconosciuta.

Modelli computazionali - 1

Come modello computazionale, è stata utilizzata una *Rete Neurale Artificiale (Artificial Neural Network - ANN)*.

Ovvero un sistema computazionale che prende ispirazione dalla rete neurale biologica.

Dal punto di vista statistico, una rete neurale è un modello di classificazione non lineare.

Modelli computazionali - 2

Una ANN si basa su una serie di nodi connessi, chiamati *Neuroni Artificiali,* le cui connessioni, che simulano le sinapsi del cervello, possono trasmettere segnali tra un neurone e l’altro.

Modelli computazionali - 3

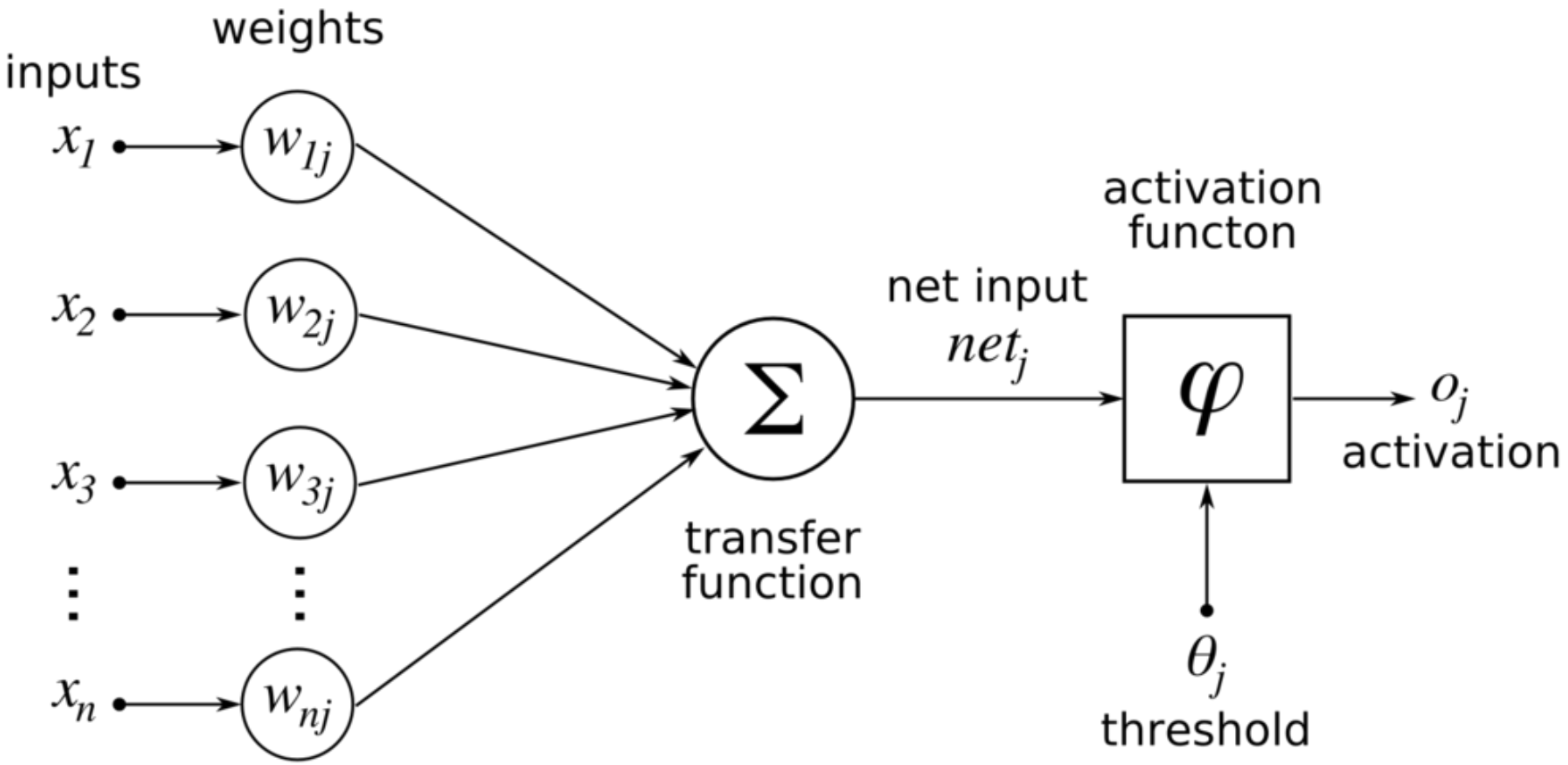


Figura 2 – Neurone artificiale.

