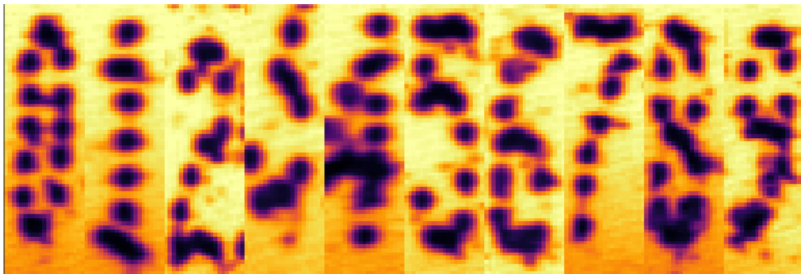
Digits Classification Challenge # Sponsored by SeaVision

Introduzione

I succhi di frutta di una nota ditta viaggiano su una linea di produzione con una velocità di 6 pezzi al secondo. Sui succhi di frutta vengono stampati dei numeri con una stampante a getto d'inchiostro che ha una risoluzione molto bassa. La Figura sotto, a sinistra, mostra un esempio di tali immagini.



Un sistema di visione automatico deve eseguire delle analisi sui numeri stampati. Un passaggio fondamentale del processo di automazione industriale è di classificare correttamente le singole cifre stampate sui succhi di frutta nel minor tempo possibile.

Obiettivo della Challenge

Vi viene chiesto di scrivere un algoritmo di classificazione che data un'immagine in bianco e nero di una singola cifra, sia in grado di identificare correttamente il numero rappresentato dall'immagine. La Figura sopra, a destra, mostra una sequenza di 10 immagini, con le cifre da 0 a 9.

DATI. I dati che avete a disposizione sono un insieme di 2000 immagini in bianco e nero con una risoluzione di $21 \times 71 = 1491$ pixel. Le immagini sono memorizzate in un file `.csv` con 2000 righe, in cui ogni riga contiene 1492 numeri: il primo numero è la *valore* dell'immagine, e poi seguono i valori di intensità dei 1491 pixels, espressi come numeri interi compresi tra 0 e 255.

Vi viene data la funzione `ReadSEAVISION(filename)` che prende in input il nome di un file del formato appena descritto e restituisce una tupla di 4 valori:

1. n: il numero di immagini.
2. m: il numero di pixel per immagine.
3. X: tutte le immagini lette, contenute in un'unica matrice con una colonna per ogni immagine. Ovvero, la matrice ha 1491 righe e 2000 colonne.
4. Y: le *label* corrette per ogni immagine, un vettore di 2000 elementi, con le cifre da 0 a 9.

Dovete usare le 2000 immagini contenute nel file `seavision_training.csv` per progettare e addestrare il vostro algoritmo di classificazione. Per velocizzare lo sviluppo dell'algoritmo di classificazione, vi viene fornito un micro data set con 10 immagini della cifra 5, e 10 immagini della cifra 6. Il file è chiamato `small_5_6.csv`.

VALUTAZIONE. Il vostro algoritmo di classificazione verrà valutato usando altre 1379 immagini a cui non avete accesso (ovvero non sono contenute nel file `seavision_training.csv`). Le immagini di test sono memorizzate utilizzando lo stesso formato e saranno rese pubbliche solo a gara terminata. I singoli algoritmi di classificazione verranno valutati considerando i due criteri seguenti in ordine lessicografico:

1. **Accuratezza**. Verrà calcolata la percentuale di classificazioni corrette sulle n_t immagini di test:

$$Acc(\hat{y}) = \frac{1}{n_t} \sum_{i=1}^{n_t} \text{equal}(\hat{y}_i, y_i) \times 100$$

   in cui il vettore $\hat{y}$ è il vettore delle predizioni del vostro modello, y è il vettore delle label corrette, e `equal` è una funzione che restituisce 1 quando i due valori di input sono uguali, e 0 altrimenti. L'accuratezza percentuale verrà arrotondata alla prima cifra decimale. Per esempio, 91.37% diventa 91.4% e 91.34% diventa 91.3%

2. **Dimensione del Modello**. A parità di accuratezza, si distingueranno le soluzioni usando un secondo criterio, ovvero il numero di parametri del vostro modello $\hat{y} = f(x; w)$, dato dal numero totale di coefficienti in w. Verranno favoriti i modelli che usano un numero inferiore di parametri.

Riferimenti Utili

Siete liberi di usare qualsiasi metodo di classificazione. Per chi intende provare con delle reti neurali usando Julia, si suggerisce di **studiare** la documentazione seguente:

- Si veda il file `XOR2.jl` in cui viene usata la libreria `Flux` per definire il modello di una rete multistrato, e per eseguire la parte di training. In generale si consiglia di leggere tutta la documentazione: `https://fluxml.ai/Flux.jl/stable/`
- Per definire eventuali **layer** degli strati nascosti, si consiglia di usare le funzioni `Chain` e `Dense`. Si veda: `https://fluxml.ai/Flux.jl/stable/models/basics/#Building-Layers-1`
- Si scelgano in modo opportuno le **funzioni di attivazione**: `https://fluxml.ai/Flux.jl/stable/models/layers/#Activation-Functions-1`
- Per velocizzare la parte di training, si considerino diversi **algoritmi di ottimizzazione**: `https://fluxml.ai/Flux.jl/stable/training/optimisers/#Optimiser-Reference-1`
- Anche se si consiglia di iniziare con un modello più semplice, chi vuole, può provare a progettare una **rete convolutiva**, cercando nella documentazione di Flux per *"Convolution and Pooling Layers"*.
- Avendo a disposizione poche immagini, si cerchi di elaborare una strategia per evitare un **overfitting** sulle immagini di training.

Facoltativo: Adversial Challenge

Verrà riconosciuto il premio della critica a chi sarà in grado di generare un'immagine artificiale, che sia in grado di far sbagliare gli algoritmi di classificazione degli altri studenti. L'immagine artificiale deve essere il più simile possibile ad una delle immagini di training. Bisogna fornire sia l'immagine di partenza, che quella modificata. La norma euclidea tra le due immagini (prese come vettori di $\mathbb{R}^{1491}$) deve essere inferiore a 255. La figura a destra mostra un'immagine del data set e un'immagine *fake*: la norma della differenza tra le due immagini è pari a 169.