# ML RELAZIONE

**Giovanni Oliverio (577999)**

**June 2024**

**GitHub**

**Abstract**

Questo progetto si concentra sull'implementazione di una rete neurale convoluzionale (CNN) per il riconoscimento in tempo reale di disegni realizzati dagli utenti su una "magic board". Utilizzando i landmark delle mani rilevati dalla libreria Mediapipe di Google, il sistema permette agli utenti di scrivere lettere direttamente davanti a una telecamera. La CNN, implementata con TensorFlow, analizza i segni scritti per identificare la lettera corrispondente, migliorando l'accuratezza del riconoscimento attraverso tecniche avanzate di deep learning. Questo studio include un'analisi approfondita dei risultati ottenuti, descrive l'architettura della CNN impiegata e suggerisce prospettive per futuri sviluppi concentrati sul miglioramento del riconoscimento dei caratteri disegnati sulla "magic board".

# Contents

# Introduzione

## 1.1 DataSet

Questo dataset è stato creato per il riconoscimento di forme scritte a mano, comprendendo una vasta collezione di immagini organizzate in file CSV contenente 26 classi (lettere dalla A alla Z). Ogni immagine ha dimensioni di 28X28 pixel, con ciascun carattere centrato in una scatola di 20X20 pixel e conservate in scala di grigi, con la possibilità di includere immagini con rumore.

Le immagini provengono principalmente dalla NIST e dal dataset esteso NMIST, insieme ad altre fonti selezionate, sono state preparate secondo specifiche dettagliate. Questo dataset originale conta oltre 370.000 immagini di caratteri dell'alfabeto inglese, presentando un significativo sbilanciamento tra le varie classi.

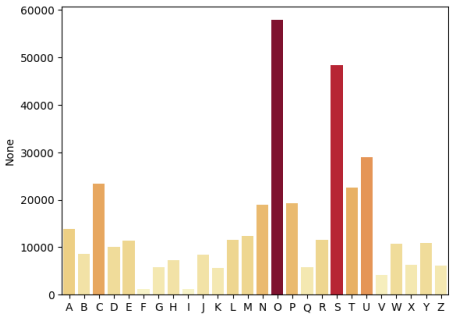


Figure 1: Sbilanciamento delle Classi

Il dataset è stato diviso in training set e test set con un rapporto 80-20, e le immagini sono state normalizzate per facilitare l'elaborazione dei dati durante le fasi di addestramento e valutazione dei modelli.

Questo dataset rappresenta una risorsa essenziale per lo sviluppo di algoritmi di riconoscimento ottico dei caratteri (OCR). La sua ricchezza e la vasta gamma di classi presenti stimolano gli sviluppatori a dare il massimo per trovare soluzioni efficaci al problema del riconoscimento di caratteri scritti a mano, contribuendo in modo significativo alla ricerca nel campo della visione artificiale e dell'intelligenza artificiale.

## 1.2 Tecnologie

Il mio progetto ha sfruttato diverse tecnologie chiave per raggiungere gli obiettivi prefissati. Ho scelto Python come linguaggio principale per la sua versatilità e la vasta gamma di librerie disponibili. Python mi ha permesso di sviluppare rapidamente e integrare diverse componenti del sistema.
Per quanto riguarda le reti neurali convoluzionali (CNN), ho adottato Keras come libreria principale, utilizzando TensorFlow come backend. Keras mi ha consentito di costruire e addestrare modelli di apprendimento automatico in modo efficiente, grazie alla sua chiara e potente API.
Ho implementato la gestione del flusso video e il riconoscimento delle forme scritte a mano utilizzando OpenCV. Questa libreria si è dimostrata essenziale per acquisire e manipolare le immagini necessarie per l'analisi visiva in tempo reale.
In particolare, ho integrato MediaPipe, sviluppata da Google, per tracciare la posizione dell'indice della mano da-

vanti alla telecamera. Questo mi ha permesso di implementare un sistema che consente agli utenti di disegnare con le dita sulla "magic board" virtuale. MediaPipe ha fornito strumenti avanzati per il rilevamento dei landmarks delle mani e ha facilitato il tracciamento preciso della posizione delle dita in tempo reale.

