Bortolussi Lorenzo SM3201370 5/5/2025

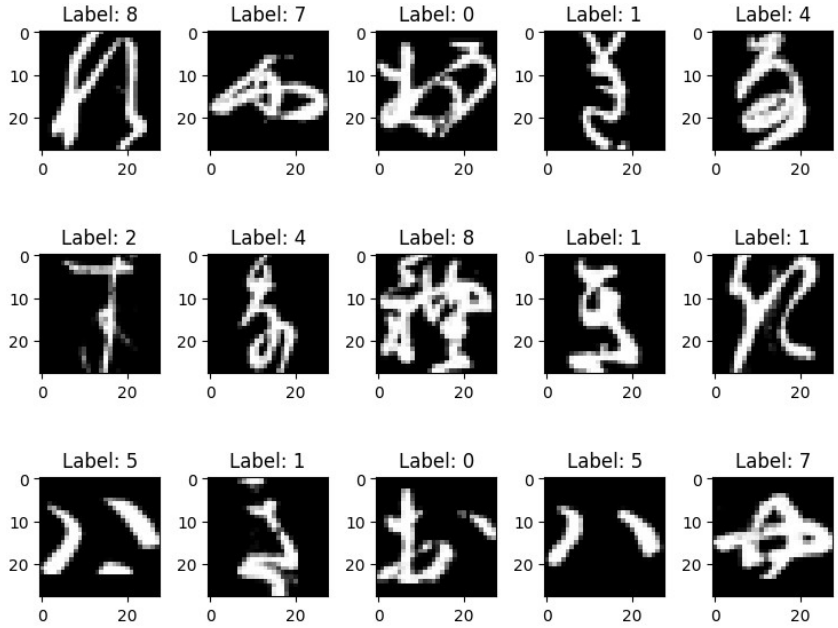
CHALLENGE 3 – REPORT

In questa challenge si ha esplorato le capacita delle reti neurali nel riconoscimento di immagini di caratteri appartenenti al dataset KMNIST. Nel mettere a confronto reti neurali convoluzionali (CNN) e completamente connesse lineari (FCNN), si ha considerato più modelli e diversi ottimizzatori, con l'intento di verificare come questi cambiamenti modificassero le prestazioni dei modelli considerati. Si ha poi approfondito un singolo modello, analizzando varie grandezze di interesse come l'evoluzione della loss durante l'allenamento.

Per l'implementazione dei modelli si ha utilizzato la libreria Torch.

ESPLORAZIONE DEI DATI

Prima di cominciare, osserviamo i dati con cui si dovrà lavorare:



Si tratta di immagini di dimensioni 28x28, con un singolo canale di colore, mentre le etichette risultano dividersi in 10 classi distinte. Si decide di utilizzare 32 elementi per batch durante l'allenamento.

DESCRIZIONE DEI MODELLI

I modelli presi in considerazione sono i seguenti:

- **FCNN_v1** : rete neurale lineare fully connected con un hiden layer
- **FCNN_v2** : rete neurale lineare fully connected con tre hidden layers
- **CNN_v1** : rete neurale con due layer convoluzionali
- **CNN_v2** : rete neurale con due layer convoluzionali seguiti da pooling
- **CNN_v3** : rete neurale con due layer convoluzionali preceduti da pooling

Il modello CNN_v3 si distingue dal CNN_v2 in quanto l'operazione di pooling viene effettuata sui dati in input, riducendo da subito la dimensione delle immagini nel tentativo di introdurre immediatamente invarianza rispetto a piccole traslazioni dei caratteri. Senza il vincolo di tre hidden layers massimi, sarebbe interessante osservare anche le peformance di altre combinazioni dei due tipi di layer (pooling e convoluzione).

Vista la bassa riusoluzione dell'immagine, i kernel dei layer convoluzionali e le dimensioni dei pool vengono mantenute ridotte, per evitare di perdere troppi dettagli dell'immagine.

OSSERVAZIONI SUI RISULTATI

Per quanto riguarda l'ottimizzatore, per ogni modello si utilizza sia Adam che SGD, utilizzando due learning rate con ordini di grandezza diversi. Il training è stato eseguito su 4 epoche.

Nella tabella seguente sono riportati i risultati dei modelli sopra descritti:

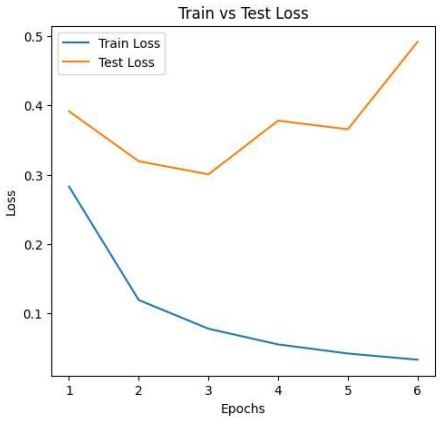
Model	Optimizer	Learning Rate	Train Accuracy	Test Accuracy
FCNN_v1	Adam	0.002	97.808831	89.906150
FCNN_v1	Adam	0.020	88.737327	74.520767
FCNN_v1	SGD	0.002	78.545224	65.095847
FCNN_v1	SGD	0.020	90.800093	81.110224
FCNN_v2	Adam	0.002	97.228522	88.638179
FCNN_v2	Adam	0.020	86.856323	71.705272
FCNN_v2	SGD	0.002	65.935165	54.173323
FCNN_v2	SGD	0.020	93.241395	84.454872
CNN_v1	Adam	0.002	98.772679	91.253994
CNN_v1	Adam	0.020	88.118663	75.489217
CNN_v1	SGD	0.002	84.446705	72.923323
CNN_v1	SGD	0.020	96.116262	89.147364
CNN_v2	Adam	0.002	98.512540	92.332268
CNN_v2	Adam	0.020	90.423226	81.170128
CNN_v2	SGD	0.002	82.305563	70.437300
CNN_v2	SGD	0.020	95.832777	89.057508
CNN_v3	Adam	0.002	98.614261	92.501997
CNN_v3	Adam	0.020	88.568903	78.863818
CNN_v3	SGD	0.002	83.049293	70.666933
CNN_v3	SGD	0.020	95.986193	88.987620

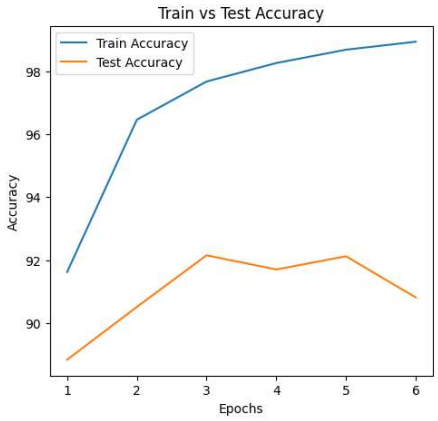
- Dai dati si nota come il learning rate più basso ha favorito l'ottimizzatore Adam, mentre quello più alto ha reso accuracy maggiore tramite SGD. Questo potrebbe essere legato al basso numero di epoche considerate;
- Sorprendentemente quasi ogni combinazione ha reso accuracy piuttosto elevate, anche quelli ritenuti più semplici;
- Il calo di accuracy tra train e test set sembra oscillare attorno al 10% per tutte le combinazioni considerate;
- Non sembrano esserci differenze significative tra i modelli CNN_v2 e CNN_v3, al contrario di quanto si pensava in precedenza.

ANALISI DI LOSS E ACCURACY

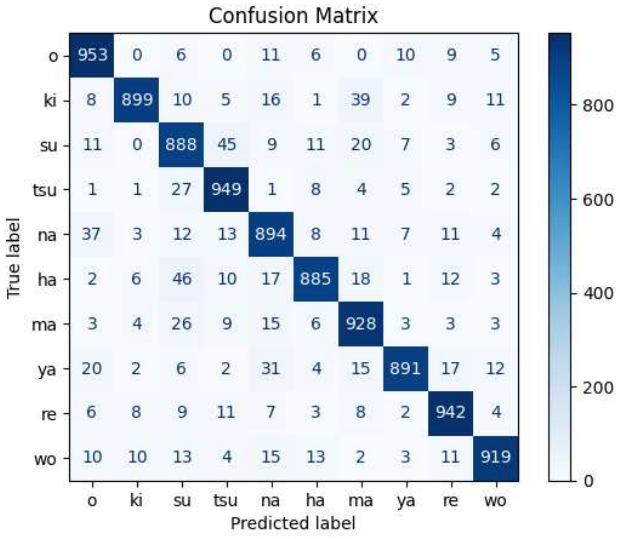
Nel seguito si considera il modello CNN_v2, ritenuto esemplificativo anche per gli altri modelli convoluzionali osservati in precedenza. In questa sezione il numero di epoche considerate è stato alzato a 6.

Come prima cosa si osserva l'evoluzione di loss e accuracy durante l'allenamento del modello:





Si nota come, superata la terza epoca, mentre la train loss continua a calare, la test loss supera un minimo e comincia a crescere. Un comportamento simile si nota nell'andamento delle accuracy, dove nel test si appiattisce per qualche epoca per poi crescere. Si suppone che questo comportamento segua dall'overfitting che il modello ha verso i dati di train, quindi è ragionevole pensare di fermare l'allenamento del modello alla terza o quarta epoca.



Dalla confusion matrix si nota come la classificazione sembra essere piuttosto buona, visti gli alti numeri sulla diagonale. Si osserva che alcuni gruppi di caratteri vengono classificati erroneamente più di altri, per esempio [su , tsu]: questo suggerisce che alcune classi condividono caratteristiche comuni e difficili da distinguere.

Vedendo come le cassi sono equilibrate, si può dire che l'accuracy è una buona misura di partenza per valutare il modello, misura che però non distingue le performance sulle singole classi. Osservando accuracy e F1-score per le singole classi si notano piccole oscillazioni ma tutti i valori rimangono comunque piuttosto elevati: