

Relazione dettagliata del progetto CIFAR-10

realizzato da: Alexandru Apostol & Tommaso Foca

1. Obiettivi del Progetto

L'obiettivo principale di questo progetto è sviluppare un sistema di **classificazione delle immagini** utilizzando una rete neurale convoluzionale (CNN) basata su **ResNet50**, per la classificazione delle immagini nel dataset Vehicles.

In particolare, l'intento è di:

- **Preprocessare** correttamente il **dataset** Vehicles.
- **Creare e allenare** un modello basato su **ResNet50**.
- Implementare tecniche come **Early Stopping** per prevenire il **sovra-allenamento** (overfitting) e migliorare la generalizzazione del modello.

2. Dataset e Preprocessamento

Dataset su Kaggle:

<https://www.kaggle.com/datasets/mohamedmaher5/vehicle-classification>

Il dataset utilizzato è Vehicles, composto da 5600 immagini totali in formato JPEG. Per il nostro progetto ne abbiamo utilizzate di meno per non appesantirlo troppo e per dimezzare i tempi esecutivi.

Il dataset è suddiviso in:

- Test, file utilizzato per testare il programma
- Train, file utilizzato per allenare il programma con le immagini modificate e filtrate
- Valid, file in cui si può vedere se le immagini sono state validate correttamente

Sebbene ResNet50 richieda maggiori risorse computazionali e tempi di addestramento più lunghi, la sua maggiore profondità consente di catturare pattern più complessi nei dati.

3. Architettura del Modello

- Nonostante il dataset non sia particolarmente grande, l'adozione di ResNet50 ha garantito un modello più robusto e preciso, in grado di distinguere in modo chiaro tra le categorie di interesse.
- È stata utilizzata una Cross-Entropy Loss come funzione di perdita, poiché il problema di classificazione è multiclasse.
- L'ottimizzazione è gestita dall'algoritmo Adam, che è una versione avanzata di Stochastic Gradient Descent.

3. Strategia di Allenamento

Durante la fase di testing con ResNet50, è stato utilizzato un dataset organizzato in una cartella denominata "test" con le sue sottocartelle contenenti immagini specifiche.

Il modello è stato addestrato utilizzando una cartella "train", fondamentale per il processo di apprendimento. Essa, infatti, contiene le sottocartelle per ciascuna delle categorie menzionate. Queste sottocartelle includono le immagini utilizzate per addestrare il modello, consentendogli di apprendere le caratteristiche distintive di ogni classe.

La cartella "test" viene invece usata dal file Vehicles.py, il quale utilizza le immagini in essa contenute per poterle analizzare e di conseguenza classificarle nelle proprie sottocartelle nella cartella output "valid".

Il modello è stato implementato nel file Model.py, mentre la classificazione delle immagini è stata gestita dal file Vehicles.py,

responsabile dell'elaborazione delle proprie categorie. In particolare, il file contiene le funzioni responsabili dell'applicazione dei filtri alle immagini classificate correttamente. Filtri analoghi sono stati utilizzati per le altre categorie menzionate. Durante ogni epoca, il modello è addestrato sui dati di addestramento e la loss è monitorata per decidere se è necessario salvare un checkpoint. Il modello viene salvato ogni volta che la loss migliora. In caso di peggioramento, viene salvato un checkpoint "non valido", indicandolo come una prestazione inferiore.

Info:

Nel codice, è stata introdotta una funzione denominata `get_random_images` per migliorare il processo di test del modello di classificazione delle immagini. Quando il modello viene eseguito attraverso la funzione `process_images`, le immagini vengono selezionate in modo casuale dalla cartella di test, che contiene le immagini dei veicoli da classificare.

L'obiettivo di questa implementazione è garantire che il modello venga testato in modo casuale ogni volta che il programma viene eseguito, anche se le immagini di test rimangono le stesse. Selezionando le immagini in ordine casuale, il modello viene sottoposto a una varietà di sequenze di test, offrendo una verifica più completa delle sue capacità di classificazione. Questo approccio aiuta a confermare che il modello non stia semplicemente memorizzando l'ordine delle immagini, ma stia effettivamente classificando correttamente le immagini basandosi sulle loro caratteristiche.

