Introduction to Machine Learning - Challenge 3

Autore: Bredariol Francesco

Data di consegna: May 5, 2025

1 Introduzione

Lo scopo di questa challenge è quello di eseguire classificazione di numeri giapponesi. I numeri sono immagini con rispettivi label derivanti dal dataset KMNIST. La classificazione di tali numeri viene eseguita tramite uso di reti neurali, sia di tipo FCNN che di tipo CNN. L'implementazione di tali reti neurali è fatta con l'uso di PyTorch.

2 Dataset

Il dataset è composto di immagini di numeri scritti a mano (con relativi label). Ogni immagine ha dimensione 28×28 ed ha un solo canale di colore. Il dataset è bilanciato nelle classi (ogni classe ha circa lo stesso numero di rappresentanti). Questa è la testa del dataset mescolato.

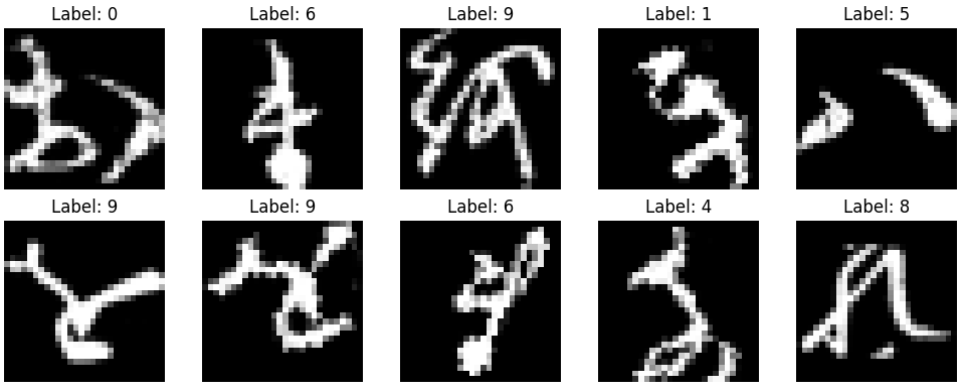

Figure 1: Sample di immagini con relativi label del dataset KMNIST.

3 Data Augmentation

Per ottenere una varietà maggiore di immagini ho preso una copia del dataset e l'ho modificata traimte trasformazioni affini di rotazione (tra -30° e +30°) e di traslazione (di un margine del 10% in entrambe le direzioni). Infine ho unito la copia modificata all'originale per generare un unico dataset più generale.

4 Modelli

Ho sperimentato modelli sia di tipo sia FCNN che di tipo CNN. Mantenendomi sempre nel range 2-4 layer (senza considera l'ultimo layer di softmax), ho per lo più usato come

funzioni di attivazione la ReLU e come funzione di Pooling il MaxPooling. Dopo aver variato più volte l'architettura in termini di numero di neuroni e di layer i due modelli che ho deciso di utilizzare sono (il primo della classe delle FCNN ed il secondo della classe delle CNN):

Layer	Type	Shape
1	Dense (ReLU)	(28x28, 256)
2	Dense (ReLU)	(256, 128)
3	Dense (ReLU)	(128, 64)
4	Dense (SoftMax)	(64, 10)

Layer	Type	Pooling	Kernel and Padding	Shape
1	Conve2D (ReLU)	MaxPool(2)	k = 3, p = 1	(1, 16)
2	Conve2D (ReLU)	MaxPool(2)	k = 3, p = 1	(16, 32)
3	Conve2D (ReLU)	MaxPool(2)	k = 3, p = 1	(32, 64)
4	Dense (ReLU)	FLATTENED	FLATTENED	(64x3x3, 128)
5	Dense (SoftMax)	-	-	(128, 10)

I risultati ottenuti, su un training di 10 epoche con batch size di 64, sono i seguenti.