Complessivamente, l'integrazione di queste tecnologie mi ha fornito una base solida per sviluppare un sistema avanzato di riconoscimento delle forme scritte a mano e interazione con il "magic board" virtuale, garantendo al contempo un'esperienza utente fluida e intuitiva.

# Stuttura ReteNeurale Convoluzionale CNN

## 2.1 Teoria

Le reti neurali convoluzionali (CNN) sono un tipo avanzato di rete neurale artificiale, ispirato al modo in cui il cervello animale elabora le informazioni visive attraverso la retina. Le CNN utilizzano filtri convoluzionali, finestre mobili di dimensioni definite, per eseguire operazioni di convoluzione su piccole regioni dell'immagine, catturando dettagli come linee, bordi e texture, similmente al cervello umano nel processare dati visivi grezzi. Ogni filtro convoluzionale produce una mappa di attivazione, evidenziando le caratteristiche salienti dell'immagine e permettendo alla CNN di apprendere gerarchie di caratteristiche visive da concetti di basso livello, come linee e bordi, fino a forme e oggetti ad alto livello. Grazie all'efficace utilizzo dei filtri convoluzionali, le CNN sono strumenti potenti per l'analisi e l'elaborazione di immagini, impiegate in applicazioni come il riconoscimento di immagini, la segmentazione semantica e ecc.

## 2.2 Strutture Rete

La rete neurale convoluzionale inizia con un primo strato convoluzionale utilizzando 32 filtri 3x3 con attivazione ReLU, seguito da un layer di MaxPooling 2x2 per ridurre le dimensioni dell'output mantenendo le caratteristiche principali. Il secondo strato convoluzionale impiega 64 filtri 3x3 con la stessa attivazione ReLU e un ulteriore MaxPooling 2x2. L'output dei livelli convoluzionali viene appiattito in un vettore tramite il layer di Flatten. Successivamente, il vettore appiattito viene processato da un layer completamente connesso composto da 128 neuroni con attivazione ReLU, il quale apprende relazioni complesse tra le feature estratte. Il layer di output finale consiste in 26 neuroni con attivazione softmax, generando una distribuzione di probabilità sulle 26 lettere dell'alfabeto inglese.

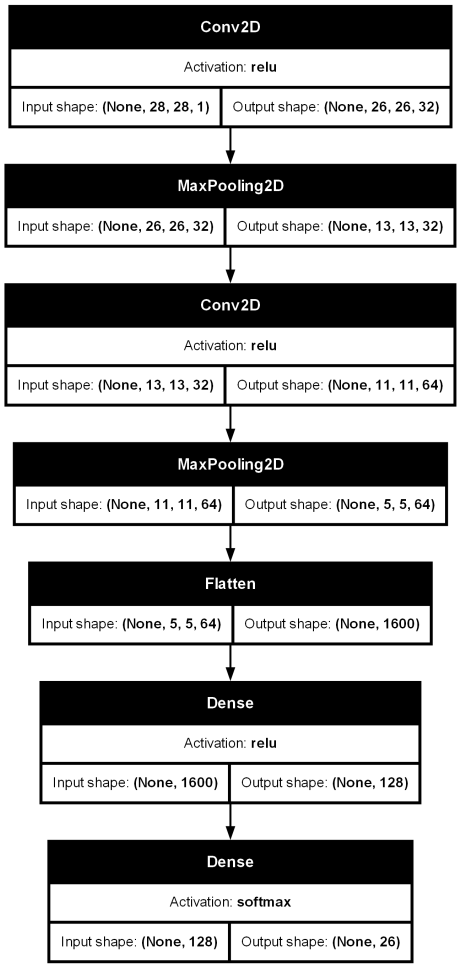


Figure 2: Strutture della Rete

Durante l'allenamento, i pesi dei filtri convoluzionali e dei neuroni nei layer completamente connessi vengono ottimizzati tramite l'algoritmo ADAM, una tecnica di ottimizzazione derivata dallo stochastic gradient descent. ADAM adatta autonomamente i tassi di apprendimento per ogni parametro, basandosi sulle stime del primo e secondo momento dei gradienti. Questo metodo accelera la convergenza dell'addestramento e può adattarsi a diversi tipi di problemi di ottimizzazione.