5. Config.json e config_schema.json

Abbiamo creato e implementato i due file per la validazione della configurazione: 'config_json' e 'config_schema.json':

config.json: contiene tutti i parametri di configurazione necessari per il training e la validazione dei modelli. Ad esempio, include le directory per i dataset, i nomi dei file per salvare i modelli e i parametri di training.

config_schema.json: definisce lo schema JSON per validare il contenuto di config.json. Questo schema è progettato per garantire che config.json abbia la struttura corretta e i tipi di dati giusti prima di essere utilizzato nel codice.

Per assicurare che il file di configurazione rispetti il formato e i requisiti definiti, abbiamo integrato la libreria jsonschema. Questa libreria consente di validare il file di configurazione rispetto allo schema JSON. In particolare, durante l'esecuzione del programma, jsonschema carica sia config.json che config_schema.json e verifica che il primo aderisca alle specifiche del secondo. Se il file di configurazione non rispetta le regole definite, viene segnalato un errore e l'esecuzione del programma viene interrotta.

L'implementazione e configurazione di questi file ci ha portato diversi vantaggi:

- Separazione della configurazione: Rendendo il codice più pulito e leggibile, poiché i parametri di configurazione sono esterni al codice principale

- Flessibilità e riutilizzabilità: Permette di modificare facilmente i parametri di configurazione o riutilizzare il codice per addestrare e inferire su diversi modelli senza modificare il codice sorgente.
- Scalabilità: Facilita l'espansione futura del progetto senza dover rifattorizzare il codice esistente, poiché le modifiche possono essere gestite attraverso i file di configurazione
- Monitoraggio e Salvataggio dei Modelli

Incorporare la validazione tramite 'jsonschema' contribuisce a garantire l'integrità e la coerenza dei dati di configurazione, riducendo il rischio di errori e migliorando la robustezza complessiva del sistema.

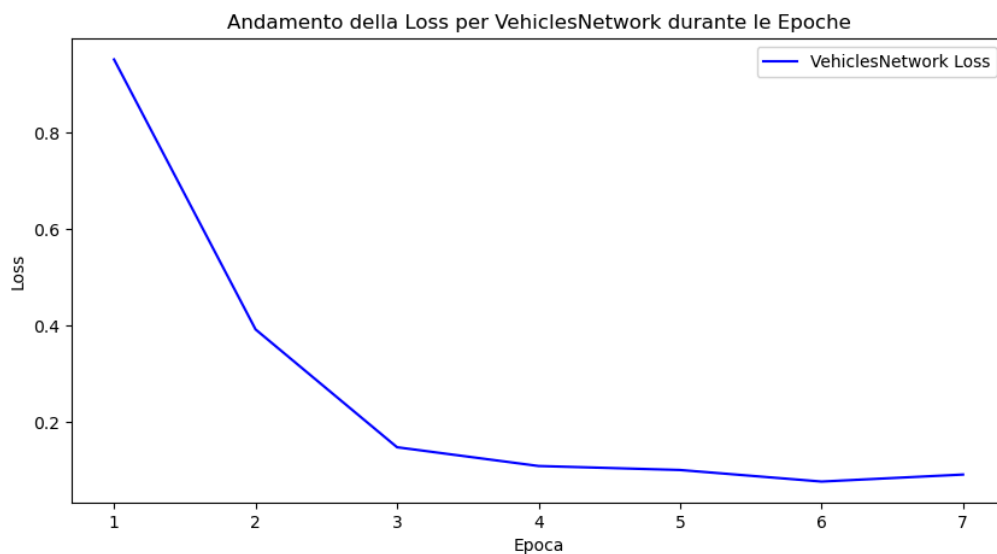
7. Rappresentazione dei dati

Per rappresentare i dati ottenuti dopo l'addestramento dei modelli e quindi dall'esecuzione del file `Vehicles.py` e `Model.py`, abbiamo utilizzato la libreria `matplotlib.pyplot` per effettuare una distribuzione delle probabilità (o confidenze) sulle classi possibili; quindi, graficare la distribuzione di queste probabilità per vedere quanto il modello è sicuro delle sue predizioni.

La libreria utilizzata ci consente di visualizzare i risultati delle predizioni dei modelli in modo chiaro e comprensibile.

7.2 Passaggi eseguiti:

- Preparazione dei dati: Abbiamo creato un DataFrame che contiene i nomi delle immagini, le categorie predette e le rispettive confidenze.
- Creazione del grafico: utilizzando `plt.bar()`, abbiamo creato un grafico a barre in cui l'asse x rappresenta i nomi delle immagini e l'asse y la confidenza della predizione; le barre sono colorate, con bordi neri per migliorare la visibilità e l'`alpha` (trasparenza) impostata a 0.7 per un effetto visivo più gradevole.
- Personalizzazione del grafico: abbiamo impostato etichette per gli assi e un titolo per il grafico; le etichette dell'asse x sono state ruotate di 45 gradi per evitare sovrapposizioni e migliorare la leggibilità.
- Esecuzione di `Model.py`
- Abbiamo agito similmente anche per graficare la funzione di loss presente nel file `Model.py`.
- Invece di usare un grafico a barre, abbiamo ritenuto essere più corretto utilizzare un grafico a linee per rappresentare l'andamento della loss durante le epoche di addestramento dei modelli. Questo tipo di grafico è utile per visualizzare le variazioni e le tendenze di una variabile continua nel corso di un intervallo, come le epoche di addestramento in questo caso. Nello specifico, all'asse delle x viene assegnato il numero di epoche e, di conseguenza, ha un intervallo da 1 a 10, mentre all'asse delle y viene assegnato il valore della loss.



8. Salvataggio del modello migliore:

Nel corso dell'addestramento di reti neurali, è fondamentale salvare il modello che raggiunge le migliori prestazioni, piuttosto che salvare semplicemente l'ultimo stato del modello alla fine dell'addestramento. Per questo motivo, è stata implementata una nuova funzionalità in **Model.py** che permette di monitorare la performance del modello su un set di validazione e salvare il modello solo quando viene raggiunta la miglior prestazione (misurata, ad esempio, dalla loss più bassa o dalla più alta accuratezza).

Questa modifica garantisce che il modello salvato al termine del processo di addestramento sia effettivamente quello più performante, riducendo il rischio di sovradattamento (overfitting) che può verificarsi nelle ultime epoche. Il modello migliore viene salvato con il suo stato ottimale, pronto per essere utilizzato nelle fasi successive del progetto, come ulteriori analisi.

Per implementare questa funzionalità, è stato aggiunto un controllo durante ogni epoca di addestramento che verifica se la loss corrente è inferiore a quella del miglior modello salvato fino a quel momento.

In caso ciò accada, il modello viene salvato con il suo stato attuale, garantendo così la miglior efficacia possibile.

Integrazione di Early Stopping

Durante l'addestramento delle reti neurali, è fondamentale evitare l'overfitting, che può verificarsi quando il modello continua ad addestrarsi anche dopo che le sue prestazioni su un set di validazione iniziano a peggiorare. Per affrontare questo problema, è stata implementata una nuova funzionalità di early stopping all'interno di **Model.py**. Questa tecnica monitora la perdita di validazione durante l'addestramento e interrompe il processo quando non si osservano miglioramenti significativi per un numero specifico di epoche consecutive.

L'integrazione di questa tecnica garantisce che l'addestramento si interrompa al momento ottimale, preservando il miglior modello possibile e riducendo i tempi di addestramento inutili. Ciò è particolarmente utile in scenari in cui il sovradattamento può degradare le prestazioni del modello nelle fasi finali dell'addestramento.