Modelli computazionali - 4

Per questo progetto, è stato utilizzato un particolare tipo di rete neurale chiamata *Rete Neurale Convoluzionale* (*Convolutional Neural Network – CNN*).

Questa rete si ispira all'organizzazione della corteccia visiva animale dove ogni neurone è disposto in modo da corrispondere alle regioni di sovrapposizione del campo visivo.

Modelli computazionali - 5

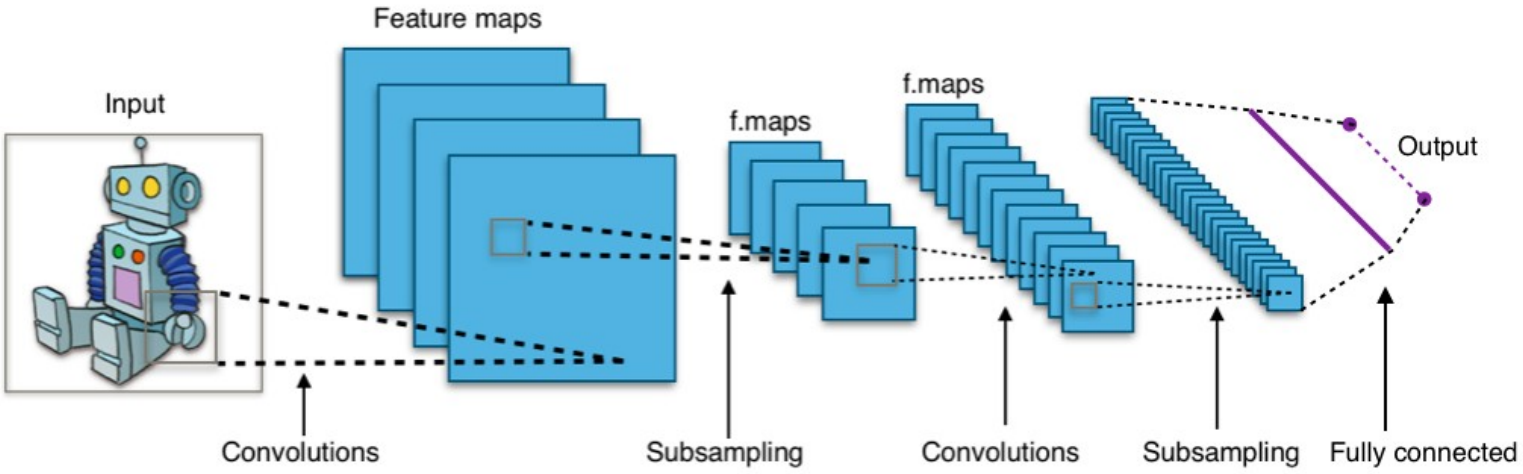


Figura 3 – Struttura tipica di una CNN.

Modelli computazionali - 6

Struttura di una CNN:

- Layer di convoluzione: Applica la convoluzione all'input per poi passare il risultrato al layer successivo (biologicamente rappresenta la risposta ad uno stimolo visivo per un neurone).
- Layer di subsampling: Riduce la dimensione della convoluzione. Serve per rimuovere la sensibilità alle piccole variazioni delle immagini di input.

