Homework 3

Autore: Alessia Notaro
Corso: ingegneria informatica, II anno, canale M-Z

Descrizione del dataset

Il dataset utilizzato è una variante di CIFAR-10.
Si tratta di un insieme di immagini a bassa risoluzione, di dimensioni 32x32 pixel, ciascuna appartenente a una delle 10 classi predefinite: aereo, automobile, uccello, gatto, cervo, cane, rana, cavallo, nave e camion.

A differenza della versione standard, questo dataset è stato modificato appositamente per introdurre rumore nelle etichette del training set. Questa alterazione ha lo scopo di simulare scenari reali, in cui i dati di addestramento possono essere affetti da errori umani, ambiguità o annotazioni imperfette. L'impatto principale è un rischio aumentato di overfitting: il modello può imparare a replicare le etichette sbagliate, perdendo capacità di generalizzazione.

Il dataset è stato fornito in formato *.npy*, quindi è già strutturato e suddiviso in tre insiemi:

- ***x_train, y_train***: dati di addestramento, contenenti etichette rumorose;
- ***x_val, y_val***: dati di validazione, con etichette pulite;
- ***x_test, y_test***: dati di test, con etichette pulite.

Descrizione della metodologia adottata

Funzione utile per trasformare i dati

Rimozione di artefatti tramite filtro mediano (remove_square_noise):

È stato applicato un filtro mediano 3x3 per ogni immagine, utile per eliminare il rumore “quadratino” nei dati.

Prende in input un array di immagini (images), dove ogni immagine ha canali (RGB).
Per ogni immagine, applica un filtro mediano 3x3 a ogni canale (R, G, B) separatamente.
Ricompone i canali filtrati in un’unica immagine con np.stack.
Restituisce un nuovo array numpy di immagini filtrate, utile per rimuovere rumore quadrato.

Esempio semplice (griglia 3x3):
Immagina questa parte di un'immagine in scala di grigi (valori da 0 a 255):
Il pixel centrale è 255 (un chiaro esempio di "rumore").
I valori della finestra 3x3 ordinati sono:
[120, 123, 124, 125, 126, 127, 129, 130, 255]

120 125 130
123 255 127
124 126 129

Valore mediano: 126
Quindi, il pixel centrale 255 viene sostituito con 126.

Descrizione della metodologia adottata

1. Caricamento e preprocessing dei dati

Il dataset è stato caricato utilizzando la funzione ***np.load***, che consente di lavorare direttamente con array NumPy, velocizzando i tempi di lettura rispetto a formati immagine standard.

Etichette in vettori monodimensionali

Le etichette, in forma di array bidimensionale con una singola colonna, cioè con shape tipo (n_samples, 1) sono state trasformate con .ravel() in vettori monodimensionali, cioè con shape tipo (n_samples,) per garantire la corretta forma richiesta dai modelli durante l'addestramento e la valutazione.

Immagini in vettori monodimensionali

Le immagini di input, originariamente in formato multidimensionale, sono state appiattite in vettori monodimensionali per ogni campione usando .reshape(). Questo passaggio è necessario perché i modelli richiedono dati in forma di matrici 2D, dove ogni riga rappresenta un campione e ogni colonna una caratteristica.

Normalizzazione dei dati

I dati di input sono stati normalizzati utilizzando lo StandardScaler, che riduce ogni caratteristica ad una distribuzione con media zero e deviazione standard uno. La normalizzazione è stata effettuata per garantire una scala omogenea, e migliorare l'addestramento del modello.

Applicare PCA per ridurre dimensionalità

Per ridurre la complessità computazionale, è stata applicata la PCA, mantenendo il 95% della varianza totale. Questo passaggio permette di conservare le informazioni più rilevanti, eliminando ridondanze.

Descrizione della metodologia adottata

2. Definizione degli iperparametri

Per ottimizzare il modello **MLP(Multi-Layer Perceptron),** viene definito un insieme di possibili valori per diversi iperparametri tramite un dizionario chiamato param_grid.

Gli iperparametri considerati sono:

- **hidden_layer_sizes** tuple che indicano il numero di neuroni in ciascun layer nascosto (ad esempio, due layer con 256 e 128 neuroni);
- **alpha** parametro di regolarizzazione L2, che penalizza i pesi troppo grandi nella rete. Questa penalizzazione aiuta a evitare l'overfitting impedendo che il modello diventi troppo complesso e si adatti troppo ai dati di training, specialmente rumorosi.
- **activation** funzione di attivazione utilizzata dai neuroni;
- **solver** algoritmo di ottimizzazione per l'addestramento;
- **learning_rate_init** tasso di apprendimento iniziale;
- **max_iter** numero massimo di epoche di training, non necessariamente raggiunto grazie all'early stopping

Descrizione della metodologia adottata

3. Definizione parametri per l'early stopping e variabili di controllo

Early stopping

Per controllare l'allenamento del modello ed evitare overfitting, vengono definiti due parametri di early stopping manuale:

- **patience**: indica il numero massimo di epoche consecutive durante le quali l'accuratezza sul validation set può non migliorare significativamente prima di interrompere anticipatamente l'addestramento. In questo caso, è fissato a 10 epoche.
- **tolerance**: rappresenta la soglia minima di miglioramento richiesta per considerare un aumento effettivo dell'accuratezza di validazione, qui impostata a 1e-4.

