Contents

Riconoscimento Alfabeto Americano dei Segni ASL Recognition

Stefano Morelli 576975
Lorenzo Sacco 580515

***Abstract*—Questo progetto mira a riconoscere in tempo reale le lettere dell'alfabeto dei segni americano utilizzando alberi di decisione, reti neurali, CNN e modelli preaddestrati come quelli forniti da MediaPipe.**

INTRODUZIONE Come primo approccio alla materia abbiamo deciso di approfondire meglio idee trovate in rete e, dopo vari tentativi, ci siamo imbattuti in un dataset su Kaggle che ha attirato la nostra attenzione per la possibilità di sfruttare il lavoro svolto effettuando delle previsioni in tempo reale; abbiamo quindi per prima cosa raggiunto un ottimo livello di accuracy sui dati forniti per poi concentrarci sulla parte live, dovendo però rivedere l'approccio iniziale.

1. Descrizione dataset

Il dataset utilizzato è stato reperito da Kaggle, consta di circa 50000 immagini, con risoluzione 200x200, che raffigurano sempre la stessa mano in posizioni diverse in base alla lettera rappresentata. La suddivisione tra test e training set è già effettuata ma il test set è di esigue dimensioni, meno di 30 immagini, proprio per questo il progetto valuta l'accuratezza del modello in tempo reale e non utilizzando i dati di test forniti nel dataset. La presenza di una sola mano sempre contornata dallo stesso sfondo poteva causare problemi ai fini dell'apprendimento, per questo motivo abbiamo pensato di sfruttare un secondo dataset con mani differenti, risoluzione più alta(400x400) e sfondi sempre diversi. I due dataset sono organizzati in una serie di sottocartelle, ognuna rappresentante una label (una lettera dell'alfabeto o un carattere speciale) [1].

2. Modelli

Per il progetto abbiamo seguito diverse strade, adottando più modelli e confrontandone i risultati abbiamo scelto l'approccio migliore per poi predire le lettere live.

2.1. Alberi decisionali

Il primo modello testato è stato quello degli alberi decisionali, fornendo in input le immagini, precedentemente convertite in vettori. Il modello scelto era volutamente semplice per osservare come si sarebbe comportato e farsi un'idea della difficoltà del problema.

2.2. Reti Neurali

Questo modello prende in input le immagini del dataset, precedentemente standardizzate, ed è una rete neurale costituita da 4 strati:

- Primo strato: è layer di tipo flatten serve

[1] nel secondo dataset i caratteri speciali "space" e "del" non sono rappresentati

ad appiattire l'input (bidimensionale) rendendolo monodimensionale.
- Due strati nascosti: sono entrambi densi[2] e con 256 neuroni con funzione ReLu per strato.
- Ultimo strato: è un layer denso con funzione d'attivazione softmax a 29 neuroni (uno per ciascuna classe).

2.3. NN con PCA

L'addestramento del modello NN era piuttosto lento per questo abbiamo pensato di ridurre le dimensioni dell'input attraverso la PCA invece di farlo appiattire dalla rete stessa per guadagnare in efficienza. Abbiamo utilizzato PCA, un algoritmo di riduzione dimensionalità che cerca le componenti a massima varianza, per perdere il minor numero di informazioni possibile.

2.4. Reti Convoluzionali

Visto che i risultati erano incoraggianti ma ancora non ottimi, abbiamo sfruttato le potenzialità delle CNN data la loro capacità intrinseca di apprendere dalle immagini. La rete utilizzata e realizzata grazie all'accoppiata di due gruppi di layer, ognuno realizzato con 4 strati:

- Primo strato: conv2d con 16 filtri dimensione 2x2 e funzione d'attivazione ReLu.
- Secondo strato: maxPooling che prende il valore massimo in finestre 3x3, riducendo quindi la dimensione dell'immagine e, di conseguenza, il numero di parametri della rete.
- Terzo strato: batch normalization che aiuta a rendere più rapido il processo di training.
- Quarto strato: Dropout che serve per ridurre l'overfitting, questo e lo strato precedente sono stati aggiunti ispirandoci alle reti trovate nella sezione code di Kaggle.

oltre a questi due gruppi abbiamo aggiunto altri tre strati:

1) flatten per ridurre dimensionalità
2) dense composto da 128 neuroni con funzione d'attivazione ReLu
3) decisionale con 29 neuroni e funzione d'attivazione softmax

sia per questo approccio che per la NN con PCA2.3 abbiamo effettuato il tuning degli iperparametri attraverso un ottimizzatore Bayesiano

2.5. NN con landmarks

Una rete neurale che non riceve in input le immagini ma una serie di valori che rappresentano la posizione di alcuni punti chiave della mano all'interno della foto (landmarks). Il calcolo di questi punti avviene grazie all'utilizzo di una rete neurale preaddestrata fornita all'interno della libreria mediaPipe. il modello a cui sono dati in input questi dati è molto simile a quello descritto in 2.2 ma lo strato di input è a dimensione 42 poiché i landmarks sono 21 (con due coordinate ciascuno).

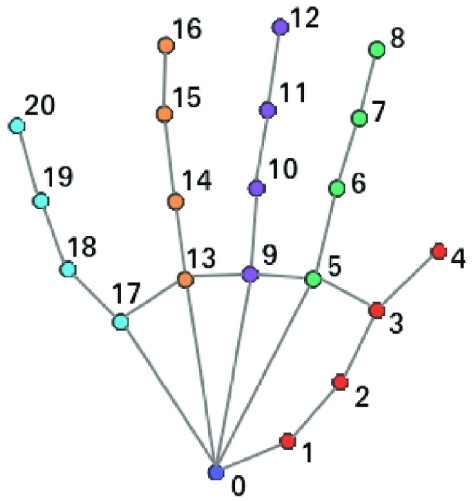

Figure 1: Esempio di Landmarks su una mano

3. Tecnologie

Per portare avanti il progetto abbiamo sfruttato diverse librerie sviluppate per Python, linguaggio scelto per la sua duttilità e semplicità.

3.1. OpenCv

[2] OpenCv è una libreria per la lettura e la manipolazione delle immagini. Ci è tornata utile per caricare le immagini in una lista su cui poi poter fare le necessarie operazioni, tra

[2]nei layer densi tutti i neuroni sono connesi ai neuroni dello strato precedente

cui la conversione delle immagini in bianco e nero utilizzando proprio questa stessa libreria.

3.2. TensorFlow

[1] Libreria molto utilizzata per la creazione di reti neurali tramite l'API **Keras** permette comodamente di specificare layer per layer la struttura della rete che si vuole realizzare. Offre inoltre uno strumento per l'analisi della rete che si chiama **tensorBoard** che abbiamo utilizzato per visualizzare graficamente il comportamento del modello.

3.3. scikit-learn

[4] Libreria pensata per il machine learning che offre delle implementazioni comode ma molto potenti di modelli come decision Tree ma anche di algoritmi quali PCA. Nel nostro progetto l'abbiamo sfruttata nella realizzazione dei modelli NN con PCA2.3 e Decision Tree2.1.

3.4. MediaPipe

[3] Utile per lavorare nell'ambito della computer vision [3]; fornisce, tra le altre cose, un modello preaddestrato per l'individuazione dei landmarks su mani,viso etc che ci è servito nella parte finale del progetto.

4. Analisi dei risultati

Ecco l'analisi dei risultati dei modelli presentati in 2.

4.1. Reti Neurali

Nei grafici che seguono di mostra l'andamento dell'accuracy e della LOSS nel corso dell'apprendimento. Notiamo che ci sono grandi sbalzi soprattutto per quanto riguarda il training set con una curva che scende e sale notevolmente nel giro di poche epoche. Nonostante tutto si raggiungono buoni risultati con accuracy che supera l'80%. Le coppie di lettere più facilmente confondibili risultano essere: F ed E, V e W, D e C.

[3]La computer vision è un campo dell'informatica che si occupa di permettere ai computer di interpretare e comprendere il contenuto visivo del mondo, come immagini e video. Utilizza tecniche di intelligenza artificiale e algoritmi per estrarre, analizzare e interpretare informazioni visive, emulando la capacità della vista umana.

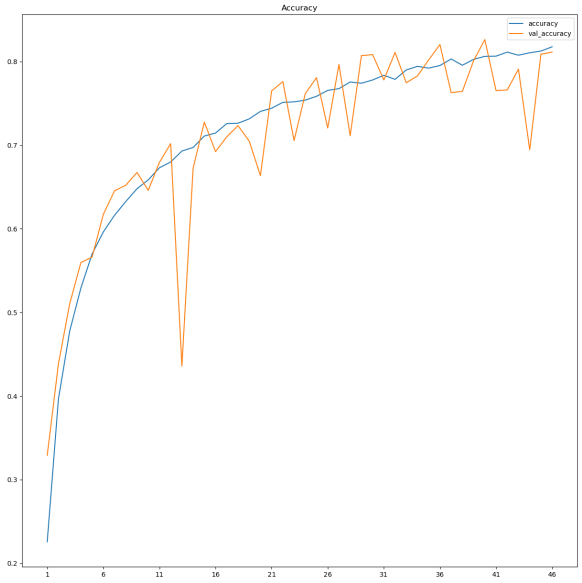

Figure 2: (a: Accuracy del modello)

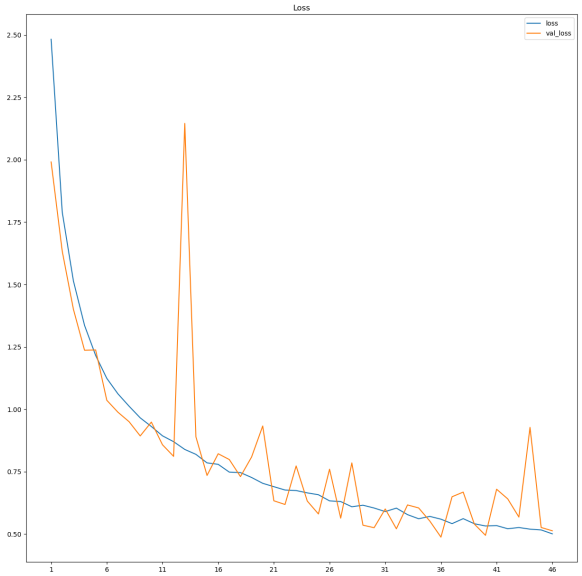

Figure 3: (b: Loss del modello)

in Blu training set , in Rosso validation set

Lo notiamo dalla figura:4, questo ha senso osservando la rappresentazione effettivamente simile delle suddette lettere in ASL.

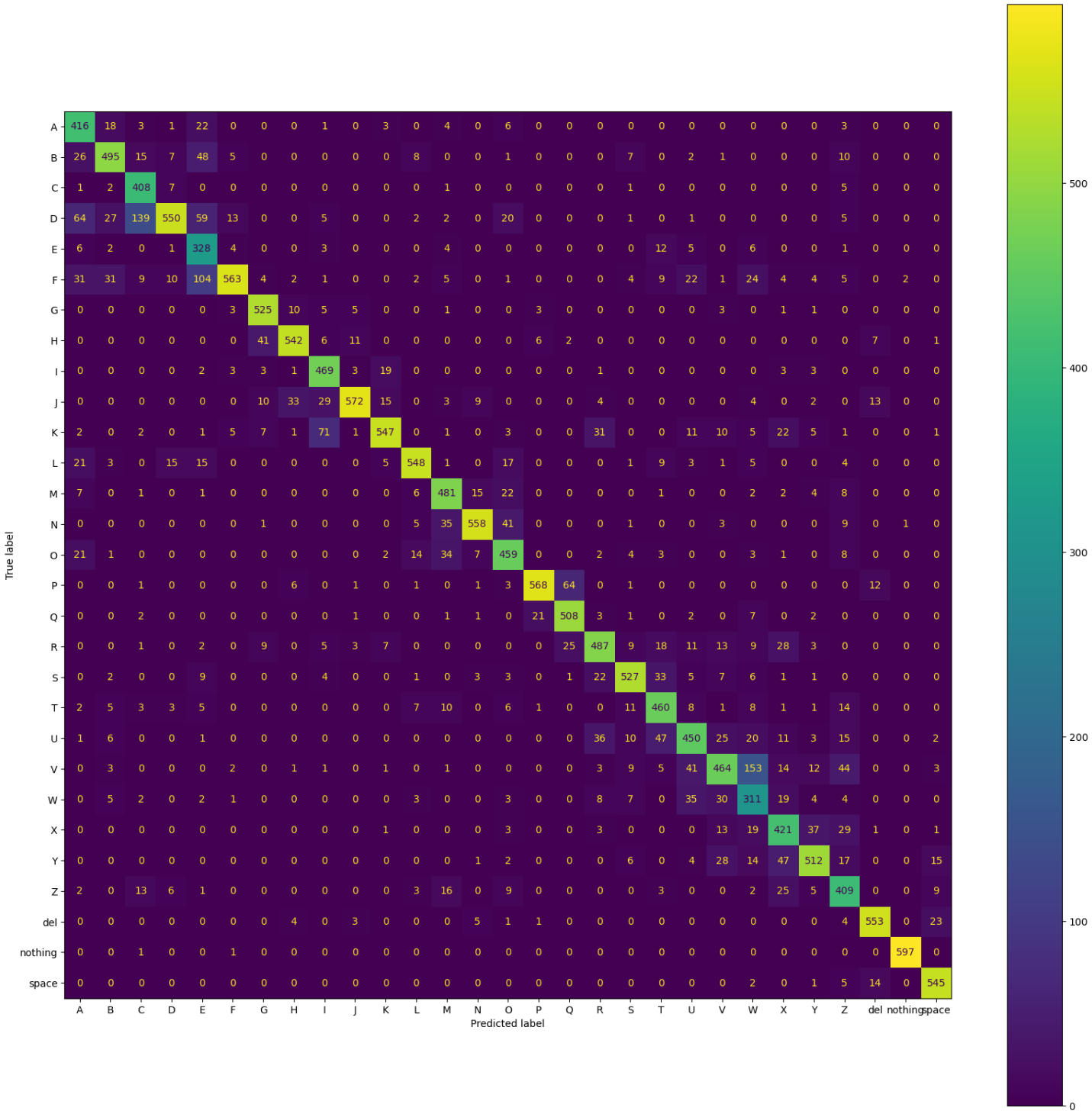

Figure 4: Confusion matrix del modello addestrato

4.2. NN con PCA

Utilizzando la PCA abbiamo ridotto notevolmente le tempistiche di addestramento guadagnando anche in accuratezza. Come possiamo facilmente osservare dai grafici 5, infatti, sia il training set che il validation rispondono bene all'addestramento aumentando costantemente la propria accuracy fino ad arrivare ad ottimi risultati, superiori al 96%. Possiamo anche notare che non c'è una sostanziale differenza tra accuracy sul training o sul test set. Per quanto riguarda la LOSS valgono i medesimi ragionamenti, Escluse quindi possibilità di overfitting.

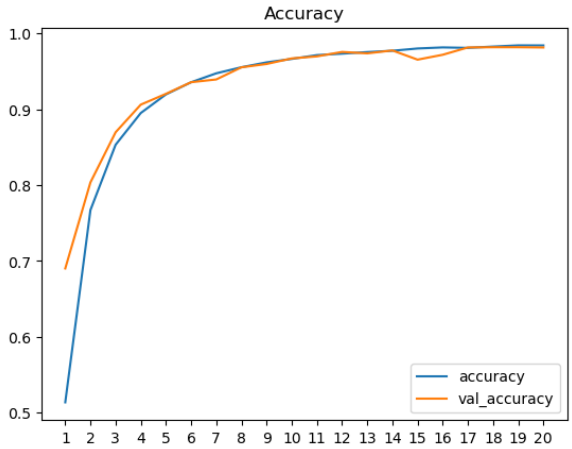

Figure 5: (a: Accuracy del modello)

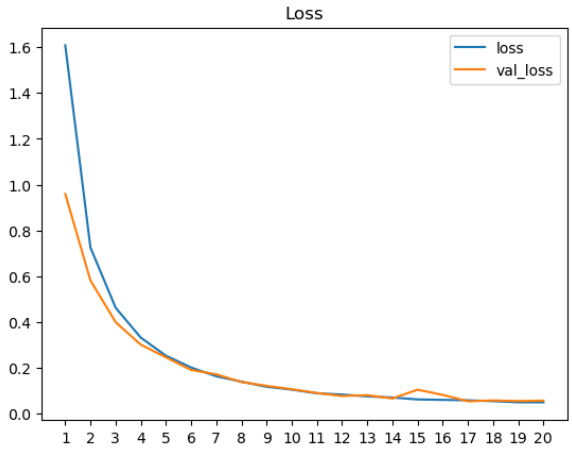

Figure 6: (b: Loss del modello)

