Contents

Real-Time Sketch Recognition using Neural Networks

Autore: Tommaso Senatori
Corso : Machine Learning A.A. 2023/2024
Università : Università degli studi Roma TRE
Github : https://github.com/tommasosenatori/ML_Real-Time-Sketch-Recognition

0 Abstract

Questo progetto si concentra sull'applicazione di reti neurali, inclusa una rete neurale convoluzionale (CNN) e una rete neurale (NN) sviluppate con TensorFlow, oltre a una NN costruita da zero, per il riconoscimento in tempo reale di disegni realizzati dagli utenti su un canvas interattivo, offrendo un'analisi dei risultati, un'implementazione di un'interfaccia grafica e prospettive future per migliorare l'efficacia del sistema. La Sezione 2 esamina le tecnologie impiegate nella realizzazione del progetto. La Sezione 3 fornisce una dettagliata descrizione del dataset utilizzato. La Sezione 4 tratta l'architettura delle reti neurali coinvolte. La Sezione 5 analizza in dettaglio i risultati ottenuti. La Sezione 6 illustra l'implementazione dell'Interfaccia Grafica (GUI). La Sezione 7 prospetta sviluppi futuri per il progetto, mentre la Sezione 8 riporta le conclusioni. Infine, nella sezione della Bibliografia, sono elencate le fonti utilizzate per il progetto.

1 Introduzione

Questo progetto prende ispirazione dal popolare gioco "Quick, Draw!" per creare un'applicazione che si concentra sull'utilizzo di reti neurali per il riconoscimento in tempo reale di disegni realizzati dagli utenti su un canvas interattivo. "Quick, Draw!" è un gioco sviluppato da Google, che è stato lanciato come esperimento AI ed è stato ampiamente giocato e apprezzato in tutto il mondo. In "Quick, Draw!", gli utenti sono sfidati a disegnare una serie di oggetti o concetti specifici in un breve lasso di tempo, solitamente 20 secondi. Mentre disegnano, il sistema tenta di indovinare ciò che l'utente sta rappresentando in tempo reale. Questo gioco coinvolgente ha dimostrato quanto sia divertente e istruttivo il riconoscimento delle immagini basato su reti neurali, aprendo la strada all'applicazione di questa tecnologia in diversi campi. L'obiettivo principale di questo progetto è consentire agli utenti di disegnare liberamente, su un canvas virtuale, una delle 20 categorie di immagini presenti nel dataset. Dopo aver completato il disegno, il sistema fornisce previsioni in tempo reale su ciò che è stato raffigurato, offrendo un'esperienza coinvolgente e creativa agli utenti. Il progetto sfrutta tre diverse reti neurali: una rete neurale convoluzionale (CNN) e una rete neurale standard (NN) create con TensorFlow, e anche una NN costruita da zero senza l'uso di librerie esterne. Ogni rete è stata addestrata sui dati raccolti da "Quick, Draw!" e offre una straordinaria opportunità per esplorare il riconoscimento delle immagini in tempo reale.

2 Tecnologie Utilizzate

Per la realizzazione del progetto, sono state utilizzate diverse tecnologie che hanno contribuito al suo sviluppo e alla sua funzionalità. In particolare, il linguaggio di programmazione scelto è stato Python, per la sua flessibilità, facilità d'uso e vasta gamma di librerie disponibili per il Machine Learning. Per la creazione e l'addestramento delle reti neurali convoluzionali (CNN) e non (NN), è stata utilizzata la libreria Keras con il backend TensorFlow. Keras è una API (Application Programming Interface) open-source di alto livello che mette a disposizione dei componenti fondamentali sulla cui base si possono sviluppare modelli complessi di apprendimento automatico, e presenta interfacce pronte all'uso che permettono un accesso rapido e intuitivo al rispettivo back end. Keras con TensorFlow rende la creazione dei Neural Network estremamente intuitiva, consentendo agli sviluppatori di concentrarsi sulla progettazione dei modelli senza dover affrontare

complessità implementative. Per l'implementazione dell'interfaccia grafica per l'utente (GUI), è stata utilizzata Tkinter, una libreria di Python che consente di creare per l'appunto interfacce grafiche interattive e intuitive. In questo progetto, è stata usata per permettere all'utente di disegnare una tra le 20 tipologie di immagini e ricevere in tempo reale una predizione dal modello.