La retropropagazione è il cuore dell'addestramento di una rete neurale, dove l'errore tra le predizioni del modello e le etichette reali viene propagato all'indietro attraverso la rete. Questo processo calcola il gradiente dell'errore rispetto ai pesi del modello, consentendo ad ADAM di aggiornarli in modo appropriato per ridurre gradualmente l'errore di previsione. La categorical-crossentropy è una funzione di perdita comune utilizzata per problemi di classificazione multiclasse, misurando la discrepanza tra le distribuzioni di probabilità predette e quelle reali delle classi.

# Analisi dei Risultati

## 3.1 Risultato

La CNN si è dimostrata subito una rete eccellente, raggiungendo un'accuratezza del 99% e un test loss di soli 0.02. Questo indica che il modello è molto resistente all'overfitting, sebbene non completamente immune. Questo è evidente nei grafici che mostrano l'andamento delle metriche di valutazione del modello nel corso delle epoche.

Il primo grafico mostra l'andamento degli errori di training e di test, mentre il secondo illustra l'andamento dell'accuratezza di training e di test. Entrambi i grafici evidenziano come le metriche del modello evolvono nel tempo.

È possibile osservare un leggero overfitting nei grafici, poiché le curve del training (in blu) sono leggermente migliori rispetto a quelle del test (in arancione), suggerendo che il modello performa meglio sui dati di training rispetto ai dati di test.

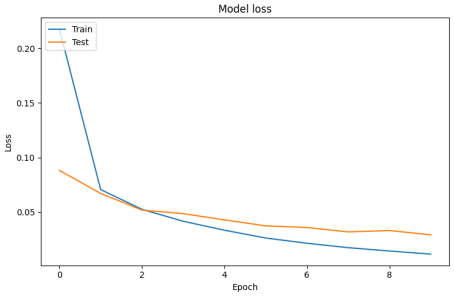


Figure 3: Plot Epoca su Loss

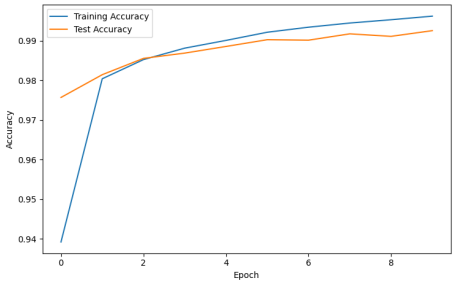


Figure 4: Plot Epoca su Accurasy

## 3.2 Analisi Predizioni

Dopo aver esplorato ulteriormente la CNN e aver effettuato test visivi più approfonditi, si conferma come un'ottima scelta per il mio progetto. Prendendo dei campioni di esempio, ho potuto osservare direttamente quanto sia preciso nella risoluzione di quella task specifica. Il modello è in grado di fornire risposte praticamente certe sulla natura delle lettere che analizza, sia per le lettere presenti nel dataset originale che per lettere disegnate ex novo a mano da me.

Questi test visivi hanno evidenziato la capacità della CNN di generalizzare bene anche a nuove varianti di lettere, confermando la robustezza del modello. come dimmostrato dal grafico che vi mostrerò, la CNN è in grado di mantenere alte performance di predizione anche su dati non visti durante il training.