4.3. Reti Convoluzionali

Per le reti convoluzionali l'addestramento è stato indubbiamente più lento, ma non aveva senso utilizzare la PCA per abbattere i tempi, a causa della natura delle CNN abilissime a lavorare con le immagini in input. I risultati sono incoraggianti infatti l'apprendimento seppur non molto lineare, permette al modello di raggiungere un accuracy vicina al 90%. Fino alle ultime epoche l'accuracy calcolata sul validation è superiore a quella calcolata sul training quindi non vi è overfitting.

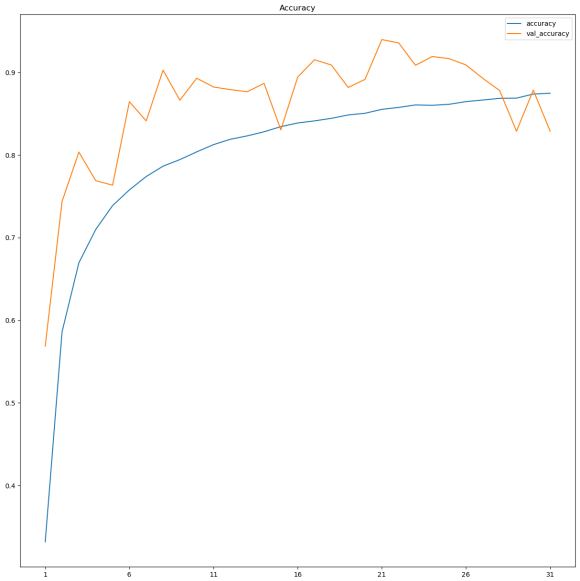

Figure 7: (a: Accuracy del modello)

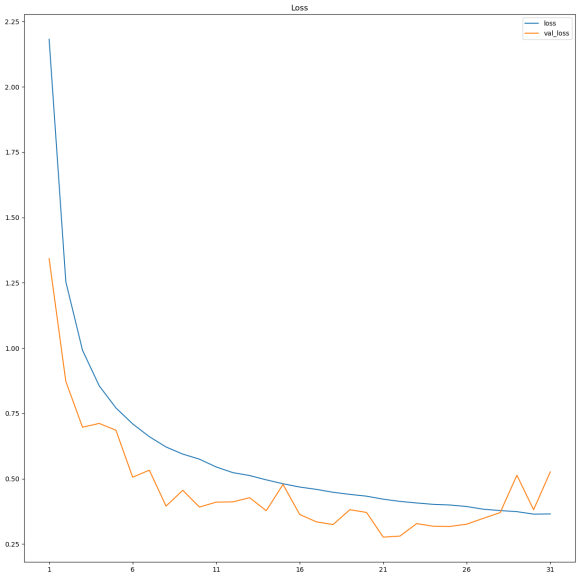

Figure 8: (b: Loss del modello)

in Blu training set , in Rosso validation set

Poiché le CNN fornivano risultati interes-

santi abbiamo deciso di effettuare il tuning degli iperparametri attraverso l'ottimizzatore bayesiano per valutare l'andamento del modello a seguito di tale operazione. Il miglioramento è netto con un accuracy che sfiora il 100% seppur nelle ultime epoche si va verso l'overfitting. Da segnalare anche una interessante ricaduta verso la metà dell'apprendimento dopo la quale però il modello torna a migliorare.

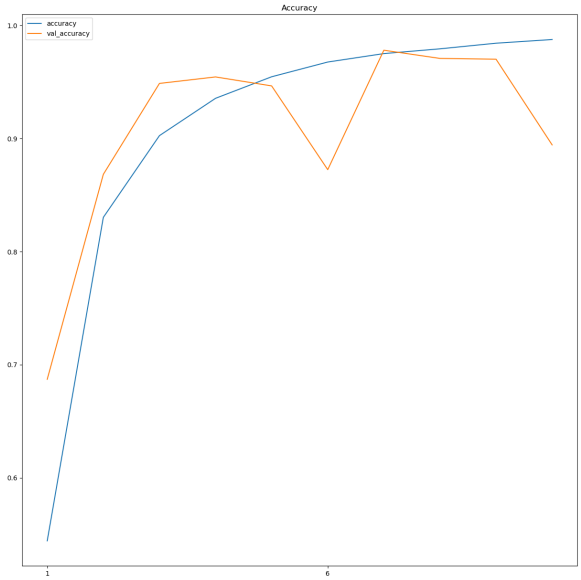

Figure 9: (a: Accuracy del modello)

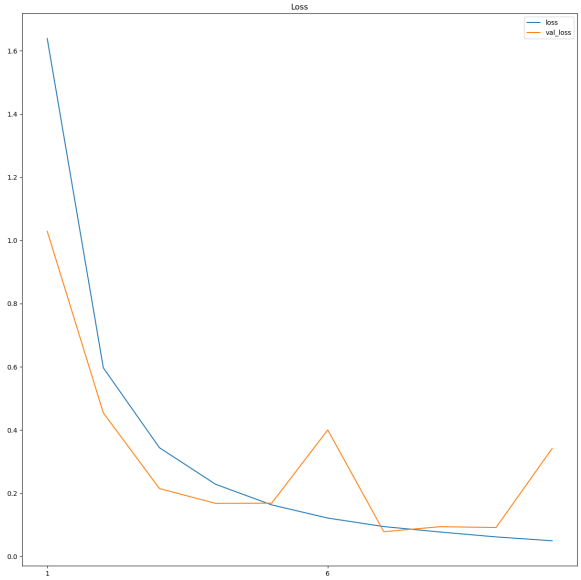

Figure 10: (b: Loss del modello)

in Blu training set , in Rosso validation set

5. Live classification

A questo punto abbiamo pensato di utilizzare i risultati ottenuti per effettuare delle prediction in tempo reale. Abbiamo subito notato che, dando in pasto al modello delle istantanee catturate mentre eseguivamo i vari gesti, i risultati non erano affatto quelli sperati. A fronte di un accuracy superiore al 90 sul validation set (o comunque intorno a questa cifra), il modello, in live produceva risutati corretti in pochissime occasioni, dando l'impressione di rispondere quasi casualmente. Abbiamo subito immaginato che il problema potesse derivare dalla struttura del dataset, in cui ogni foto era contornata sempre dallo stesso sfondo e la mano era sempre della stessa persona. Aggiungendo un secondo dataset con mani differenti e sfondi nuovi, però, i risultati non sono migliorati e, per questo, la scelta è ricaduta su un approccio completamente diverso. Siamo partiti allenando una nuova rete neurale (descritta in 2.5) che non prendesse in input le immagini ma i punti chiave della mano. Così facendo, abbiamo completamente superato il problema degli sfondi e della mano. Il modello allenato ha prodotto ottimi risultati sia in fase di addestramento, anche qui superando il 90 di accuracy, sia, questa volta, in live riuscendo a individuare con una certa accuratezza la lettera.

6. Sviluppi futuri

Per questo progetto sono numerose le applicazioni immaginabili: dalla generazione di sottotitoli a partire dal linguaggio dei segni, fino a una possibile tastiera sostitutiva. In futuro si potrebbe comunque migliorare il risultato ottenuto, rendendo la predizione delle lettere ancora più accurata. Un idea potrebbe anche essere quella di integrare il progetto con un'intelligenza artificiale, che comprenda le frasi scritte con ASL e risponda con dei gesti. Questo potrebbe incentivare molte persone con disabilità uditive, e che non hanno grande confidenza con la tecnologia, a comunicare con degli assistenti visuali, sostitutivi a quelli vocali già molto in voga quali alexa o google home.

References

[1] Martín Abadi et al. “TensorFlow: A system for large-scale machine learning”. In: 2016.

[2] G Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[3] Google. *Mediapipe*. 2023.

[4] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of Machine Learning Research* 12 (2011). ISSN: 15324435.