3 Descrizione Dataset

Il dataset in questione è stato preso da Quick, Draw!, un gioco online in cui ai giocatori viene dato un oggetto specifico da disegnare entro un limite di tempo. Il dataset di questo gioco è una vastissima collezione di oltre 50 milioni di sketch disegnati da utenti di tutto il mondo suddivisi in categorie, che vanno dagli oggetti di uso quotidiano agli animali, alle piante e altro ancora. Il dataset fornisce una preziosa risorsa per l'addestramento e la valutazione dei modelli di apprendimento automatico nel campo del riconoscimento delle immagini e della comprensione del linguaggio naturale, oltre a coinvolgere gli utenti in modo divertente e interattivo. Nel contesto di questo progetto, è stata effettuata una selezione specifica del dataset, concentrandosi su 20 classi di immagini. Per ogni classe, sono state scelte 8000 immagini, per un totale di 160000 immagini. Questa scelta è stata motivata da diverse ragioni. Innanzitutto, lavorare con un dataset più piccolo, riducendo il numero di classi e il numero di immagini per classe, ha permesso di gestire meglio le limitazioni di risorse computazionali come la potenza di calcolo, la memoria e il tempo di addestramento. Questo ha reso possibile un processo di addestramento più efficiente e veloce. È importante sottolineare che, nonostante la riduzione del numero di classi e delle immagini per classe, il dataset selezionato è ancora rappresentativo delle caratteristiche distintive di ciascuna classe. Le 20 classi di immagini sono: airplane, apple, banana, cactus, car, clock, cloud, door, eye, fish, fork, ice cream, line, lollipop, octopus, pencil, smiley face, star, sun, umbrella. Il training set e il test set sono stati divisi con un rapporto 70-30, e successivamente normalizzati (i valori delle feature sono scalati in un intervallo simile, tra 0 e 1) in modo da velocizzare l'esecuzione dei programmi e da non distorcere differenze negli intervalli o perdere informazioni.

4 Struttura Reti

4.1 Rete CNN

La rete inizia con un layer di reshape che trasforma l'input da un vettore piatto a una matrice bidimensionale. Questo passaggio è essenziale per abilitare l'applicazione dei filtri convoluzionali successivi. Il layer successivo utilizza 32 filtri convoluzionali di dimensione 3x3, e l'applicazione delle attivazioni ReLU introduce la componente non lineare nel modello. I filtri di una CNN servono per riconoscere pattern nei dati di input. In generale, i filtri nei primi layer tendono a riconoscere pattern più semplici, come linee e forme elementari, mentre quelli presenti in layer più profondi possono individuare pattern notevolmente più complessi. Successivamente, viene utilizzato un layer di max pooling, che opera prendendo il valore massimo in finestre 2x2. Questo processo riduce la dimensione spaziale dell'immagine, mantenendo le caratteristiche salienti e contribuendo alla riduzione del numero complessivo di parametri nel modello. La sequenza Conv2D e MaxPool2D è ripetuta una seconda volta, e in seguito, l'output viene appiattito mediante un layer di tipo Flatten. Questa operazione converte l'output in un vettore unidimensionale, preparandolo per l'input ai due successivi Dense layer, di cui l'ultimo funge da layer di output. Quest'ultimo strato utilizza l'attivazione softmax, che normalizza le probabilità delle diverse classi in modo che la loro somma sia sempre uguale a 1, consentendo così la classificazione dell'immagine in una delle 20 classi possibili.

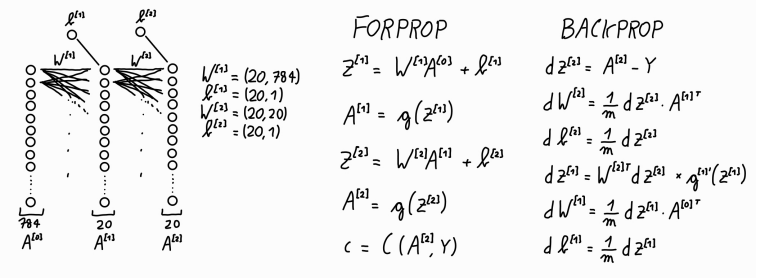
4.2 Rete NN

Questa rete presenta una struttura semplice, con due hidden layers composti da 128 neuroni ciascuno e funzioni di attivazione ReLU per introdurre non linearità, seguiti dall'output layer con la classica attivazione softmax che è stata usata in tutte le reti neurali del progetto. Nonostante la rete non sia stata progettata con una grande profondità per evitare problemi di overfitting, è emersa la necessità di mitigare tale problema aggiungendo due layer di dropout tra i due hidden layers. Questa tecnica del Dropout aiuta a prevenire l'overfitting disattivando casualmente una

percentuale dei neuroni per ciascun esempio di training, rendendo così i neuroni più adattabili e riducendo il rischio che diventino troppo specializzati su dati specifici di addestramento.

4.3 Rete da 0

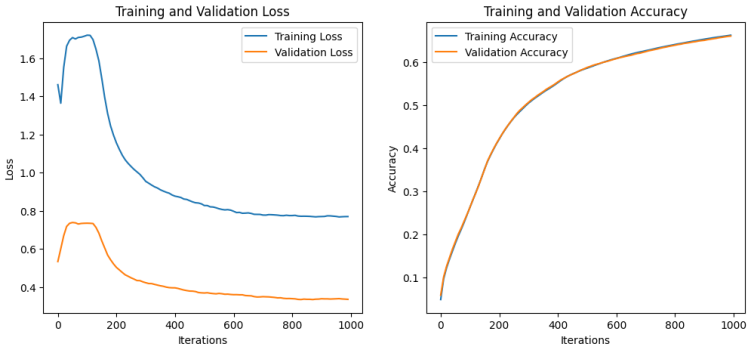
Nonostante l'ampia disponibilità e la facilità d'uso di Tensorflow per la costruzione di reti neurali, è stato ritenuto importante esplorare in dettaglio il funzionamento interno delle reti neurali. Questo perché affrontando direttamente le equazioni e costruendo la rete neurale da zero, si sviluppa una comprensione più concreta del funzionamento intrinseco della rete stessa, in contrasto con l'utilizzo di funzioni di cui non si ha conoscenza dettagliata della loro implementazione interna. Quest'ultima rete infatti è stata creata senza l'uso di nessuna libreria (a parte NumPy). La sua architettura è composta esclusivamente da tre strati: un layer di input per le immagini in ingresso, un hidden layer con 20 neuroni e funzione di attivazione ReLU, e un output layer con attivazione Softmax. L'implementazione del Neural Network si articola in diverse fasi. Innanzitutto, si procede con l'inizializzazione casuale dei parametri del modello, compresi i pesi e i bias. Successivamente, durante la fase di Forward Propagation, le immagini di input vengono elaborate strato dopo strato, passando attraverso la rete neurale per generare un output.



La Backward Propagation è un passaggio cruciale in cui si valuta quanto l'output predetto dal modello si discosti dai valori reali (errore o loss). Successivamente, si torna indietro attraverso la rete per determinare quanto ciascuno dei pesi e dei bias precedenti abbia contribuito a quell'errore. Questi pesi e bias vengono quindi aggiornati in modo da ridurre l'errore complessivo, migliorando progressivamente le prestazioni della rete durante il processo di training. La funzione gradient_descent svolge un ruolo fondamentale nel calcolo dei gradienti dei pesi e dei bias durante ogni epoca di addestramento. Questi gradienti vengono poi utilizzati per aggiornare i parametri del modello utilizzando un tasso di apprendimento (alpha) specifico. Questo processo si ripete per un numero definito di epoche (iterations), con l'obiettivo di ottimizzare i parametri della rete al fine di minimizzare la loss e massimizzare l'accuracy.

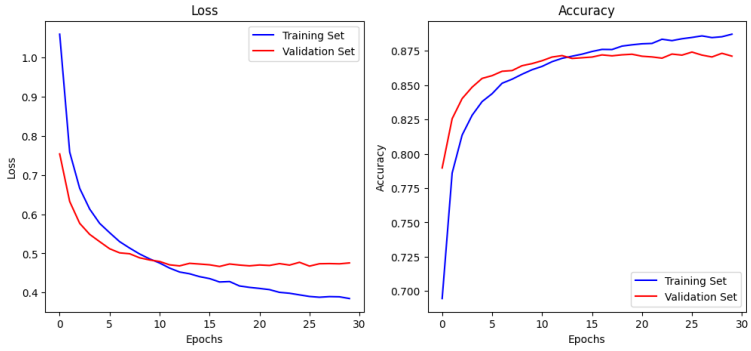
5 Analisi dei Risultati

5.1 Rete da 0



Nonostante l'accuracy finale di solo 0.67, si può osservare come training e validation set abbiano delle accuracy che vanno praticamente di pari passo dall'inizio alla fine, mostrando un andamento a derivata sempre positiva.

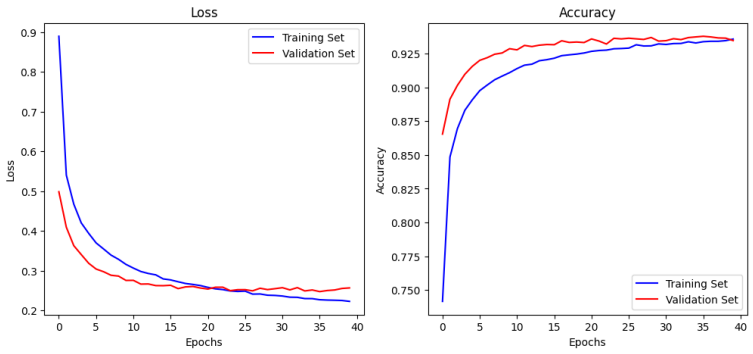
5.2 Rete NN



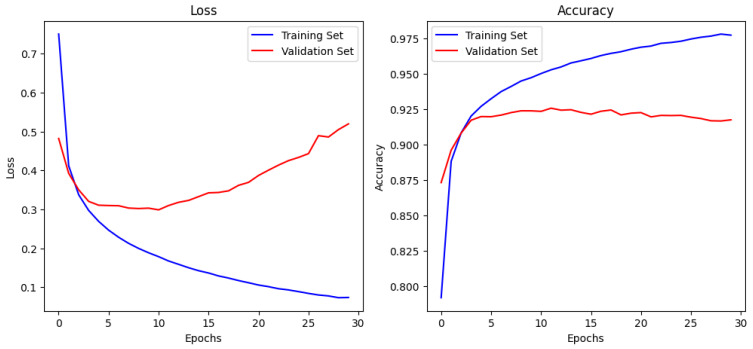
I risultati di questa rete dimostrano l'efficacia dei layer di dropout nel ridurre l'overfitting, poiché si osserva una differenza minima nell'accuracy tra i set di training e validation. Inoltre, i grafici delle loss mostrano una tendenza costantemente decrescente, al netto di fluttuazioni minori. Il numero di epoche è stato fissato a 18, in quanto le accuracy sono già relativamente alte e poco distanti tra loro.

5.3 Rete CNN

Con Dropout



Senza Dropout



I grafici mostrano le differenze tra l'utilizzo e l'assenza dei layer di Dropout. È evidente come il Dropout svolga un ruolo cruciale nel prevenire l'overfitting, poiché nel grafico senza Dropout si nota un aumento significativo della differenza tra le accuracy del training e del validation set dopo la quinta epoca. Nel grafico con il Dropout, invece, la differenza tra le accuracy rimane estremamente bassa, con uno scarto di circa 10^{-4} nelle ultime epoche. Nel modello senza Dropout, il validation loss raggiunge il suo minimo globale intorno alla quinta epoca e inizia ad aumentare dopo questo punto, segnalando l'inizio dell'overfitting. Nonostante ciò, è stata scelta la rete senza Dropout con un numero di epoche fissato a 5 (prima che l'overfitting iniziasse) per due motivi principali. In primo luogo, questo approccio consente di risparmiare risorse computazionali significative. In secondo luogo, questa configurazione del modello offre prestazioni superiori per i disegni realizzati tramite la GUI finale. Una possibile spiegazione per questo fenomeno

potrebbe essere che il modello senza Dropout, avendo raggiunto valori di accuratezza più elevati nelle prime 5 epoche, potrebbe essere in grado di catturare con maggiore precisione i dettagli specifici dei disegni creati dagli utenti sulla GUI finale, il che si traduce in previsioni più accurate per queste immagini particolari.