Model	Test Accuracy	Test Loss
FCNN	89.38	0.4682
CNN	95.98	0.2210

5 Dropout

Una volta sperimentati i modelli semplici ho deciso di sperimentare i modelli con l'ausilio di dropout. Ho trovato interessante i risultati ottenuti, in quanto mettono in evidenza un legame tra validation set e train set. Ho inserito dropout dopo ogni attivazione (o dopo ogni pooler nel caso della CNN). Per la FCNN ho fissato un dropout di 0.5 fisso per ogni layer. Per la CNN ho usato il seguente scheduler per il dropout:

Layer	Dropout
1	0.5
2	0.3
3	0.2
4	0

I risultati ottenuti, su un training di 10 epoche con batch size di 64, sono i seguenti.

Model	Test Accuracy	Test Loss
FCNN	84.51	0.5071
CNN	96.25	0.1500

Si nota come, se la CNN ha avuto un miglioramento, la FCNN ha avuto un netto peggioramento.
Per quanto riguarda invece il legame tra validation e train set invece è molto interessante andare a valutare la loss over time del modello CNN con e senza dropout.

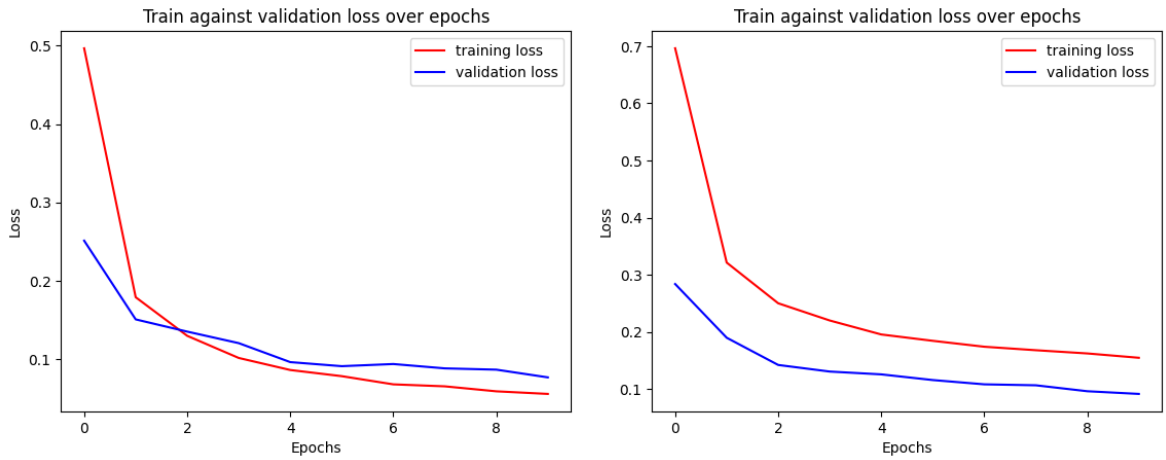

Figure 2: Nel riquadro di sinistra: Plot di "Train vs Validation loss over epochs" (10 epochs totali) per il modello CNN senza dropout. Nel riquadro di destra: Plot di "Train vs Validation loss over epochs" (10 epochs totali) per il modello CNN con dropout

La primissima cosa da notare è come la Validation Loss parta, generalmente, con un valore più basso della Train Loss. Questo è da giustificarsi con il fatto che, in realtà, il primo valore di validation loss è calcolato con un modello che ha già eseguito un ciclo di training, mentre il primo valore di training loss è calcolato con un modello completamente casuale.
Seconda cosa da notare è che quando c'è il dropout la validation loss tende ad essere sempre più bassa della train loss. Questo però non deve stupire: la train loss è calcolata con il modello in modalità train e dunque con il dropout attivo, motivo per cui giustamente esso performa peggio che in modalità eval.
Questo secondo punto mostra secondo me la forza reale del dropout: il dropout infatti riesce a migliorare il comportamento generale del modello proprio "facendolo sbagliare". Questo secondo me è molto interessante e può in qualche modo ricondursi al solito tradeoff tra Exploration ed Exploitation.