Modelli computazionali - 7

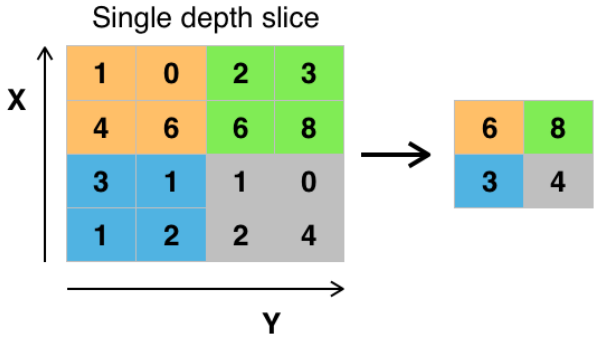


Figura 4 – Subsampling (Max Pooling).

In figura 4 si può vedere come il *Max Pooling* mantenga in uscita solamente i parametri più importanti di quelli che ha ricevuto in ingresso dalla convoluzione.

- Layer completamente connesso: Connette ogni neurone in un strato ad ogni altro neurone di un altro strato (parte finale della rete). Serve per correlare tutti i risultati ottenuti.

Ambiente di lavoro - 1

Il progetto è stato interamente sviluppato in Python sfruttando principalmente la libreria **Keras.**

Keras è una libreria di alto livello in grado di lavorare al di sopra di altre librerie di più basso livello che ne costituisono il beck-end.

Come beck-end per questo progetto è stato utilizzato il framework **TensorFlow**.

Ambiente di lavoro - 2

TensorFlow è una libreria open source per il calcolo numerico ad altre prestazioni compatibile su diverse piattaforme (CPU, GPU, TPU), desktop, cluster di server, device mobili e periferici.

Ambiente di lavoro - 3

Per il training della rete è stato usato **COKA** (*Computing On Kepler Architectures - COKA*).

COKA è un cluster costruito dall'Università degli Studi di Ferrara con il supporto dell'Istituto Nazionale di Fisica Nucleare (INFN). Esso è composto da quattro nodi computazionale, ognuno dei quali è provvisto di due Intel Xeon CPUs e otto dual-GPU K80 NVIDIA.

Il picco delle prestazioni di calcolo sono dell'ordine di 100Tflops.

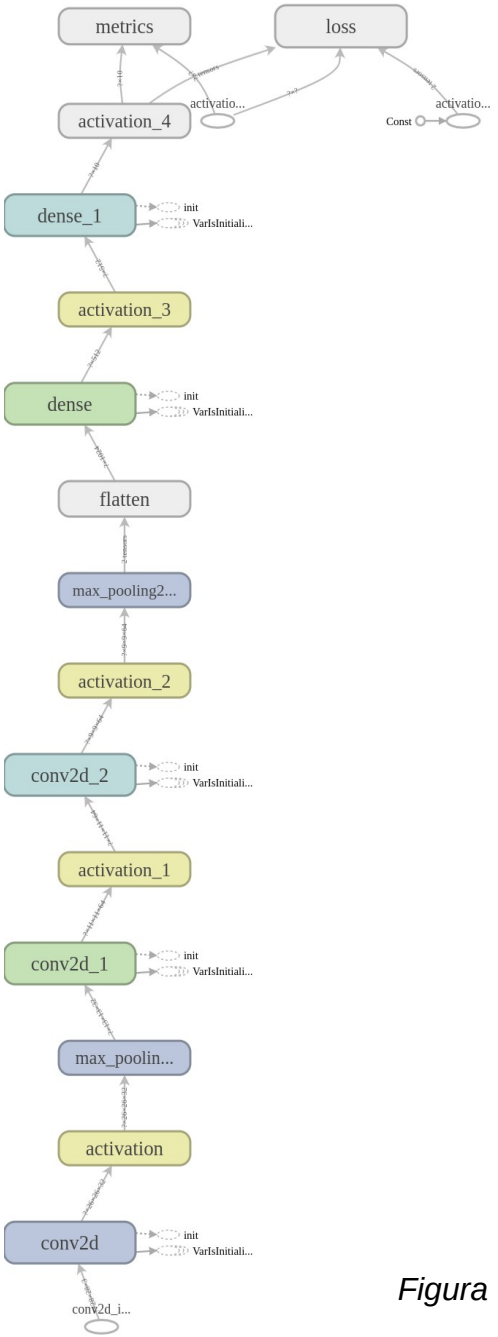
Sviluppo - 1

Figura 5 – Dataset.

La priva fase di sviluppo consiste nel labeling dei dati del dataset. La figura 5 mostra la divisione in dati di training e di test e la suddivisione in sottocategorìa.

Sviluppo - 2

La figura 5 mostra l’architettura della CNN sviluppata per questo progetto.



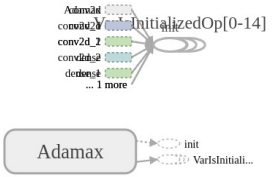


Figura 6 – Cnn sviluppata.

Sviluppo - 3

Parametri usati nello sviluppo:

- Funzione di attivazione: ReLU.
- Ottimizzatore: Adamax.
- Dati di training: 552.
- Dati di test: 230.
- Learning rate: 0.00247875
- Dimensione del Batch: 69.
- Epoche: 400.

Risultati - 1

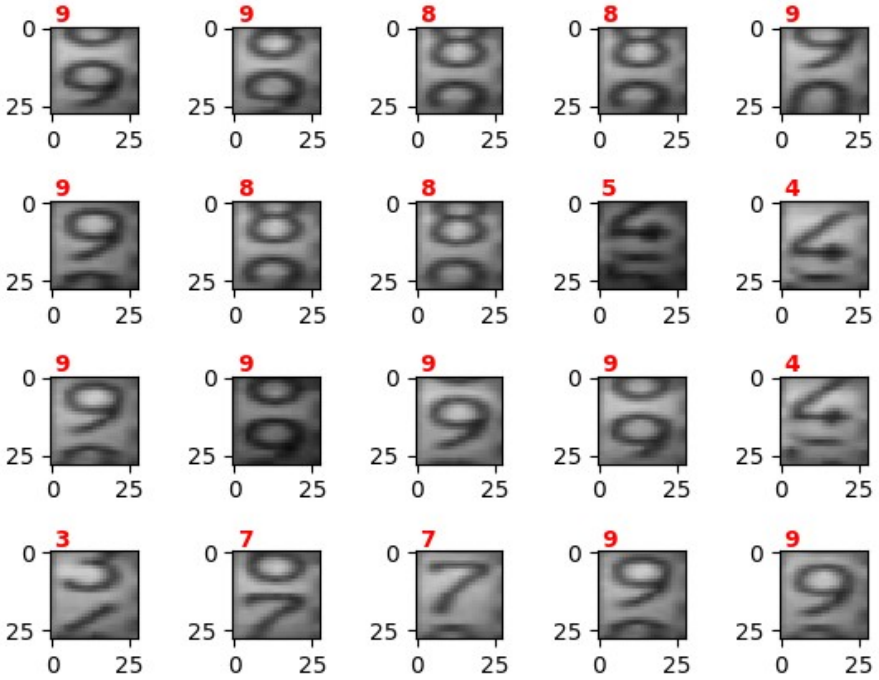


Figura 7 – Matrice di risultati.