5.4 Metriche delle Reti: Classification Report e Confusion Matrix

Classification report:

	precision	recall	f1-score	support
airplane	0.46	0.48	0.47	2350
apple	0.80	0.83	0.82	2350
banana	0.64	0.65	0.65	2383
cactus	0.47	0.43	0.45	2395
car	0.65	0.70	0.67	2390
clock	0.76	0.78	0.77	2405
cloud	0.64	0.60	0.62	2347
door	0.85	0.85	0.85	2395
eye	0.53	0.55	0.54	2434
fish	0.70	0.72	0.71	2491
fork	0.68	0.62	0.65	2325
ice cream	0.66	0.73	0.69	2363
line	0.84	0.82	0.83	2449
lollipop	0.77	0.77	0.77	2455
octopus	0.60	0.64	0.62	2446
pencil	0.62	0.67	0.65	2393
smiley face	0.64	0.58	0.61	2452
star	0.57	0.56	0.57	2435
sun	0.55	0.50	0.53	2346
umbrella	0.76	0.74	0.75	2396
accuracy			0.66	48000
macro avg	0.66	0.66	0.66	48000
weighted avg	0.66	0.66	0.66	48000

Classification report:

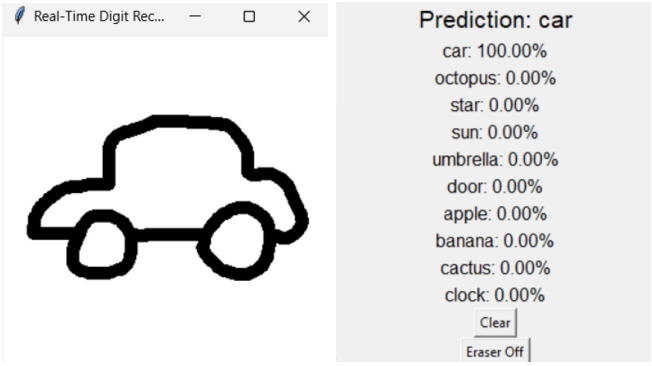
	precision	recall	f1-score	support
airplane	0.76	0.82	0.79	2350
apple	0.91	0.94	0.92	2350
banana	0.88	0.90	0.89	2383
cactus	0.82	0.76	0.79	2395
car	0.89	0.86	0.87	2390
clock	0.90	0.89	0.89	2405
cloud	0.82	0.82	0.82	2347
door	0.96	0.94	0.95	2395
eye	0.84	0.85	0.84	2434
fish	0.86	0.85	0.86	2491
fork	0.87	0.86	0.86	2325
ice cream	0.92	0.92	0.92	2363
line	0.94	0.93	0.93	2449
lollipop	0.93	0.90	0.92	2455
octopus	0.83	0.83	0.83	2446
pencil	0.84	0.90	0.87	2393
smiley face	0.91	0.82	0.86	2452
star	0.74	0.87	0.80	2435
sun	0.84	0.84	0.84	2346
umbrella	0.94	0.89	0.91	2396
accuracy			0.87	48000
macro avg	0.87	0.87	0.87	48000
weighted avg	0.87	0.87	0.87	48000

Classification report:

	precision	recall	f1-score	support
airplane	0.82	0.88	0.85	2350
apple	0.96	0.95	0.96	2350
banana	0.96	0.92	0.94	2383
cactus	0.86	0.89	0.88	2395
car	0.90	0.92	0.91	2390
clock	0.95	0.94	0.95	2405
cloud	0.87	0.91	0.89	2347
door	0.95	0.97	0.96	2395
eye	0.90	0.91	0.91	2434
fish	0.95	0.88	0.91	2491
fork	0.89	0.89	0.89	2325
ice cream	0.98	0.93	0.95	2363
line	0.96	0.92	0.94	2449
lollipop	0.94	0.93	0.94	2455
octopus	0.88	0.93	0.91	2446
pencil	0.91	0.92	0.91	2393
smiley face	0.94	0.92	0.93	2452
star	0.89	0.88	0.89	2435
sun	0.93	0.93	0.93	2346
umbrella	0.94	0.95	0.94	2396
accuracy			0.92	48000
macro avg	0.92	0.92	0.92	48000
weighted avg	0.92	0.92	0.92	48000

