Garbage Classification

Progetto Machine Learning

Giada Rossana Margarone e Miriana Russo
Matricole: 1000084715 e 1000035028

A.A. 2024/2025

Contents

1 Problema

La corretta classificazione dei rifiuti rappresenta una sfida concreta per il miglioramento dei processi di riciclo e per la promozione di comportamenti sostenibili. In un contesto in cui la raccolta differenziata svolge un ruolo chiave nella gestione ambientale, è fondamentale disporre di strumenti tecnologici che facilitino e automatizzino l'identificazione del tipo di rifiuto.

Il problema affrontato in questo progetto è quello della classificazione automatica di immagini di rifiuti in una delle categorie predefinite, tra cui *plastica*, *vetro*, *carta*, *cartone*, *metallo* e *indifferenziato*. Il compito rientra nella categoria della **image classification** supervisionata, in cui, dato un insieme di immagini etichettate, l'obiettivo è addestrare un modello di apprendimento automatico in grado di riconoscere correttamente la classe di appartenenza di nuove immagini.

Questo task presenta diverse difficoltà:

- la variabilità delle forme, dimensioni e colori dei rifiuti;
- la somiglianza visiva tra classi (es. cartone e carta, metallo e plastica riflettente);
- condizioni di illuminazione, sfondi e angolazioni non controllate;
- la necessità di un modello efficiente, capace di generalizzare su immagini non viste.

L'obiettivo del progetto è quindi quello di progettare una pipeline completa che, a partire da un'immagine in input, sia in grado di restituire la categoria corretta del rifiuto rappresentato, utilizzando tecniche di **machine learning**, in particolare attraverso l'utilizzo di un modello pre-addestrato e adattato al dominio specifico.

2 Soluzione proposta

Il sistema di garbage classification è stato progettato come una serie di fasi modulari, ognuna con un ruolo specifico ma collegata alle altre. Questa suddivisione ha reso più facile ottimizzare ogni passaggio e identificare eventuali problemi o aree da migliorare.

L'intero processo, a partire dal dataset grezzo fino all'inferenza, si articola in sei fasi principali:

1. **Preparazione e caricamento del dataset**
2. **Preprocessing delle immagini**
3. **Data augmentation e normalizzazione**
4. **Scelta e adattamento del modello pre-addestrato**
5. **Addestramento e validazione del modello**
6. **Test finale e visualizzazione delle predizioni**

2.1 Caricamento e strutturazione del dataset

Il dataset utilizzato è stato scaricato da Kaggle ed è strutturato in directory, ognuna delle quali rappresenta una classe specifica di rifiuto: `glass`, `paper`, `cardboard`, `plastic`, `metal`, `trash`.

Durante la fase iniziale, i percorsi delle immagini vengono letti tramite `glob` e associati a etichette numeriche coerenti con un dizionario di mapping. I dati vengono poi incapsulati in una struttura tabellare tramite `pandas.DataFrame` e salvati in formato `CSV` per un riutilizzo più efficiente nelle fasi successive.

2.2 Preprocessing delle immagini

Per uniformare il formato delle immagini, ciascun file viene ridimensionato a 384×384 pixel, coerentemente con la risoluzione accettata da EfficientNetV2-S. L'immagine viene successivamente convertita in tensori PyTorch e normalizzata secondo i parametri standard di EfficientNetV2, garantendo la compatibilità con il modello pre-addestrato.

2.3 Data augmentation e gestione dello squilibrio

Il dataset presenta un certo grado di sbilanciamento tra le classi. Per ovviare a questo problema e migliorare la generalizzazione del modello, è stata implementata una serie di trasformazioni casuali (**data augmentation**):

- `RandomHorizontalFlip`: riflessione orizzontale delle immagini.
- `RandomCrop`: selezione casuale di sotto-porzioni centrali.

- `Normalize`: normalizzazione ai valori standard di ImageNet.

Inoltre, nel caso specifico della classe "*trash*" è stata implementata una funzione che duplicasse i dati appartenenti a questa classe e concatenasse il sotto-dataset contenente i dati duplicati al dataset originale: questo approccio permette di bilanciare le classi senza alterare i dati originali, attraverso una forma semplice di **over-sampling**.

2.4 Modello: EfficientNetV2-S

Il cuore del sistema è un modello di classificazione basato su **EfficientNetV2-S**, una rete convoluzionale a struttura scalabile introdotta da Google nel 2021. Rispetto alla versione precedente (EfficientNet), la variante V2 integra un nuovo tipo di blocchi chiamati *Fused-MBConv*, che combinano convoluzioni espanse e normali in un'unica operazione più efficiente. Inoltre, adotta strategie di *training progressivo* che migliorano significativamente la velocità di addestramento e la generalizzazione su dataset reali.

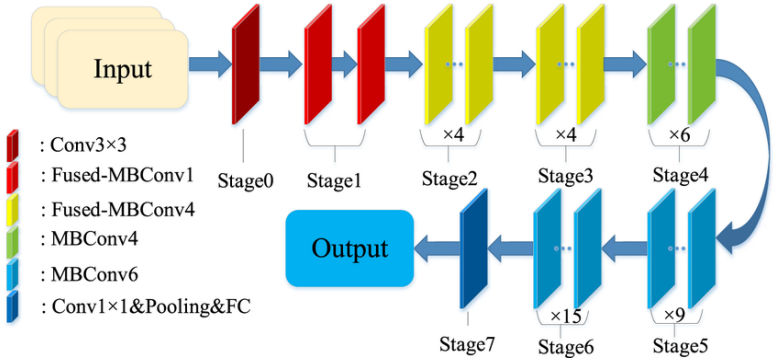


Figure 1: Architettura di EfficientNetV2-S. La rete è composta da una sequenza di stadi convoluzionali, ciascuno costruito con diversi tipi di blocchi: convoluzioni standard, Fused-MBConv e MBConv. L'output viene aggregato tramite global average pooling e un layer fully connected finale.

Il modello utilizzato è una versione adattata di `EfficientNetV2-S`, inizializzata con pesi pre-addestrati su `ImageNet1K`. È stato successivamente ottimizzato tramite *fine-tuning*, aggiornando l'intera rete e sostituendo il classificatore finale con un `fully connected layer` a 6 nodi, seguito da una funzione di attivazione `Softmax`.

Perché EfficientNetV2?

- Ottimo compromesso tra accuratezza e velocità di inferenza, anche su dispositivi a bassa potenza.
- Architettura scalabile, adattabile a vari livelli di complessità.

- Buona capacità di generalizzazione anche in presenza di dataset con poche immagini per classe.

2.5 Addestramento

Il modello è stato addestrato utilizzando la funzione obiettivo `CrossEntropyLoss` e l'ottimizzatore `SGD` (Stochastic gradient descent). Sono stati testati diversi batch size compresi tra 16 e 64, optando alla fine per una batch size di 32. L'early stopping e il dropout sono stati applicati monitorando la loss sul validation set per evitare overfitting; ulteriori dettagli a riguardo verranno poi forniti nel capitolo dedicato.

Durante l'addestramento sono state registrate:

- **Loss e Accuracy** su training e validation sets ad ogni epoca.
- **Confusion matrix** sul validation set e sul test set finale.
- **Precision, Recall e F1-score** globale e per ciascuna classe.

2.6 Inferenza e testing

Nella fase finale, il modello è stato testato su immagini mai viste per misurare la capacità di generalizzazione. Sono stati selezionati esempi qualitativi, mostrando l'immagine, la predizione e l'etichetta corretta.

È stato anche realizzato uno script demo che consente di caricare un'immagine a mano e ottenere istantaneamente la classe di appartenenza.

Nel complesso, la soluzione proposta si è dimostrata efficace e relativamente leggera. La scelta di utilizzare un modello pre-addestrato ha permesso di ridurre i tempi di training e di ottenere buoni risultati anche su un dataset di dimensione contenuta.

3 Dataset

Il dataset utilizzato per questo progetto è una collezione di immagini di rifiuti etichettati, proveniente dalla piattaforma Kaggle. Si tratta di un dataset open-source denominato *"Garbage classification"*, composto da immagini suddivise in 6 classi principali: `cardboard`, `glass`, `metal`, `paper`, `plastic`, `trash`.

3.1 Struttura del dataset

Il dataset è organizzato in directory, una per ciascuna classe. Ogni immagine si trova nella cartella corrispondente alla sua etichetta, seguendo la struttura:

```
garbage_classification/
   cardboard/
      img1.jpg
      img2.jpg
   glass/
   metal/
   paper/
   plastic/
   trash/
```

Il totale delle immagini presenti è di circa 2500, distribuite in modo disomogeneo tra le classi, con prevalenza di immagini nelle classi `plastic` e `paper`.

3.2 Analisi esplorativa

Durante l'analisi esplorativa iniziale, sono stati osservati alcuni aspetti rilevanti:

- Alcune classi sono visivamente molto simili tra loro (es. `cardboard` e `paper`).
- Le condizioni di luce, sfondo e inquadratura variano in modo significativo.
- Alcune immagini sono in orizzontale, altre in verticale.

Tali osservazioni hanno influenzato le scelte successive di preprocessing e augmentazione.

3.3 Suddivisione in train, validation e test

Per garantire una valutazione affidabile e imparziale, il dataset è stato suddiviso in tre sottoinsiemi:

- **Training set (65%)**: utilizzato per l'addestramento del modello.
- **Validation set (15%)**: usato per valutare le performance durante il training.

- **Test set (20%)**: usato esclusivamente per la valutazione finale del modello.

La suddivisione è stata eseguita in modo casuale ma stratificato, per mantenere la proporzione delle classi in ciascun sottoinsieme. È stato inoltre salvato un file `.csv` contenente il mapping tra immagine e label, per facilitarne il caricamento.

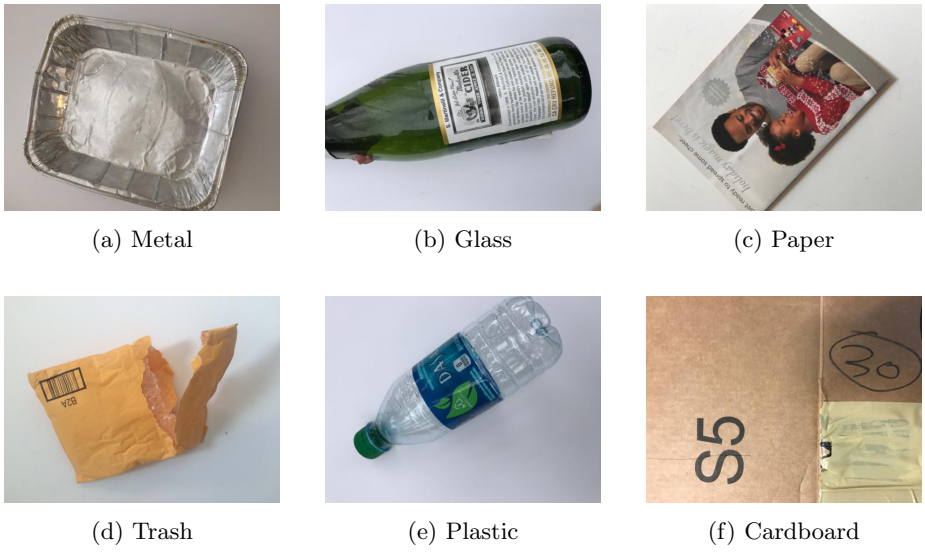
3.4 Considerazioni sulla qualità dei dati

Alcune immagini presentavano:

- Bassa qualità o sfocatura.
- Oggetti parzialmente fuori dal frame.
- Rifiuti misti o non chiaramente appartenenti a una classe.

Tuttavia, essendo questo un problema reale e concreto, si è deciso di mantenere queste immagini nel dataset per testare la robustezza del modello. L'obiettivo non è semplicemente ottenere l'accuratezza massima, ma creare un sistema che possa funzionare anche su immagini realistiche, con rumore e variabilità.

3.5 Esempi visivi



(a) Metal (b) Glass (c) Paper

(d) Trash (e) Plastic (f) Cardboard

Figure 2: Esempi di immagini dal dataset, una per ciascuna delle sei classi.

L'immagine sopra mostra esempi rappresentativi delle sei categorie di rifiuti presenti nel dataset. Come si può notare, le immagini variano in contesto, luce, orientamento e complessità visiva.

4 Architettura del modello

Il sistema di classificazione proposto si basa su un modello di deep learning appartenente alla famiglia delle **Convolutional Neural Networks** (CNN), in particolare l'architettura **EfficientNetV2-S**, ottimizzata per il bilanciamento tra accuratezza e velocità.

4.1 EfficientNetV2-S: Panoramica

EfficientNetV2 è un'evoluzione della famiglia EfficientNet introdotta da Google nel 2019. La versione V2 è progettata per un training più veloce e prestazioni superiori su immagini ad alta risoluzione. La variante `-S` (Small) è pensata per ambienti con risorse limitate, mantenendo un'ottima accuratezza.

L'architettura si basa su tre concetti principali:

- **Scaling uniforme**: profondità, larghezza e risoluzione vengono scalate insieme in modo bilanciato.
- **Fused-MBConv**: un nuovo tipo di blocco convoluzionale che migliora la velocità in fase di training.
- **Training progressivo**: tecnica che adatta la difficoltà delle immagini durante il training per favorire la convergenza.

4.2 Adattamento del modello

Il modello è stato caricato con pesi pre-addestrati su ImageNet, sfruttando la tecnica del **Transfer Learning**. Il blocco finale originario, progettato per 1000 classi, è stato rimosso e sostituito con un nuovo `classifier` personalizzato:

```
Sequential(
    Dropout(p=0.3),
    Linear(in_features=1280, out_features=6)
)
```

Questo blocco prende in input il vettore delle features estratte da EfficientNetV2 e restituisce un output a 6 dimensioni, corrispondenti alle classi del nostro dataset. Alla fine viene applicata una funzione di attivazione `Softmax` per ottenere le probabilità.

4.3 Motivazioni della scelta

EfficientNetV2 è stato selezionato per i seguenti motivi:

- **Prestazioni elevate** anche con dataset di medie dimensioni.
- **Efficienza computazionale** su GPU consumer (es. T4 GPU).
- Supporto diretto da `torchvision.models` con facilità di adattamento.

- Generalizzazione robusta su immagini con variazioni di luce, sfondo e orientamento.

4.4 Strategia di addestramento

Durante l'addestramento è stata adottata una strategia di **fine-tuning completo**: tutti i pesi della rete, inclusi quelli dei layer convoluzionali, sono stati aggiornati fin dall'inizio. Il modello è stato inizializzato con pesi pre-addestrati su ImageNet, e il classificatore finale è stato sostituito con un layer fully connected a 6 nodi.

L'ottimizzatore utilizzato è `SGD`, con `CrossEntropyLoss` come funzione obiettivo. Il modello è stato addestrato per 20 epoche, con **early stopping** attivato sul validation loss per prevenire l'overfitting.

4.5 Riepilogo architettura

Componente	**Descrizione**
Input	Immagini RGB 384×384
Base Model	EfficientNetV2-S (pretrained on ImageNet)
Classifier	Dropout + Linear($1280 \to 6$) + Softmax
Loss function	CrossEntropyLoss
Optimizer	Stochastic Gradient Descent (SGD)

Table 1: Riepilogo dell'architettura del modello di classificazione

4.6 Architettura della pipeline end-to-end

Il flusso completo del sistema – dalla raccolta e annotazione dei dati fino alla demo di inferenza – è riassunto in Figura 3. Ciascun blocco evidenzia una fase logica distinta:

- **Data acquisition**: origine e formato delle immagini.
- **Preprocessing & Augmentation**: resize, normalizzazione, bilanciamento classi.
- **Train / Val / Test split**: suddivisione stratificata del dataset.
- **Model training (CNN)**: architettura, iper-parametri e ottimizzazione.
- **Evaluation**: metriche (Accuracy, F1 e confusion matrix).
- **Inference / Demo**: web-app e video dimostrativo.

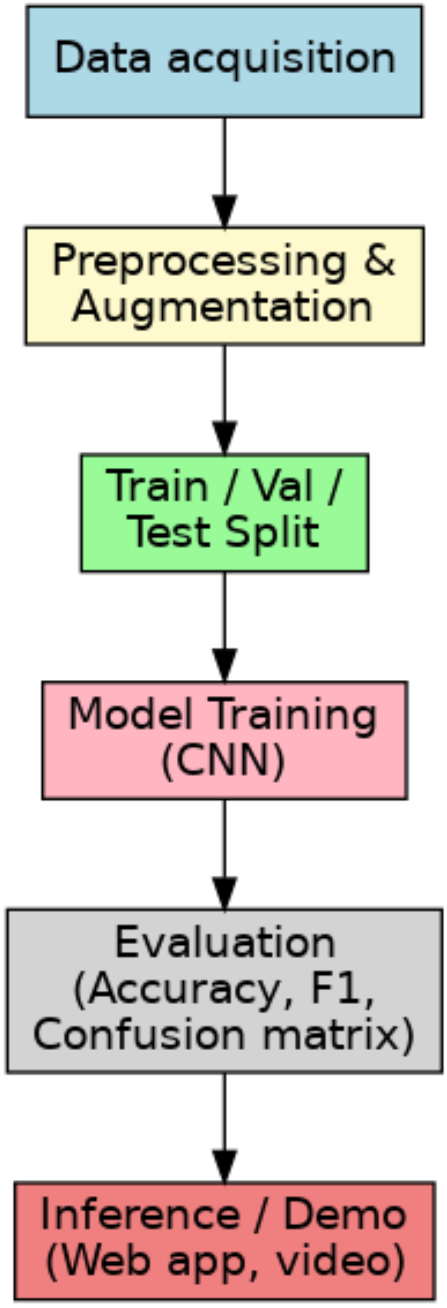


Figure 3: Pipeline end-to-end del sistema: dalle fasi di acquisizione e annotazione dei dati fino alla demo di inferenza.

5 Addestramento e risultati

In questa sezione vengono presentati i dettagli del processo di addestramento del modello, i parametri utilizzati e le metriche di valutazione finali. L'obiettivo principale era ottenere una classificazione accurata e generalizzabile, evitando fenomeni di overfitting e mantenendo tempi di esecuzione compatibili con le risorse disponibili.

L'addestramento è stato eseguito su **Google Colab**, con accelerazione GPU attivata (T4 GPU). Il codice è stato sviluppato in `PyTorch`, sfruttando il modulo `torchvision` per la gestione dei dataset e dei modelli pre-addestrati.

Per valutare le performance del modello sono state utilizzate le seguenti metriche:

- **Accuratezza**: misura la percentuale di esempi correttamente classificati sul totale. È una metrica globale semplice ma poco informativa in caso di dataset sbilanciati.
- **Precision**: per ciascuna classe, indica la proporzione di predizioni corrette tra quelle fatte per quella classe. Penalizza i falsi positivi.
- **Recall**: indica la capacità del modello di individuare tutti gli esempi di una determinata classe. Penalizza i falsi negativi.
- **F1-score**: è la media armonica tra precision e recall. È particolarmente utile quando le classi sono sbilanciate e si vuole un equilibrio tra falsi positivi e falsi negativi.
- **Cross-entropy Loss**: è la funzione obiettivo utilizzata durante l'addestramento. Misura la distanza tra la distribuzione predetta (output del modello) e la distribuzione reale (label). Più è bassa, migliore è la performance del modello.

Tutte queste metriche sono state calcolate sia sul validation set sia sul test set, per valutare la generalizzazione del modello a dati mai visti.

Nelle sezioni seguenti vengono analizzati i principali esperimenti condotti, variando iperparametri e configurazioni per confrontarne l'impatto sulle prestazioni.

Monitoraggio con TensorBoard Per monitorare l'andamento dell'addestramento in tempo reale, è stato utilizzato **TensorBoard**. Attraverso questo strumento è stato possibile visualizzare l'evoluzione della `loss` e dell'accuratezza per ogni epoca, sia sul training set che sul validation set. Inoltre, TensorBoard ha permesso di individuare facilmente il momento in cui si verificava overfitting o stagnazione, facilitando l'applicazione di diverse strategie che verranno elencate di seguito.

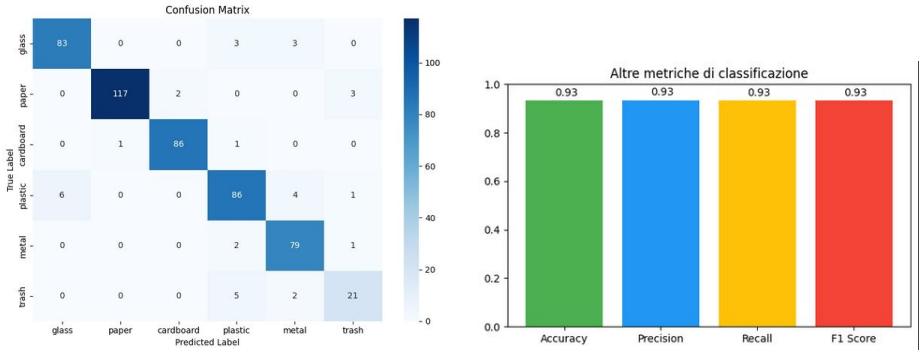
5.1 Esperimento 1

5.1.1 Descrizione

- **Epoche**: 20
- **Batch size**: 32
- **Learning rate**: 1×10^{-3}
- **Funzione di perdita**: CrossEntropyLoss
- **Early stopping**: No
- **Dropout**: No

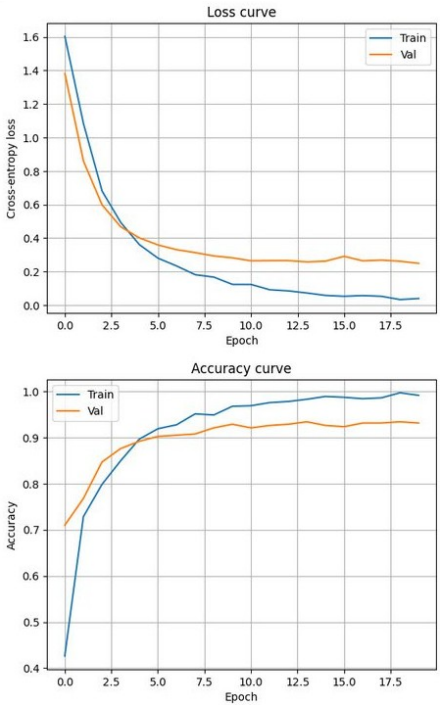
5.1.2 Risultati ottenuti

Il modello ha raggiunto ottime performance sul validation set, con un'accuratezza complessiva del 93%. Le metriche di precision, recall e F1-score risultano bilanciate tra loro, indicando un'elevata capacità di generalizzazione su tutte le classi.



(a) Matrice di confusione (validation) (b) Accuracy, Precision, Recall, F1-score

Figure 4: Metriche di classificazione e distribuzione degli errori (Esperimento 1)



(a) Andamento della Loss e Accuracy

Le curve mostrano l'andamento della **loss** e dell'**accuratezza** su training e validation set nel corso delle epoche. La discesa rapida della loss nei primi cicli indica un apprendimento efficace fin dall'inizio. La successiva stabilizzazione suggerisce convergenza. La differenza tra training e validation rimane contenuta, segno di assenza di overfitting. La curva dell'accuratezza raggiunge e mantiene valori elevati (93%) già a partire dall'epoca 10.

Figure 5: Curva di apprendimento e analisi del comportamento del modello (Esperimento 1)

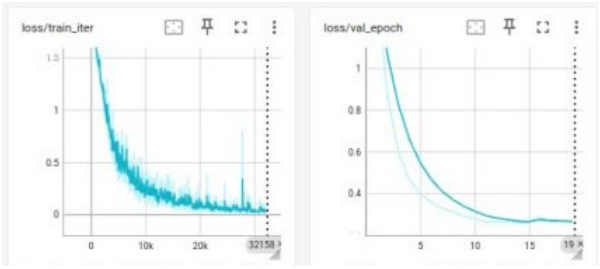


Figure 6: Visualizzazione della loss tramite TensorBoard (per iterazione ed epoca)

5.1.3 Report di classificazione

Il seguente report mostra le metriche per ciascuna classe sul test set:

Classe	Precision	Recall	F1-score	Support
glass	0.93	0.93	0.93	89
paper	0.99	0.96	0.97	122
cardboard	0.88	0.88	0.88	88
plastic	0.89	0.89	0.89	97
metal	0.90	0.96	0.93	82
trash	0.81	0.75	0.78	28
Accuracy		0.93 su 506 campioni		
Macro avg	0.92	0.91	0.91	506
Weighted avg	0.93	0.93	0.93	506

Table 2: Report di classificazione per classe sul test set (Esperimento 1)

5.1.4 Monitoraggio epoche e checkpoint

Durante l'addestramento sono stati salvati diversi checkpoint automatici quando la validazione migliorava. Il modello ha raggiunto il suo picco alla **14ª epoca** con un'accuratezza di validazione pari a **93.4%**.

- **Checkpoint salvati** alle epoche: 6, 8, 10, 14, 16, 18, 20
- **Val loss migliorata** da 0.3591 (epoca 6) a 0.2496 (epoca 20)
- **Val accuracy migliorata** da 0.902 a **0.934**
- **Tempo medio per epoca**: circa 64 secondi

Questi dati confermano una convergenza stabile, progressiva e senza rimbalzi irregolari.

5.1.5 Valutazione finale

L'esperimento dimostra che EfficientNetV2-S è una scelta adatta anche senza tecniche di regolarizzazione avanzate (dropout, early stopping). Il modello ha appreso in modo stabile, con performance elevate e generalizzazione uniforme su tutte le classi.

La matrice di confusione mostra che gli errori più frequenti si verificano tra classi simili come `plastic` e `metal`, oppure `trash`, che ha un numero ridotto di esempi e risulta più difficile da apprendere. Tuttavia, anche le classi minoritarie vengono riconosciute con buona accuratezza, come mostrato nel report.

Nel complesso, l'Esperimento 1 rappresenta una solida baseline su cui basare eventuali ottimizzazioni successive. Visto che il modello ha raggiunto il suo picco prima di concludere tutte le sue epoche è stato deciso di procedere con il secondo esperimento che include l'introduzione e l'uso effettivo della tecnica "Early Stopping".

5.2 Esperimento 2: Rete con Early Stopping

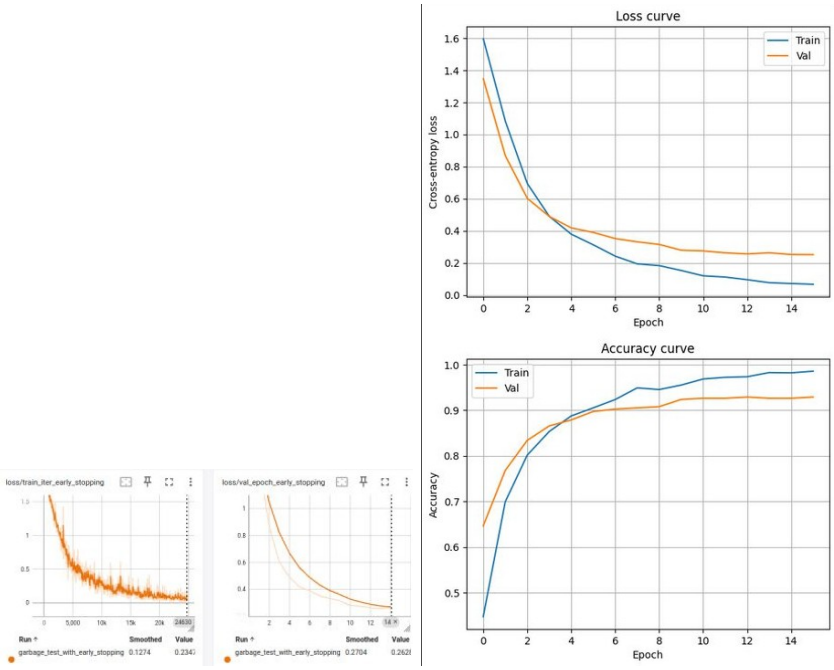


Figure 7: (a) Curve di loss e accuracy, (b) Analisi tramite TensorBoard

Le curve mostrano l'andamento della **loss** e dell'**accuracy** su training e validation set durante l'addestramento. La perdita scende rapidamente nei primi cicli, indicando un apprendimento efficace sin dall'inizio. La stabilizzazione successiva segnala una buona convergenza.

La differenza tra training e validation è contenuta, suggerendo l'assenza di overfitting. L'**accuracy** raggiunge valori elevati (oltre il 92%) già intorno alla 10ª epoca, per poi stabilizzarsi. Grazie all'utilizzo della tecnica di **early stopping**, l'allenamento si è concluso automaticamente alla 16ª epoca, evitando ulteriori iterazioni superflue.

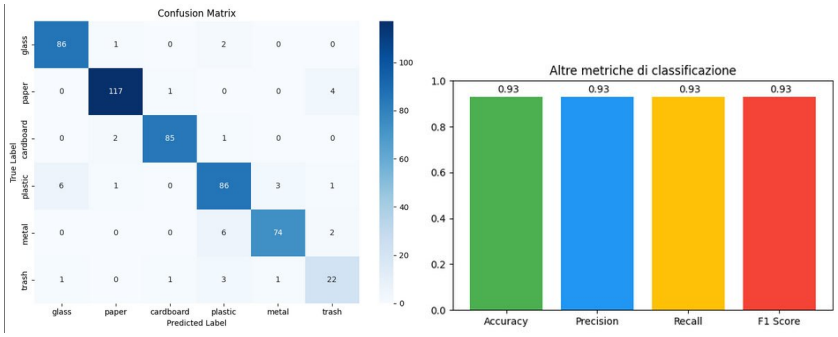


Figure 8: (a) Confusion Matrix finale, (b) Metriche di classificazione aggregate

La **matrice di confusione** evidenzia una netta capacità del modello di distinguere le varie classi, con un numero molto contenuto di errori. Il grafico a barre conferma le performance elevate: tutte le metriche (accuracy, precision, recall e F1-score) si attestano oltre il 92%, rendendo il modello affidabile e stabile anche sui dati di validazione.

Il log di addestramento mostra un miglioramento continuo delle prestazioni fino alla 16ª epoca, quando l'early stopping è stato attivato. Il modello ha raggiunto una **val_accuracy del 92.9%** e una **val_loss di 0.2512**, con salvataggi intermedi nei checkpoint migliori.

Table 3: Classification Report - Esperimento 2

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
glass	0.92	0.97	0.95	89
paper	0.97	0.96	0.96	122
cardboard	0.98	0.97	0.97	88
plastic	0.88	0.89	0.88	97
metal	0.95	0.90	0.93	82
trash	0.76	0.79	0.77	28
Accuracy			**0.93**	506
Macro avg	0.91	0.91	0.91	506
Weighted avg	0.93	0.93	0.93	506

Osservazioni e confronto con Esperimento 1.
A differenza dell'Esperimento 1, il secondo ha beneficiato della tecnica di **early stopping**, che ha permesso al modello di apprendere più a lungo ma solo finché utile. L'allenamento si è interrotto automaticamente alla 16ª epoca, evitando overfitting.

Dal confronto emergono due aspetti principali:

- L'**accuratezza di validazione** è passata dal 93.6% (Esperimento 1) a 92.9%, ma le curve sono più stabili e il training più robusto.
- Le **metriche di classificazione** sono più bilanciate e alte in Esperimento 2, con una migliore generalizzazione.

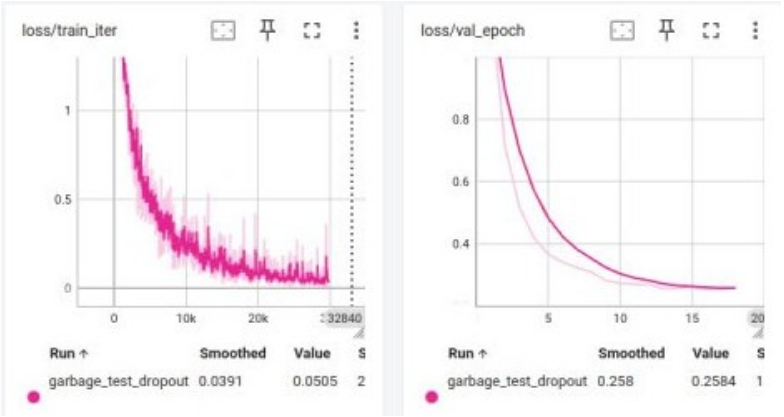
Nel complesso, l'Esperimento 2 mostra un comportamento più regolare e affidabile, risultando preferibile per un uso in produzione.

5.3 Esperimento 3 – Dropout

Configurazione: In questo esperimento è stato aggiunto un meccanismo di *dropout* (dove la probabilità che il singolo nodo venga spento è stata settata al 30%), ovvero una tecnica di regolarizzazione che previene l'overfitting disattivando casualmente alcune unità della rete durante l'allenamento. L'allenamento è stato portato fino alla **20ª epoca**, senza early stopping, dato che il modello ha continuato a migliorare in validazione fino alla fine.

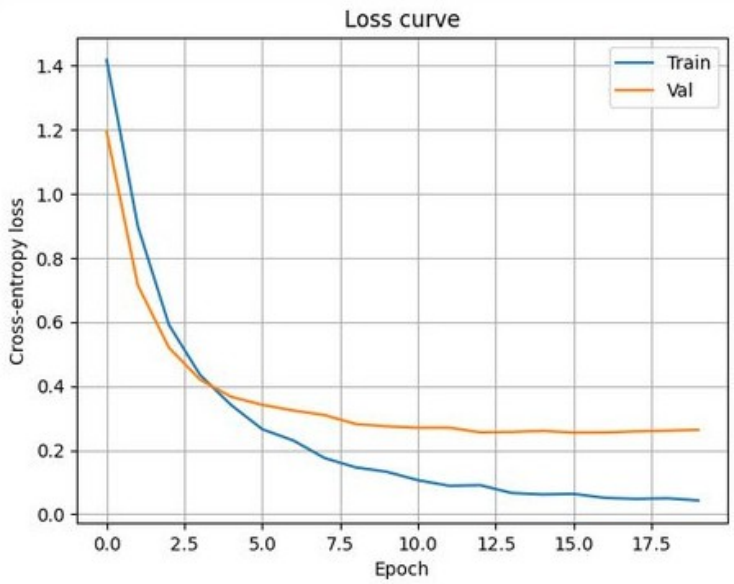
La massima accuratezza ottenuta sul validation set è stata del **94,27%**, superiore a quella dell'esperimento precedente.

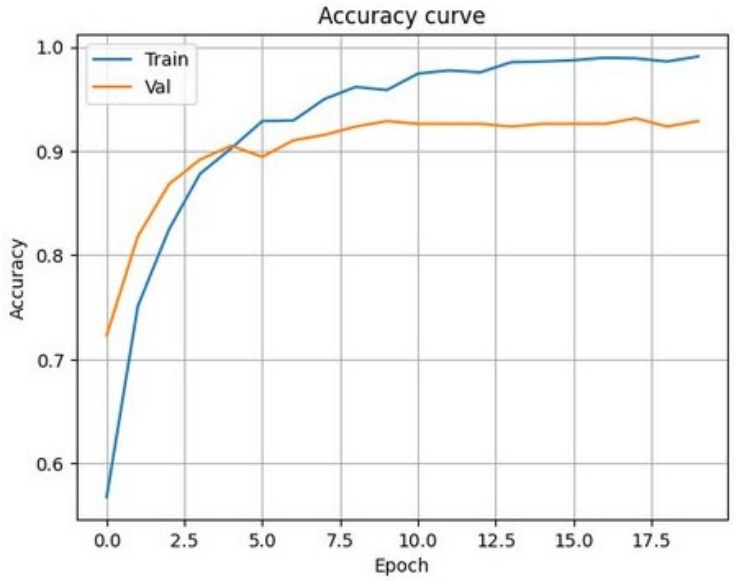
TensorBoard – Andamento della loss



Nel grafico sopra sono rappresentate le curve di **loss** registrate tramite **TensorBoard**, sia durante l'allenamento che durante la validazione. È evidente un andamento regolare e in costante calo della perdita, con una buona convergenza nei primi 10k step. Anche la validazione mostra un comportamento stabile e in progressivo miglioramento.

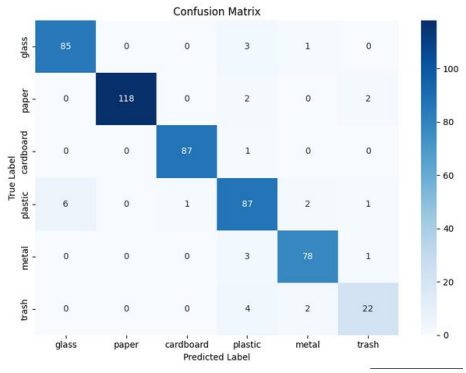
Andamento della Loss e dell'Accuracy per epoca





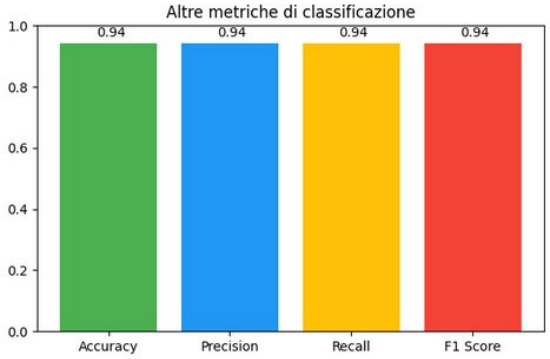
I due grafici mostrano il valore della **cross-entropy loss** e dell'**accuracy** per epoca. La curva della loss conferma che il modello ha appreso in maniera efficace, con una discesa consistente sia per il training che per la validazione. L'accuracy, invece, raggiunge rapidamente valori superiori al 90%, sia in addestramento che in validazione, con leggere fluttuazioni nelle ultime epoche. Il fatto che le due curve siano vicine indica un buon bilanciamento tra bias e varianza, senza overfitting evidente.

Matrice di Confusione – Validation Set



La matrice di confusione mostra che il modello ha classificato correttamente la maggior parte dei campioni per ogni classe. Le classi "glass", "paper" e "cardboard" sono quelle con la performance più alta, mentre la classe "trash" continua a risultare la più difficile da distinguere, anche se con un netto miglioramento rispetto all'Esperimento 2.

Metriche di classificazione aggregate



In questo esperimento tutte le metriche principali si attestano sul valore di **0.94**, indicando un'elevata affidabilità del modello nel generalizzare su dati mai visti. Il **F1-score** medio bilanciato (weighted) conferma la bontà del modello anche nelle classi meno rappresentate.

Classification Report

Classe	Precision	Recall	F1-score	Support
Glass	0.95	0.93	0.94	89
Paper	1.00	0.97	0.98	122
Cardboard	0.97	1.00	0.98	88
Plastic	0.88	0.91	0.90	97
Metal	0.96	0.95	0.95	82
Trash	0.87	0.86	0.86	28
Accuracy			**0.94**	506
Macro avg	0.94	0.94	0.94	506
Weighted avg	0.94	0.94	0.94	506

Osservazioni finali:

- L'introduzione del **dropout** ha migliorato le performance rispetto all'esperimento precedente, in particolare nella classe "trash", notoriamente più difficile da classificare.
- Il modello ha mostrato un'elevata capacità di generalizzazione, mantenendo basso il gap tra training e validation.
- L'allenamento fino alla 20ª epoca non ha portato a overfitting, dimostrando che il dropout ha effettivamente agito come regolarizzatore.