Variabili di controllo

Per tracciare i risultati e mantenere il miglior modello trovato durante la ricerca degli iperparametri vengono inizializzate le varibili

- **best_val_acc_overall**: memorizza la migliore accuratezza di validazione riscontrata finora;
- **best_model**: conserva il modello corrispondente alla migliore performance;
- **best_params**: registra la combinazione di iperparametri che ha prodotto il miglior risultato;
- **best_train_accs e best_val_accs**: liste che tengono traccia dell'andamento dell'accuratezza rispettivamente su training e validation set durante le epoche, utili per analisi e visualizzazioni successive.

Descrizione della metodologia adottata

4. Addestramento epoca per epoca, early stopping e salvataggio dei pesi migliori

Generazione Modelli MLP

Grazie a **product()**, si generano tutte le possibili combinazioni degli iperparametri definiti.
Per ogni combinazione, viene creato un nuovo modello MLPClassifier con quei parametri, pronto per essere allenato un'epoca alla volta.

Inizializzazione Variabili

- **best_val_accuracy** → Memorizza la migliore accuratezza ottenuta sul set di validazione fino a quel momento.
- **epochs_without_improvement** → Tiene traccia di quante epoche consecutive non hanno portato a un miglioramento significativo dell'accuratezza di validazione (in base a una soglia definita, tolerance).
- **train_accuracies** → Lista che memorizza, epoca dopo epoca, l'accuratezza ottenuta sul set di training.
- **val_accuracies** → Lista analoga alla precedente, ma per le accuratezze ottenute sul set di validazione.
- **best_coefs** → Inizializza una variabile destinata a salvare i pesi (matrici dei coefficienti) del modello quando si ottiene la migliore accuratezza di validazione.
- **best_intercepts** → Inizializza una variabile per salvare i bias (termini di scostamento) associati ai neuroni.

Descrizione della metodologia adottata

4. Addestramento epoca per epoca, early stopping e salvataggio dei pesi migliori

Addestramento

Ad ogni epoca, il modello viene addestrato su tutto il training set chiamando mlp.fit().

Check Accuratezza

Vengono calcolate le predizioni e le accuratezze sia sul training che sul validation set, e vengono memorizzate nelle liste train_accuracies e val_accuracies per tracciare l'andamento del modello nel tempo

1. Se l'accuratezza sul validation set migliora in modo significativo rispetto al miglior risultato finora ottenuto (superando la soglia definita da tolerance), si aggiorna il valore migliore e si salvano i pesi e i bias correnti del modello, azzerando il contatore delle epoche senza miglioramento.
2. Se non si osserva un miglioramento, il contatore epochs_without_improvement viene incrementato.
3. Quando il numero di epoche consecutive senza miglioramento raggiunge il valore di patience, viene attivato l'early stopping e il ciclo si interrompe anticipatamente.

Se la migliore accuratezza di validazione ottenuta in questa iterazione supera quella globale salvata in best_val_acc_overall, il modello e i parametri associati vengono salvati come i migliori trovati finora, insieme alle serie di accuratezze di training e validazione per il successivo confronto e visualizzazione.

Descrizione della metodologia adottata

5. Valutazione del modello migliore e analisi delle performance

Dopo la ricerca tra tutte le combinazioni di iperparametri, vengono stampati:

best_params	la combinazione di iperparametri che ha prodotto la migliore accuratezza di validazione
best_val_acc_overall	il valore massimo di accuratezza ottenuto sul validation set durante l'intero processo di ricerca.

Segue la valutazione finale del modello migliore (best_model) sul test set:

1. Si calcolano le predizioni (test_preds) sul test set e si misura l'accuratezza finale (test_accuracy) con accuracy_score, per stimare la capacità del modello di generalizzare su dati mai visti.
2. Viene stampato un classification report dettagliato, che include precision, recall e F1-score per ciascuna classe.
3. Si genera una matrice di confusione visualizzata con Seaborn, che mostra come le predizioni si distribuiscono rispetto alle vere classi.

Descrizione della metodologia adottata

Strategie adottate per evitare l'overfitting

Pulizia del rumore nei dati	Prima dell'addestramento, è stato applicato un filtro mediano 3×3 alle immagini per rimuovere rumori locali. Questo aiuta il modello a concentrarsi su pattern significativi, riducendo il rischio di apprendere dettagli casuali non generalizzabili.
Standardizzazione dei dati	I dati sono stati normalizzati tramite `StandardScaler`, portando le feature su stessa scala. Questo rende l'ottimizzazione più stabile e veloce, riducendo la sensibilità del modello a feature dominanti.
Riduzione dimensionale con PCA	È stato applicato PCA per conservare il 95% della varianza dei dati. Questo passaggio ha permesso di ridurre la complessità del modello, limitando l'overfitting dovuto all'alta dimensionalità, mantenendo comunque l'informazione rilevante.
Regolarizzazione L2	È stato utilizzato un termine di regolarizzazione (`alpha`) nel modello `MLPClassifier`. Questo penalizza i pesi troppo grandi, incentivando il modello a soluzioni più semplici e meno soggette a overfitting.
Early Stopping	Durante l'addestramento epoca per epoca, è stato implementato early stopping manuale. Questo ha permesso di interrompere l'allenamento quando il modello non mostrava più miglioramenti sulla validazione, evitando di apprendere rumore o pattern non significativi.
Controllo del learning rate	learning_rate= adaptive, riduce automaticamente il tasso di apprendimento quando l'errore non migliora. Ciò previene oscillazioni o eccessivo adattamento ai dati di training.

Descrizione dei risultati

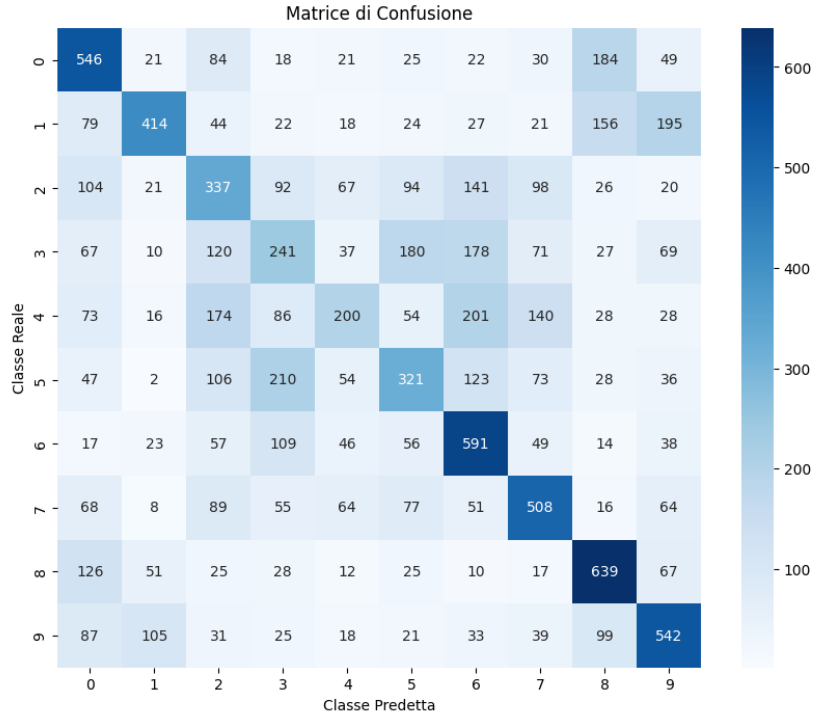
```
Migliori parametri trovati: {'hidden_layer_sizes': (512, 256, 128), 'alpha': 0.01, 'activation': 'relu', 'solver': 'sgd', 'learning_rate_init': 0.005, 'max_iter': 150}
Migliore accuratezza di validazione: 0.4412

Accuracy finale sul test set: 0.4339

Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.55      0.49      1000
           1       0.62      0.41      0.50      1000
           2       0.32      0.34      0.33      1000
           3       0.27      0.24      0.26      1000
           4       0.37      0.20      0.26      1000
           5       0.37      0.32      0.34      1000
           6       0.43      0.59      0.50      1000
           7       0.49      0.51      0.50      1000
           8       0.53      0.64      0.58      1000
           9       0.49      0.54      0.51      1000

    accuracy                           0.43     10000
   macro avg       0.43      0.43      0.43     10000
weighted avg       0.43      0.43      0.43     10000
```



La coerenza tra accuracy di validation e di test indica che il modello non ha overfittato.

Dall'analisi delle singole classi, si nota che alcune sono riconosciute meglio di altre, mentre alcune risultano più difficili da distinguere. Questo suggerisce che ci sono margini di miglioramento, ad esempio introducendo architetture più sofisticate o ulteriori tecniche di pre-elaborazione.