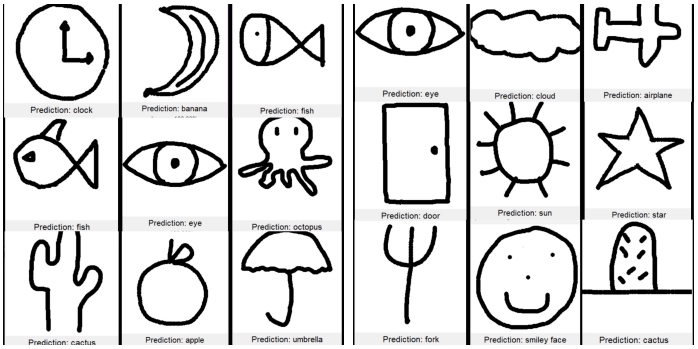
Nei Classification Report, le metriche di Precision, Recall e F1 Score per ogni classe sono relativamente consistenti, con solo lievi variazioni tra le diverse categorie. Ciò suggerisce che le prestazioni dei modelli sono stabili e affidabili per le diverse classi. L'analisi dei classification report suggerisce che la classe "Clock" è generalmente la più facilmente predetta dal modello, mentre la classe "Airplane" rappresenta una sfida maggiore, come indicato dai valori di Recall e soprattutto di Precision più bassi. Questo concetto è meglio approfondito e chiarito vedendo le confusion matrix. Le confusion matrix rafforzano questa conclusione, poiché la colonna relativa alla classe "Airplane" mostra valori mediamente più elevati. Ciò indica che il modello ha spesso predetto erroneamente la classe "Airplane" quando la True Label era diversa. In altre parole, la classe "Airplane" ha un numero più elevato della media di falsi positivi. Inoltre, la confusion matrix evidenzia alcune ambiguità dirette del modello, ad esempio tra "Fork" e "Pencil", il che è comprensibile dato che i disegni possono avere somiglianze.

6 GUI



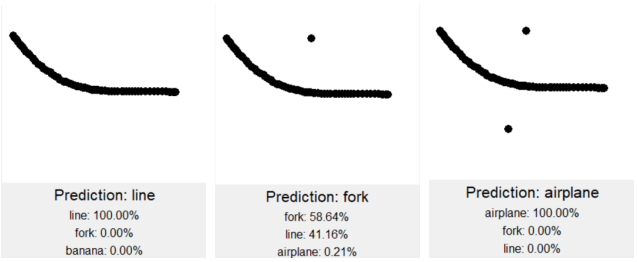
Questo programma crea un'interfaccia grafica usando la libreria Tkinter , consentendo agli utenti di disegnare su un canvas interattivo e visualizzare le predizioni in tempo reale del modello. La CNN è stata selezionata come modello per effettuare previsioni sul canvas in quanto ha dimostrato le migliori prestazioni. Ogni volta che l'utente smette di disegnare (rilascia il mouse), il programma esegue una predizione real-time del disegno effettuato, che viene quindi visualizzata sotto l'area di disegno. Inoltre, l'interfaccia presenta le 10 migliori predizioni del modello in una tabella, fornendo un'indicazione del grado di fiducia del modello nella sua previsione. Sono anche stati integrati 2 pulsanti a disposizione dell'utente: il pulsante "Clear" per cancellare tutto il disegno e il pulsante "Eraser On/Off" per attivare/disattivare la gomma per apportare modifiche al disegno con facilità.

Esempi di immagini riconosciute correttamente



Sorprendentemente, quando il modello deve effettuare previsioni su immagini disegnate sul canvas, emergono delle differenze significative rispetto alle prestazioni ottenute sul test set. Le metriche di accuratezza

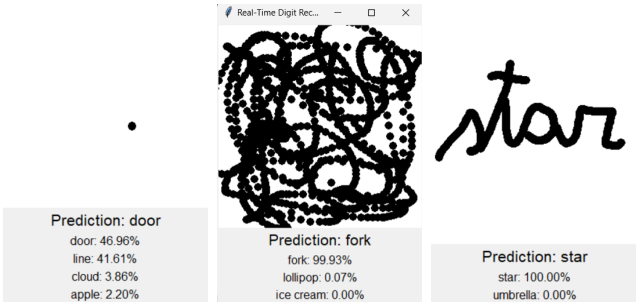
riportate precedentemente per ciascuna classe nel Classification Report sul test set non riflettono accuratamente le prestazioni del modello durante la previsione di immagini in tempo reale. È notevole osservare che la maggior parte delle classi viene predetta con successo, dimostrando l'efficacia del modello nell'ambito dell'utilizzo del canvas. In particolare, la classe "Airplane", che aveva mostrato risultati inferiori in termini di F1 Score nel Classification Report, sembra essere predetta con successo in molte istanze, suggerendo che il modello si comporta in modo soddisfacente per questa classe. Tuttavia, è vero anche il contrario per alcune classi, le cui prestazioni che erano buone nel contesto del test set iniziano a mostrare problematiche nell'ambito delle immagini disegnate in tempo reale.



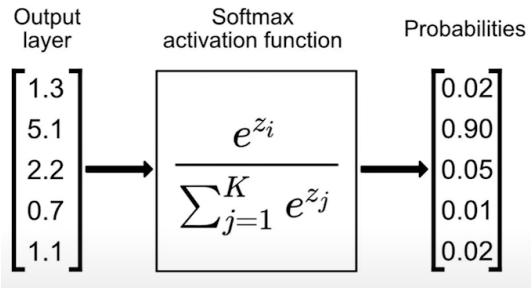
Inaspettatamente, la classe più difficile da riconoscere per il modello (caratterizzata da una bassa precision) è "Line". La sua semplicità sembra presentare sfide, poiché il modello fatica a prevederla con precisione. Anche quando viene indovinata, aggiungere anche solo pochi puntini al disegno può cambiare completamente le previsioni del modello, mentre per altre classi il modello mantiene una previsione più stabile.
Parallelamente, è emerso un aspetto di rilevanza più ampia noto come "shift and scale variance". Questo si verifica quando il modello non è in grado di riconoscere correttamente le immagini che non sono ben centrate o che sono troppo piccole. In sostanza, il classificatore non dovrebbe essere influenzato dalla posizione dell'oggetto nell'immagine.
Per affrontare questa problematica, è possibile ricorrere alla tecnica della data augmentation, che consiste nell'aumentare artificialmente le dimensioni del set di addestramento mediante l'applicazione di varie trasformazioni e modifiche alle immagini esistenti. Questo processo implica la modifica delle immagini attraverso rotazioni, scalature, ritagli, ribaltamenti e spostamenti, al fine di creare nuovi esempi di addestramento con variazioni rispetto ai dati originali. L'obiettivo principale della data augmentation è migliorare la capacità del modello di generalizzare e aumentare la sua robustezza rispetto a diversi tipi di input.

È stato quindi esaminato l'effetto dell'introduzione della data augmentation nel modello. In generale, questa tecnica ha consentito al modello di riconoscere immagini sul canvas spostate o rimpicciolite con una maggiore precisione. Tuttavia, si è notato un calo significativo nella frequenza di corretti riconoscimenti delle immagini rispetto alla configurazione precedente. Di conseguenza, l'idea di utilizzare la data augmentation è stata abbandonata, privilegiando invece un riconoscimento più affidabile delle immagini create direttamente sul canvas, rinunciando ai potenziali vantaggi legati alla capacità di generalizzazione del modello.

Un'ultima curiosità di questo modello è che, quando si disegna un immagine che non corrisponde a nessuna classe in particolare, ci si potrebbe aspettare che il modello mostri incertezza. Invece, il modello fornisce una previsione con discreta sicurezza.



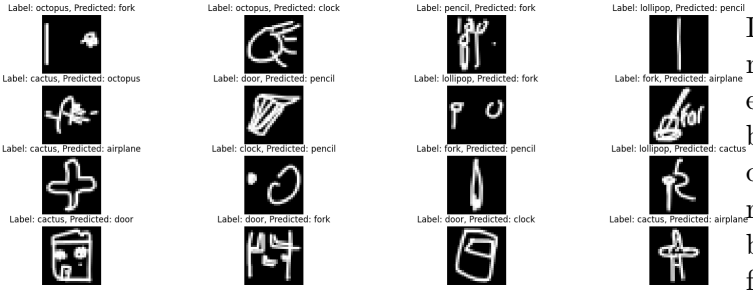
Questo è un classico fenomeno noto nel Machine Learning come "Overconfidence", ovvero quando il modello dimostra un eccessivo grado di fiducia nelle previsioni, anche se sono immagini molto diverse da quelle presenti nel training set. L'attivazione softmax è la causa principale dell'overconfidence del modello perché enfatizza e amplifica le differenze nei punteggi delle classi, portando i valori più bassi a diventare ancora più bassi e i valori più alti a diventare ancora più alti.



Uno studio afferma che per affrontare questa problematica si può usare un approccio noto come Last Layer Laplace Approximation (LLLA). Questo approccio coinvolge l'aggiunta di una distribuzione di probabilità sui pesi dell'ultimo strato del modello, il che permette di quantificare l'incertezza del modello. La varianza di questa distribuzione influisce sulla fiducia delle previsioni del modello, con valori più elevati che riducono la fiducia nelle previsioni "fuori distribuzione". Una volta che la probabilità si è abbassata, si potrebbe considerare l'aggiunta di una soglia di certezza, in modo che il modello fornisca previsioni solo quando la sua certezza supera una determinata soglia. Questo potrebbe aiutare a identificare situazioni in cui il modello è davvero sicuro della sua previsione e ignorare i casi in cui la certezza è troppo bassa.