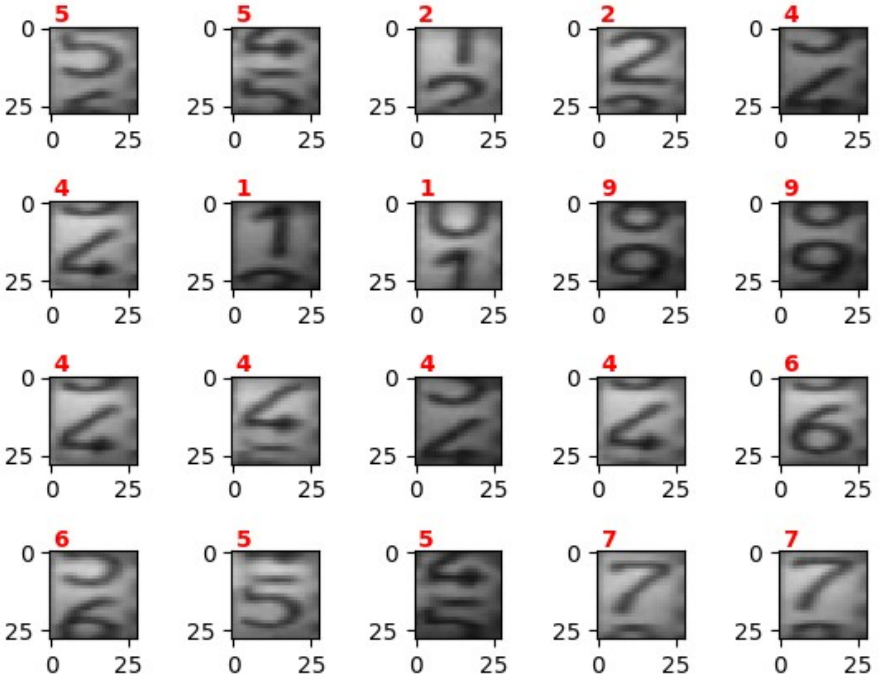


Figura 8 – Matrice di risultati.

Risultati - 2

Come si nota dalle figure 7 e 8 della slide precedente, ogni cifra è stata correttamente classificata.

Nelle situazioni intermedie, a meno di un netto riconoscimento da parte della rete, si è scelto di classificare la cifra più in basso nella fase di labeling iniziale.

Risultati - 3

Figura 9 – Andamento dell'accuratezza sui dati di training.

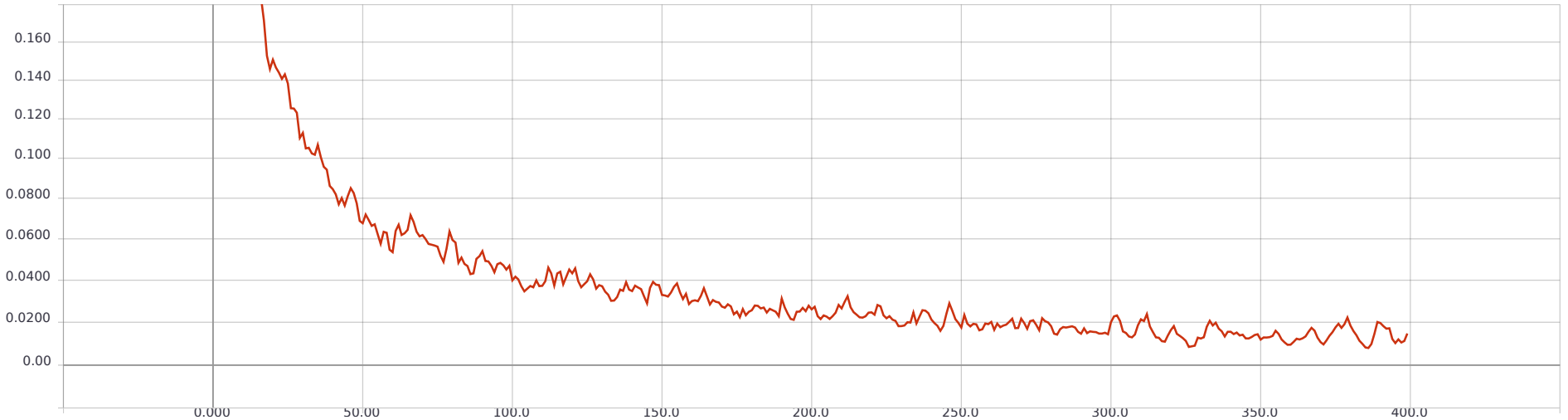


Figura 10 – Andamento della perdita sui dati di training.

Risultati - 4

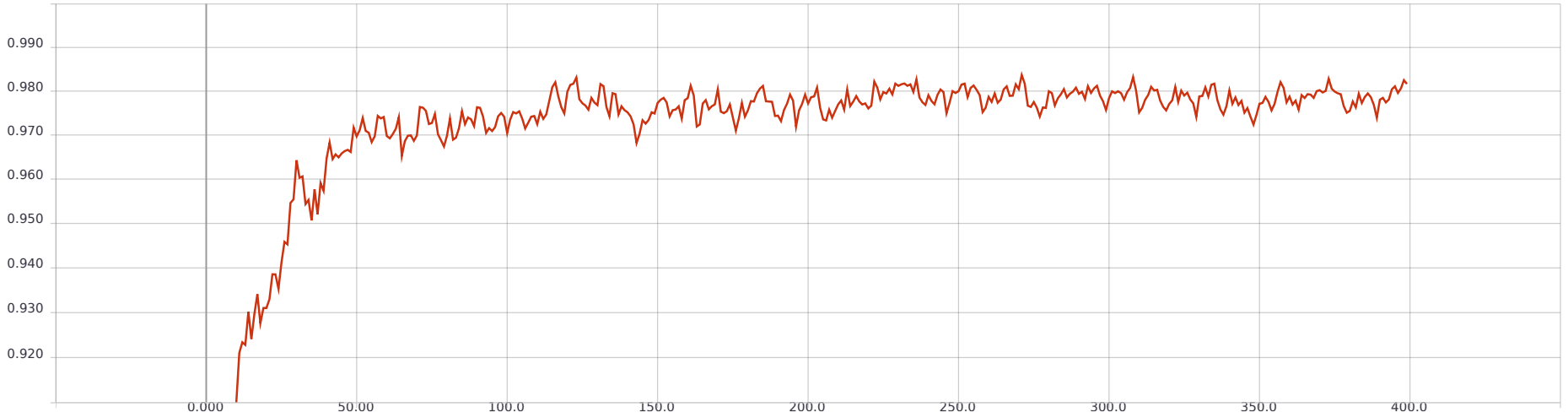


Figura 11 – Andamento dell'accuratezza sui dati di test.

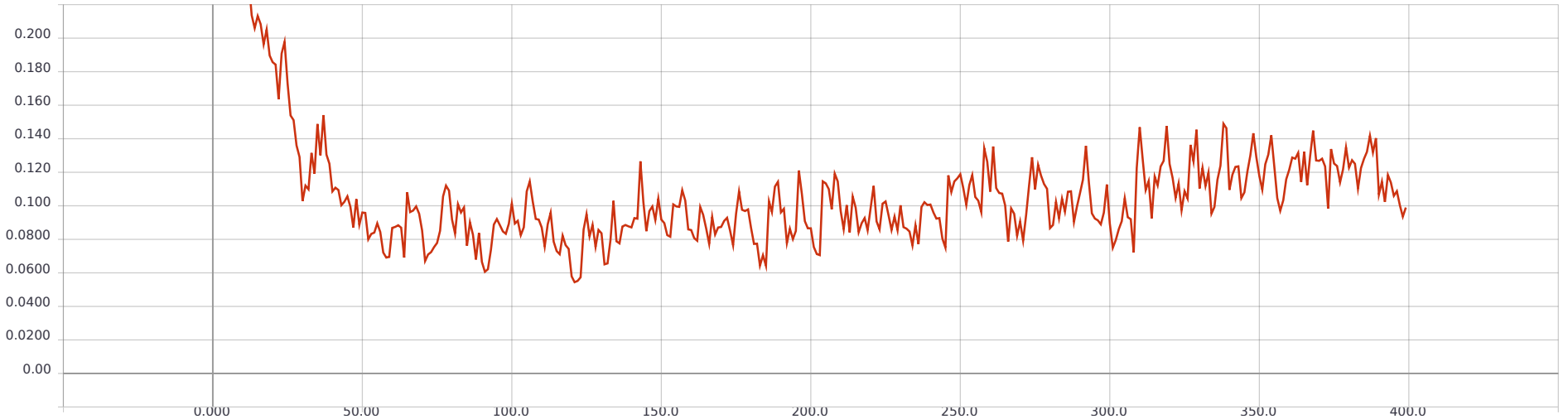


Figura 12 – Andamento della perdita sui dati di test.

Risultati - 5

- Accuratezza: 0.986956524848938 (~99%)
- Perdita: 0.095732490904629 (<10%)