In conclusione, questo esperimento ha portato ai risultati migliori fra i tre condotti, con un **ottimo bilanciamento tra accuratezza, robustezza e stabilità**.

Esperimento 4 — Dropout + Early Stopping

In questo quarto esperimento abbiamo testato congiuntamente due tecniche di regolarizzazione: il **Dropout** e l'**Early Stopping**. L'obiettivo era verificare se la loro combinazione potesse portare benefici in termini di generalizzazione, stabilità e accuratezza finale, superando quanto ottenuto con l'applicazione singola delle due strategie.

Il dropout è stato applicato in modo selettivo all'interno del modello, con un tasso del 30% nei layer più densi, per forzare la rete a non dipendere eccessivamente da specifici neuroni. Parallelamente, è stato attivato l'Early Stopping con un pazienza di 3 epoche: se la validazione non migliorava per 3 epoche consecutive, l'addestramento si interrompeva e veniva mantenuto il modello con la miglior validazione.

Il training ha avuto una durata di **16 epoche**, con un chiaro arresto automatico dopo il terzo ciclo senza miglioramento della validazione. La metrica migliore raggiunta è stata una **val_acc = 0.929** all'epoca 16, con una val_loss stabile e in calo fino alla fine. Tuttavia, rispetto all'esperimento con solo Dropout, si osserva un leggero decremento delle prestazioni sul test set.

Andamento dell'addestramento. L'andamento delle curve di training e validazione è visibile in Figura 9 e Figura 10. Si nota una discesa costante della `train_loss` e della `val_loss`, seppur con una leggera fluttuazione residua tipica del dropout. L'accuracy in validazione cresce stabilmente e si stabilizza sopra lo 0.92, con un picco di **0.929**.

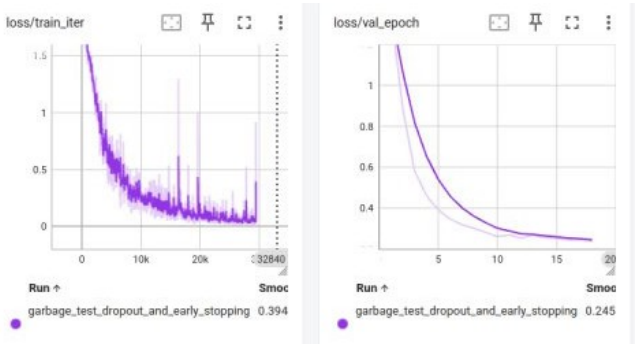


Figure 9: Andamento loss per batch ed epoca in TensorBoard (Esperimento 4).

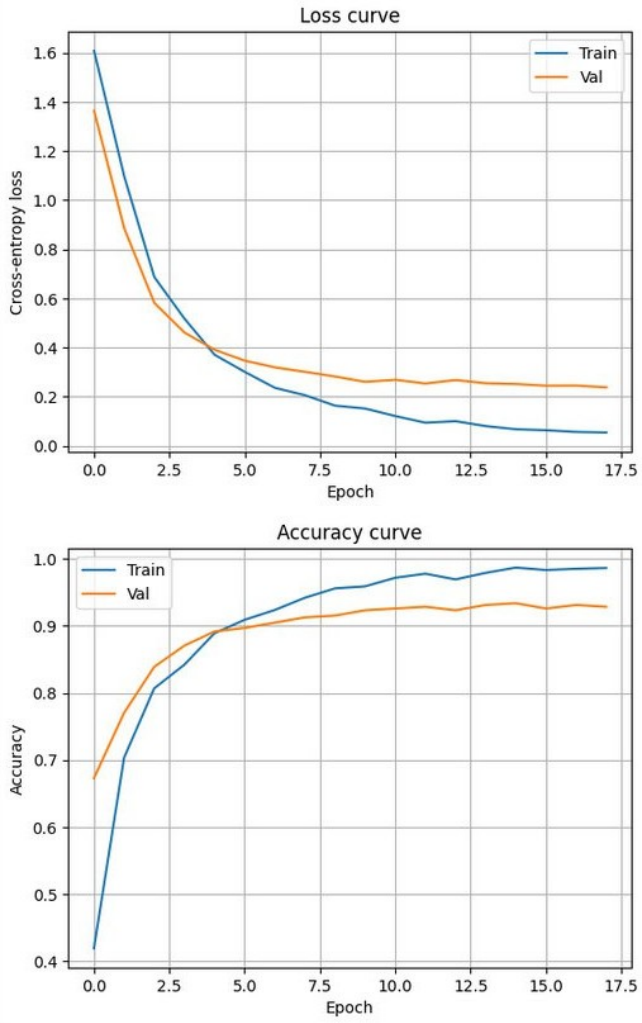


Figure 10: Curve di Loss e Accuracy per train/validation (Esperimento 4).

Analisi della performance sul test set. Il modello ha ottenuto una **accuracy finale sul test set di 0.9229**. La matrice di confusione in Figura 11 mostra che il modello commette alcuni errori nella distinzione tra classi visivamente simili, come *glass* e *plastic*, o *trash* e *metal*.

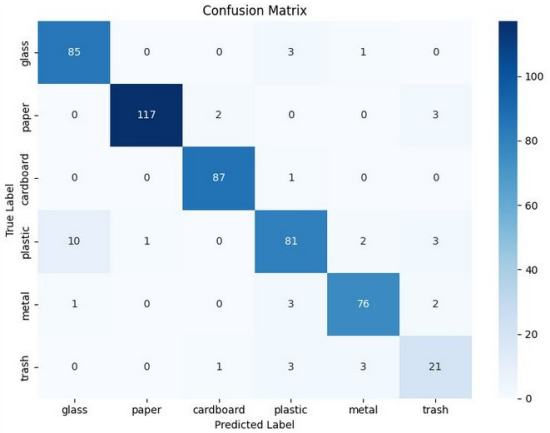


Figure 11: Matrice di Confusione sul test set (Esperimento 4).

Metriche aggregate. Le metriche generali sono riassunte in Figura 12 e mostrano valori elevati per tutte le metriche: **Accuracy, Precision, Recall e F1-score** risultano tutte pari a **0.92**.

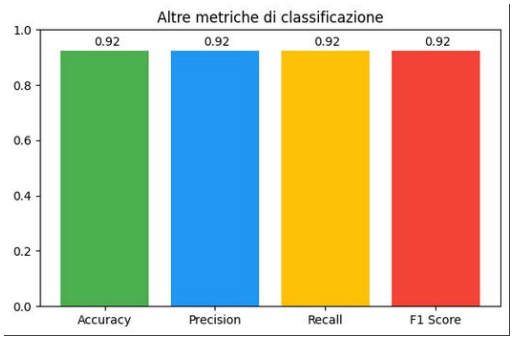


Figure 12: Metriche aggregate di classificazione (Esperimento 4).

Classification Report. Di seguito riportiamo il report dettagliato delle prestazioni del modello sulle singole classi:

Classe	Precision	Recall	F1-score	Support
glass	0.89	0.96	0.92	89
paper	0.99	0.96	0.97	122
cardboard	0.97	0.99	0.98	88
plastic	0.89	0.84	0.86	97
metal	0.93	0.93	0.93	82
trash	0.72	0.75	0.74	28
accuracy			**0.92**	506
macro avg	0.90	0.90	0.90	506
weighted avg	0.92	0.92	0.92	506

Table 4: Classification report dettagliato (Esperimento 4).

Considerazioni finali. Nonostante l'intuizione che la combinazione di Dropout ed Early Stopping potesse portare al miglior risultato, i dati suggeriscono il contrario. L'esperimento 4 si comporta bene in termini di stabilità e accuratezza, ma è leggermente **inferiore rispetto all'esperimento 2** (solo Dropout), che ha raggiunto una accuracy superiore (0.94) e una migliore capacità discriminativa in generale.