7 Sviluppi Futuri

Un ampio spazio di miglioramento si presenta nei futuri sviluppi del progetto. Diverse idee e strategie potrebbero essere esplorate al fine di rendere il sistema ancora più efficace e versatile nel riconoscimento di immagini disegnate sul canvas. La più importante area di miglioramento riguarda il "data cleaning" del dataset. Vediamo degli esempi di immagini del test set classificate erroneamente dal modello:



Sono oggettivamente di pessima qualità e difficilmente riconoscibili. Effettuare un'attenta revisione del dataset e rimuovere le immagini di bassa qualità o ambigue potrebbe contribuire a migliorare la precisione complessiva del modello, eliminando dati di disturbo che potrebbero confondere l'addestramento. Questo processo richiederebbe una valutazione manuale o l'implementazione di algoritmi di filtro per identificare e rimuovere le immagini non informative o fuori standard. Un altro aspetto chiave potrebbe riguardare l'approfondimento delle reti neurali. L'aggiunta di strati o la progettazione di architetture più complesse potrebbe consentire al modello di acquisire una migliore capacità di generalizzazione e di effettuare previsioni più precise sul canvas. Inoltre, combinando una rete più complessa con la data augmentation, si potrebbe trovare un equilibrio tra l'ottimizzazione delle previsioni sul canvas e il mantenimento dell'accuratezza generale del modello. C'è ampio spazio per il miglioramento attraverso l'espansione del dataset e l'aumento del numero di classi. Sebbene il dataset originale Quick, Draw! contenesse oltre 50 milioni di immagini disegnate dagli utenti, questo progetto ha utilizzato solo una piccolissima parte, concentrandosi su 20 classi con 8000 esempi ciascuna. Espandere il dataset potrebbe aumentare la diversità delle immagini, consentendo al modello di apprendere da un insieme di dati più rappresentativo, mentre aumentare il numero delle classi potrebbe arricchire notevolmente le capacità del modello e rendere il gioco più vario e divertente. Un'ultima idea potrebbe consistere nel condurre test con utenti reali. Questo permetterebbe di raccogliere feedback diretti e osservazioni sul comportamento del sistema nell'uso quotidiano, identificando potenziali aree di miglioramento. Queste idee rappresentano direzioni interessanti per lo sviluppo futuro del progetto e potrebbero contribuire a ottimizzare il sistema, rendendolo più efficace e versatile.

Il progetto potrebbe essere evoluto in un'applicazione mobile o per desktop dedicata all'intrattenimento educativo e all'apprendimento interattivo per i bambini. Questa app permetterebbe ai bambini di disegnare oggetti o scene e ricevere immediatamente riscontri sulle loro creazioni. Ad esempio, un bambino potrebbe disegnare una macchina e il modello fornirebbe il feedback se il disegno corrisponde o meno a una macchina. Questo avrebbe un duplice vantaggio:

stimolerebbe la creatività dei bambini e li aiuterebbe a imparare a riconoscere oggetti comuni in modo divertente.

8 Conclusioni

Questo progetto ha esplorato le potenzialità dell'apprendimento automatico nel riconoscimento di disegni realizzati dagli utenti su un canvas interattivo. Nonostante alcune sfide e opportunità di miglioramento, il risultato è stato sorprendentemente positivo, dimostrando come il modello possa compiere previsioni precise in un contesto di utilizzo pratico. Le applicazioni future possono spaziare dall'educazione all'intrattenimento, aprendo nuove prospettive per coinvolgere le persone in attività creative e di apprendimento.

Bibliografia

(tutti i link sono cliccabili)

1. Corso di Machine Learning e Sistemi Intelligenti A.A. 2022/2023, Università degli studi Roma TRE
2. Canale YouTube "deeplizard"
3. Quick, Draw! - Dataset
4. Articoli sul Machine Learning da Humanativa Spa
5. Articolo su "Overconfidence" e LLLA
6. Video su YouTube: "Neural Network From Scratch" di Samson Zhang
7. Video su YouTube: "How to Create a Neural Network (and Train it to Identify Doodles)" di Sebastian Lague