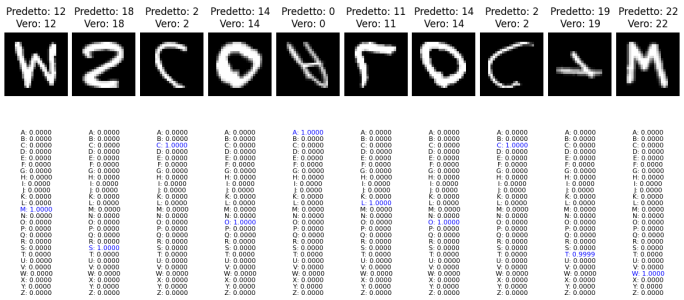


Figure 5: Previsioni di alcuni campioni con relative probabilità

Inoltre, ho utilizzato la Confusion Matrix per valutare le prestazioni del modello. Questo strumento fornisce una rappresentazione dettagliata delle predizioni del modello rispetto alle vere etichette di classe. Ogni cella della matrice mostra il numero di casi in cui il modello ha predetto correttamente o erroneamente ciascuna classe. La Confusion Matrix è fondamentale per valutare la precisione e il recall del modello su diverse classi, permettendo di identificare con precisione quali classi il modello confonde più frequentemente e dove potrebbe esserci spazio per miglioramenti.

Utilizzare la Confusion Matrix è cruciale per interpretare e ottimizzare le prestazioni del modello di classificazione, fornendo una guida dettagliata per il miglioramento continuo della sua capacità predittiva.

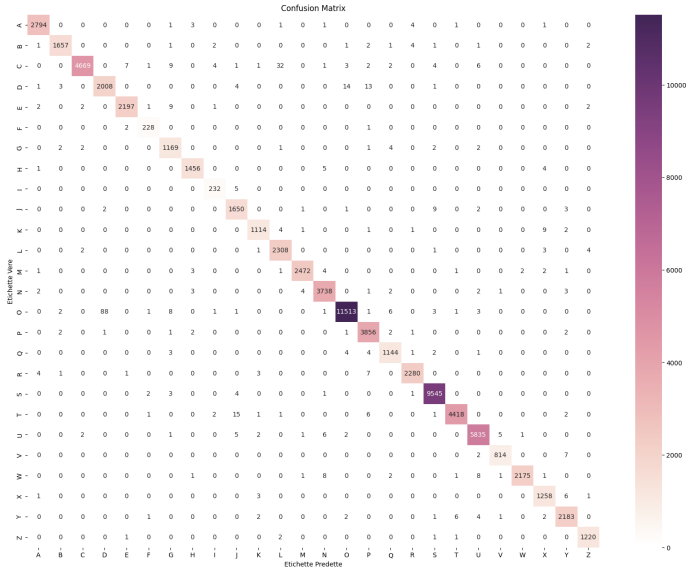


Figure 6: Confusion Matrix

# Implementazione Magic Board

## 4.1 Disegno

La Magic Board, per essere implementata, ho utilizzato principalmente due librerie, MediaPipe e cv2. Con MediaPipe, ho adoperato la soluzione handmarks per rilevare i punti di riferimento delle mani in un'immagine. Questo approccio sfrutta un modello di machine learning ottimizzato anche per un flusso continuo di dati, permettendomi di ottenere punti di riferimento precisi sulla mano con questo patten:

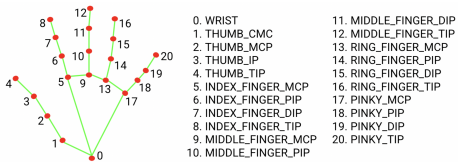


Figure 7: patten hand landmarks

questi landmarks li ho sfruttati principalmente in questi due modi:

1→ Per creare un riconoscitore di gesti molto basilare, utilizzo l'interpolazione tra i vari punti delle dita e calcolo le distanze reciproche per determinare se un dito è aperto o chiuso. Ciclo attraverso ogni dito per generare un array di 5 elementi che indica:

$$\begin{cases} 0 & \text{se il dito è aperto} \\ 1 & \text{se il dito è chiuso} \end{cases}$$

per ogni mano. Questo mi permette di capire la posizione della mano e attivare modalità di disegno tramite semplici condizioni if come i seguenti.

```
# la funzione di cui parlavo prima
posizione = dita_alzate(punte, iniziodita, landmarks)
if posizione == [1, 1, 0, 0, 0]:
    draw_mod_on = True
    #Attiva La DrawMode
if posizione == [1, 1, 1, 1, 1]:
    #Pulisce l'intero Canvas
```

Listing 1: Esempio di codice Python

2→ Il sistema principale per il funzionamento della modalità di disegno (draw mode) si basa sull'uso della punta dell'indice come riferimento. Utilizzando il punto numero 8 nei landmarks della mano (handmarks) di MediaPipe, si traccia il disegno sul canvas. Per creare il disegno, si tracciano delle linee che collegano la posizione attuale dell'indice con quella del frame precedente. Questo metodo permette di ottenere un tratto continuo che segue i movimenti dell'indice.

Invece CV2 lo usato per trovare il contorno delle figure tramite l'usilio della funzione

```
gray = cv2.cvtColor(imagecanvas, cv2.COLOR_BGR2GRAY) # convertiva il canvas in scala di grigi
# Applica una soglia binaria all'immagine grigia
# threshold: valore di soglia
```

```
4 # Se il valore di un pixel e'
      superiore a 'threshold',
      viene impostato a 255 (
      bianco)
5 # Se il valore di un pixel e'
      inferiore o uguale a '
      threshold', viene impostato
      a 0 (nero)
6 _, threshold = cv2.threshold(
      gray, 1, 255, cv2.
      THRESH_BINARY)
7 #attraverso la soglia Binaria di
       threshold trova i bordi e
      ne crea una gerarchia
8 contours, hierarchy = cv2.
      findContours(threshold, cv2.
      RETR_TREE, cv2.
      CHAIN_APPROX_SIMPLE)
```

Listing 2: Esempio di codice Python

per disegnare sul canvas fornendo le funzioni che tracciano le linee di qui ho parlato prima per gestire il flusso di immagini in diretta che venivano da 3 windows necesarie per la "magic board"

1→ La prima finestra è quella gestita da VideoCapture, che riceve e gestisce tutti i dati di input provenienti dalla telecamera. È questa finestra che Mediapipe sfrutta per tracciare i landmarks delle mani (handmarks).

2→ La Seconda finestra, chiamata canvas con sfondo nero, è quella su cui si disegnano e si estraggono i contorni delle figure, nonché le regioni di interesse per l'elaborazione e la previsione.

3→ La Terza finestra è quella combinata tra il canvas e quella di input, ed è quella che l'utente finale vedrà effettivamente. È su questa finestra che vengono stampate le previsioni o i risultati dell'elaborazione.

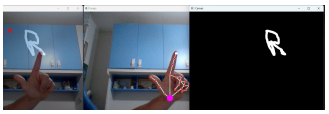

Figure 8: ord. combinata input canvas

## 4.2 Predizione

Nel progetto ci sono due tipi di predizioni, entrambe effettuate dallo stesso modello descritto in precedenza, ciascuna con le proprie differenze nella preparazione del dato grezzo prima di essere fornito al modello.

1→ ObjectRecognizer, il più semplice dei due, ma efficiente riconosce una figura alla volta. Identifica il primo contorno, ne individua i 4 vertici, centra la figura e la rende quadrata. Successivamente, pre-elabora i dati convertendo l'immagine in scala di grigi, ridimensionandola per adattarla al modello e normalizzandola (dividendo per 255). Effettua quindi la previsione e visualizza il risultato nella finestra combinata in alto a sinistra, sotto la scritta che indica se si sta disegnando. Questo sistema è stato utile per testare l'efficacia del modello sui nuovi dati. L'evoluzione del modello è visibile nelle cartelle delle immagini post-work, organizzate per lettera con le relative previsioni.

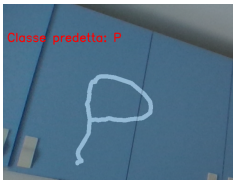


Figure 9: ObjectRecognizer

2→ ObjectDetector si differenzia da ObjectRecognizer nel modo in cui identifica i contorni all'interno di una box. Si concentra esclusivamente sui contorni con la gerarchia più alta, ignorando quelli di gerarchia inferiore che potrebbero essere contenuti all'interno della stessa box. Questa scelta rende più complicata l'estrazione delle immagini, che ora è gestita da una funzione separata. Questa funzione restituisce una tupla contenente una lista di tuple di vertici e una lista di immagini preprocessate. Per migliorare le prestazioni, specialmente in termini di framerate durante la classificazione delle immagini, è stato implementato un worker che opera in parallelo. Questo worker utilizza una coda per restituire i risultati, permettendo al ciclo principale di continuare senza interruzioni mentre il modello esegue le sue elaborazioni.
il worker viene gestito da ThreadPoolExecutor

```
# worker
def process_canvas(imagecanvas, model, queue):
    global cancella
    if not cancella:
        drawn_parts, drawn_part_normalizeds = estrai_parte_disegnata(imagecanvas)
        classi_predette = esegui_previsioni(model, drawn_part_normalizeds)
        queue.put((drawn_parts, classi_predette))
```

Listing 3: Esempio di codice Python

Successivamente, il queue aggiornato viene letto nel ciclo principale e le informazioni al suo interno vengono passate alle funzioni gestite da OpenCV (cv2). Queste funzioni iterano su tutti gli array di dati e stampano le box. Inoltre, in basso a destra di ogni box viene aggiunta la scritta corrispondente alla previsione.
Un'ultima cosa che ho fatto diversamente dal Detector è cercare di far lavorare il worker il meno possibile. Si attiva solo quando stai disegnando, ma quando cancelli, visto che non mi interessano più i risultati che arrivano, blocco l'aggiornamento della coda finché il worker non ha finito di elaborare. Altrimenti, dopo la cancellazione, potrebbe verificarsi un aggiornamento della coda con dati vecchi e riapparire le vecchie box.

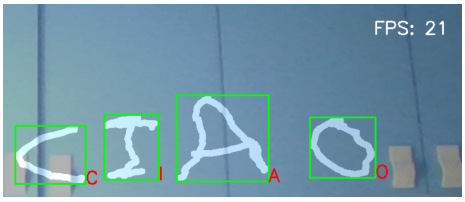


Figure 10: ObjectDetector

# Conclusione

Sono estremamente soddisfatto di vedere come l'idea che ho concepito si sia concretizzata nel corso dello sviluppo del progetto. Questo progetto esplora le potenzialità dell'intelligenza artificiale nel riconoscimento delle lettere scritte dagli utenti su una magic board. Nonostante alcune sfide e opportunità di miglioramento, il risultato è stato straordinariamente positivo, dimostrando l'accuratezza del modello nelle previsioni all'interno di un contesto pratico di utilizzo.

## 5.1 Sviluppi Futuri

Un ampio spazio di miglioramento si presenta nei futuri sviluppi del progetto.
Ad esempio, potrebbe essere valutato l'aumento delle classi, come l'inclusione di caratteri di punteggiatura, numeri e altri caratteri speciali, o l'implementazione di alfabeti diversi. Questo potrebbe complicare il modello attuale, rendendolo in grado di riconoscere una gamma più ampia di caratteri. Allo stesso tempo, potrebbe essere utile esplorare modelli più efficienti per la classificazione multi-oggetto su un canvas, ad esempio utilizzando reti neurali ricorrenti per comprendere il contesto delle lettere all'interno di frasi o equazioni complesse. Questa evoluzione potrebbe consentire alla magic board non solo di riconoscere e interpretare le espressioni scritte, ma anche di risolvere equazioni matematiche, trasformandosi così in un valido sostituto delle costose lavagne multimediali. In sintesi, il progetto ha un potenziale significativo per crescere e innovare nel campo dell'interazione uomo-macchina attraverso l'intelligenza artificiale, aprendo nuove possibilità nell'educazione e nella collaborazione digitale.

## 5.2 Bibliografia

1→ Corso di Machine Learning e Deep Learning A.A. 2023/2024, Universit'a degli studi Roma TRE

2→ DataSetInfo

3→ MediaPipe

4→ CV2