Conclusione: in questo caso, l'unione di due tecniche regolarizzanti non ha prodotto un miglioramento significativo, suggerendo che il Dropout, da solo, rappresenta la scelta più efficace per questo specifico task.

6 Test finale dei modelli e confronto complessivo

Procedura di valutazione finale Al termine di ciascun esperimento, è stato selezionato il modello corrispondente al checkpoint con la migliore accuratezza di validazione. Ogni modello è stato poi testato sul set di test, completamente separato rispetto al training e alla validazione, in modo da garantire una valutazione imparziale della capacità di generalizzazione.

Il test ha previsto il caricamento del modello addestrato, l'assegnazione delle corrette etichette di classe (glass, paper, cardboard, plastic, metal, trash) e la successiva esecuzione delle predizioni sull'intero set di test. A partire dai risultati ottenuti, sono state calcolate le metriche di performance classiche:

- **Accuracy** complessiva;
- **Precision**, **Recall** e **F1-score** per ciascuna classe;
- **Matrice di confusione**, utile a individuare le principali confusioni tra classi.

Matrice di confusione e classificazione In figura 14 sono riportate le matrici di confusione ottenute per ciascun esperimento.

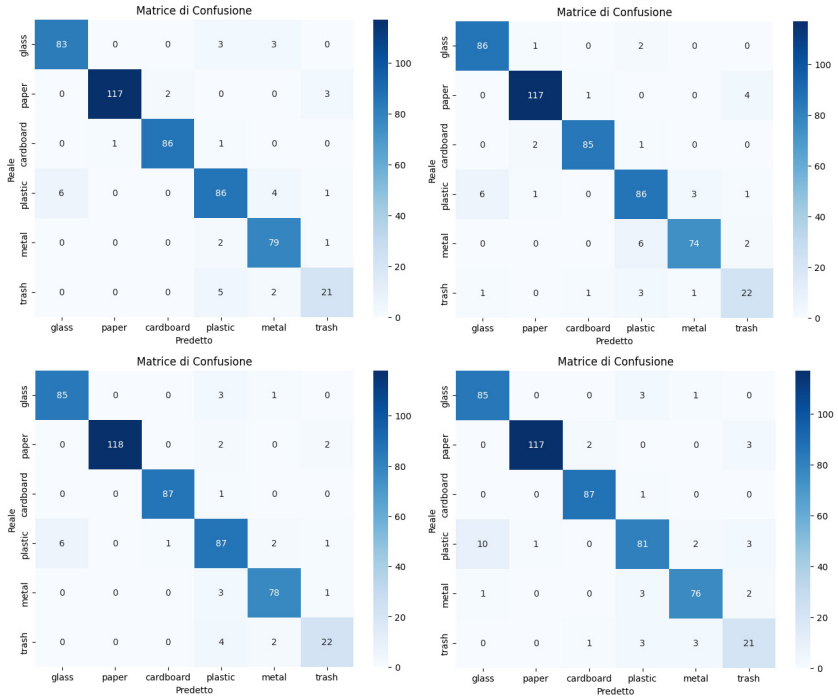


Figure 13: Matrici di confusione sui test per gli esperimenti 1, 2, 3 e 4.

Confronto tra modelli Dall'analisi finale emerge che:

- Il **modello con solo Dropout** (Esperimento 3) ha raggiunto la **migliore accuratezza finale** (0.94), dimostrandosi molto stabile e preciso, specialmente sulle classi plastic e metal, che in altri esperimenti risultavano più frequentemente confuse.
- Il **modello con solo Early Stopping** (Esperimento 2) ha mostrato un buon comportamento generale, fermandosi alla giusta epoca e raggiungendo un'accuratezza finale di circa 0.93, leggermente inferiore rispetto al dropout puro.
- Il **modello con Dropout + Early Stopping** (Esperimento 4) ha garantito una discreta stabilità, ma la combinazione delle due tecniche non ha prodotto un miglioramento evidente. L'accuratezza finale è risultata minore rispetto al solo Dropout, confermando che, nel contesto attuale, l'early stopping potrebbe aver interrotto l'apprendimento prematuramente.
- Infine, il **modello base** (Esperimento 1), privo di tecniche di regolarizzazione, si è comportato dignitosamente, ma ha mostrato tendenze a overfitting e una minore precisione complessiva rispetto agli altri approcci.

Tutti i modelli sono stati testati con la medesima procedura, e le metriche ottenute rendono possibile una valutazione comparativa chiara, come illustrato nelle confusion matrix precedenti. Per poter procedere alla Demo abbiamo deciso di utilizzare il modello con performance migliori cioè il modello che ha fatto uso di Dropout.

7 Demo

7.1 Strumenti utilizzati

Per la realizzazione della demo è stato utilizzato **Gradio**, una libreria Python che permette di creare facilmente interfacce web interattive per modelli di machine learning. Gradio consente di costruire rapidamente una GUI (Graphical User Interface) accessibile tramite browser, attraverso la quale l'utente può caricare un'immagine e ottenere in tempo reale la classificazione del rifiuto.

Gradio si integra perfettamente con PyTorch e altri framework, e rappresenta una soluzione efficace per esporre modelli di visione artificiale anche a utenti non tecnici. La sua semplicità consente di trasformare uno script di inferenza in un'applicazione web funzionante con poche righe di codice.

7.2 Esempi di test

Nella demo vengono presentati diversi esempi rappresentativi per ciascuna delle classi del dataset: *glass*, *paper*, *cardboard*, *plastic*, *metal* e *trash*. L'utente può selezionare uno degli esempi predefiniti o caricare un'immagine personalizzata.

Dopo il caricamento, l'immagine viene trasformata e preprocessata come durante la fase di test, e inviata al modello. Il sistema restituisce le tre classi più probabili, con le relative percentuali di confidenza. La previsione viene visualizzata in maniera chiara e intuitiva, rendendo l'esperienza di utilizzo fluida anche per utenti senza conoscenze tecniche.

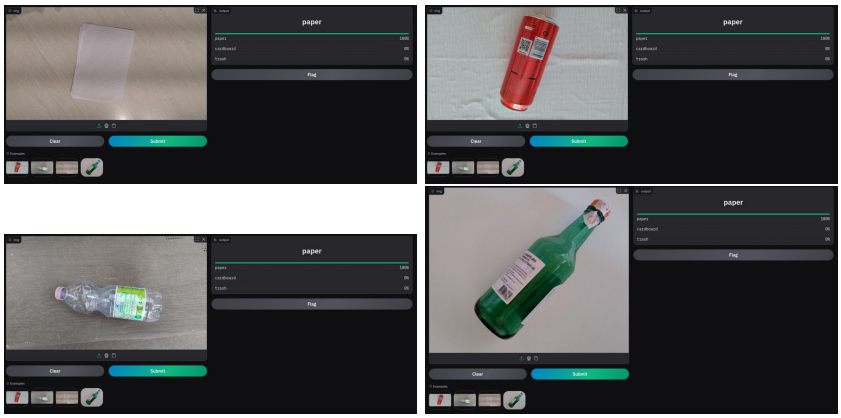


Figure 14: Matrici di confusione sui test per gli esperimenti 1, 2, 3 e 4.

7.3 Come replicare la demo

Il progetto è stato caricato sul profilo Github al seguente link e ha la seguente struttura:

```
project-garbage-classifier/
```

```
README.md
requirements.txt

database/
   database.py

src/
   pre_processing.py
   classifier.py
   main.py

.gitignore
```

Come elencato dal README.md per poter replicare la demo sono sufficienti tre passaggi:

1. Creare e attivare un ambiente virtuale:

   ```
   python3 -m venv .venv && source .venv/bin/activate
   ```

2. Installare tutte le dipendenze necessarie con:

   ```
   pip install -r requirements.txt
   ```

3. Accedere alla cartella della demo ed eseguirla:

   ```
   cd demo
   python3 demo.py
   ```

Lo script `demo.py` carica il modello pre-addestrato, applica le trasformazioni sull'immagine selezionata e ne fornisce la classificazione tramite una semplice interfaccia Gradio. L'interfaccia mostra le tre classi più probabili per ogni immagine, insieme alle percentuali di confidenza associate. Per poter replicare la demo è possibile selezionare una delle immagini già precaricate così da visualizzare immediatamente il comportamento del modello oppure l'utente può caricare un'immagine a sua scelta e una volta cliccato il pulsante "Submit" apparirà il risultato.

8 Codice

Il cuore del progetto è il file *main.ipynb*: esso importa i vari moduli e si occupa di effettuare nel giusto ordine i vari passi che portano al training del modello con determinate caratteristiche e al testing finale. I moduli utilizzati sono:

- dataset/ dataset.py
- src/ classifier.py
- src/ evaluation.py
- src/ prePocessing.py

8.1 dataset

Questo modulo si occupa di gestire tutto ciò che riguarda i dati e la loro divisione nei vari set. Dopo aver estratto i dati da un file ZIP, assegna etichette numeriche alle class e crea un DataFrame contenente i percorsi delle immagini e le relative etichette.
La funzione `split_train_val_test()` divide il dataset in *training*, *validation* e *test set* secondo proporzioni specificate. I tre sottoinsiemi vengono poi salvati come file CSV nella cartella `dataset/CSV`.
Infine, il modulo definisce una classe personalizzata CSVImageDataset, compatibile con PyTorch, che consente di caricare immagini direttamente dai file CSV, applicando eventualmente trasformazioni (*transform*) utili per il preprocessing dei dati durante il training e l'inferenza.

8.2 preprocessing

Questo modulo si occupa della configurazione del preprocessing e della gestione dei dati. Vengono definite due pipeline di trasformazione:

- train_transform, che include operazioni di data augmentation come crop casuale e flip orizzontale per aumentare la varietà dei dati di addestramento;
- test_transform, pensata per la valutazione coerente del modello, con crop centrale.

La funzione data_augumentation() consente di bilanciare il dataset duplicando i campioni di una specifica classe (class_to_aug) un certo numero di volte (times). Questa tecnica è utile nei casi di sbilanciamento tra classi, migliorando la qualità dell'apprendimento del modello: nel nostro caso, è stata utile per la classe *trash*

8.3 Classifier

Questo modulo definisce l'intero processo di addestramento, validazione e test di un modello di classificazione delle immagini. La funzione principale, `train_validate()`, si occupa di:

- inizializzare e personalizzare il modello (*efficientnet_v2_s*);
- configurare l'ottimizzatore e la loss function;
- da la possibilità di inserire early stopping e/o dropout;
- gestire il salvataggio dei checkpoint migliori;
- monitorare metriche e perdite con TensorBoard;
- fornire log in tempo reale dettagliati per il tracciamento dell'addestramento;
- da la possibilità di riprendere il modello da uno stato salvato (*resume_from*)

La funzione `load_model()` consente di ricaricare un modello già addestrato da un checkpoint, rendendolo pronto per la valutazione o l'uso.
Infine, `test_model()` permette di testare il modello caricato su un set di test, calcolando l'accuratezza, un classification report e visualizzando la matrice di confusione, al fine di analizzare le prestazioni per ogni classe.

8.4 Evaluation

Questo modulo si occupa di fornire funzioni che valutano e mostrano attraverso dei grafici le metriche relative al modello passato in input:

- `plot_class_distribution`, mostra tramite un grafico a barre il numero di dati per classe;
- `evaluate`, calcola e stampa le metriche di classificazione principali (accuracy, precision, recall, F1-score) e la matrice di confusione, restituendo anche i valori;
- `plot_confusion_matrix`, visualizza la matrice di confusione;
- `plot_metrics_bar`, rappresenta in un grafico a barre le metriche principali per una comprensione immediata delle prestazioni;
- `plot_loss_acc_per_epoch`, traccia l'andamento della loss e dell'accuratezza nel tempo per training e validation, al fine di analizzare il comportamento dell'addestramento.

9 Considerazioni finali e sviluppi futuri

Il progetto ha dimostrato la fattibilità e l'efficacia di una soluzione automatizzata per la classificazione dei rifiuti basata su tecniche di deep learning. L'utilizzo di un modello pre-addestrato come EfficientNetV2-S ha permesso di ottenere ottimi risultati in termini di accuratezza e generalizzazione, pur disponendo di un dataset relativamente contenuto.

9.1 Punti di forza

Durante lo sviluppo, sono emersi diversi aspetti positivi:

- Il modello ha raggiunto una media di **90% di F1-score**, con buoni risultati su tutte le classi.
- L'adozione del transfer learning ha ridotto sensibilmente i tempi e i costi computazionali.
- La pipeline è modulare e facilmente estendibile a nuove classi di rifiuto.
- L'intero processo può essere replicato su dispositivi a basso costo, rendendolo adatto anche ad applicazioni educative o civiche.

9.2 Criticità riscontrate

Nonostante le buone performance, sono stati osservati alcuni limiti:

- Alcune classi (es. `trash`) presentano ambiguità semantica e visiva.
- La presenza di sfondi non omogenei e luci variabili ha influenzato negativamente alcune predizioni.
- Il dataset non copre tutte le tipologie di rifiuti reali (es. organico, elettronica, tessili).

9.3 Sviluppi futuri

Il progetto può essere esteso e migliorato in diversi modi:

- **Raccolta di un dataset proprietario**, con immagini acquisite in ambienti reali (scuole, uffici, ambienti pubblici).
- **Integrazione con sistemi embedded**, ad esempio Raspberry Pi + telecamera + inferenza in tempo reale.
- **Espansione del numero di classi**, includendo nuove tipologie di rifiuto e sottocategorie.
- **Sperimentazione con altri modelli leggeri**, come MobileNetV3 o Vision Transformers ottimizzati.
- **Applicazione di tecniche explainable AI (XAI)**, per visualizzare le aree dell'immagine su cui il modello basa la classificazione.

9.4 Conclusione

In conclusione, il progetto rappresenta un primo passo verso l'applicazione dell'intelligenza artificiale per la sostenibilità ambientale. Sebbene semplice, il sistema proposto è in grado di svolgere un compito utile con buoni risultati, ed è facilmente scalabile.

Lezioni apprese. Durante il progetto, sono emerse numerose osservazioni chiave:

- lavorare con dati reali e imperfetti è essenziale per testare la robustezza del modello in condizioni realistiche;
- l'adozione di strategie di *data augmentation* e bilanciamento ha migliorato la capacità del modello di generalizzare;
- l'impiego di tecniche di regolarizzazione come *dropout* ed *early stopping* ha contribuito a evitare l'overfitting;
- la valutazione tramite metriche complete (accuracy, F1-score, confusion matrix) è fondamentale per una comprensione approfondita delle prestazioni;
- strumenti come TensorBoard e Gradio hanno reso più efficiente sia il monitoraggio dell'addestramento che la realizzazione della demo.

Sviluppi futuri. Sebbene i risultati ottenuti siano ottimi (fino al 94% di accuratezza), il sistema può essere ulteriormente migliorato:

- espandendo il dataset con immagini raccolte in ambienti reali (es. scuole, uffici, spazi pubblici);
- aggiungendo nuove classi di rifiuto complesse come organico, elettronica o tessili;
- adottando modelli *explainable AI* per aumentare la trasparenza delle decisioni del modello;
- integrando la pipeline in sistemi *embedded*, come una Raspberry Pi con telecamera, per applicazioni in tempo reale.

Impatto e prospettive. Il progetto, pur nella sua semplicità, rappresenta un primo passo verso l'uso dell'intelligenza artificiale per la sostenibilità ambientale. Il sistema è:

- **Scalabile**, grazie a una pipeline modulare e ben documentata;
- **Replicabile**, grazie all'uso di tecnologie open-source e ambienti facilmente accessibili (es. Google Colab);

- **Educativo**, poiché utilizzabile in scuole, workshop o campagne civiche per promuovere la raccolta differenziata.

Con un miglioramento del dataset e una maggiore integrazione hardware-software, questo sistema può evolversi in uno strumento concreto per l'educazione ambientale e la gestione dei rifiuti urbani.