Per implementare questa funzionalità, è stato aggiunto un meccanismo che controlla la perdita di validazione dopo ogni epoca. Se la perdita non migliora rispetto alla miglior perdita osservata entro un certo numero di epoche (definito dal parametro `patience`), l'addestramento viene interrotto anticipatamente. Questo assicura che il modello finale sia il più generalizzabile possibile, riducendo il rischio di overfitting e ottimizzando l'efficacia complessiva del modello per l'inferenza o ulteriori applicazioni.

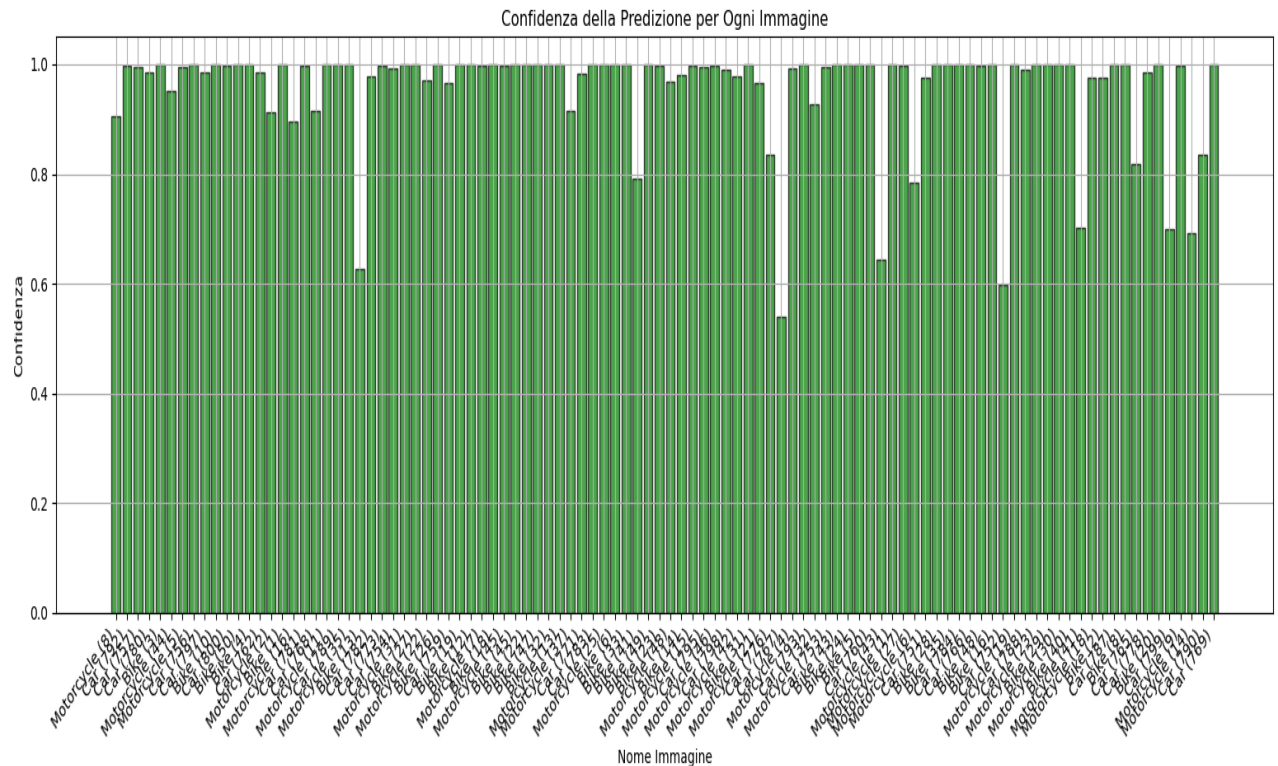
9. Valutazione dei dati ottenuti durante l'esecuzione del progetto

Di seguito osserviamo i risultati ottenuti con una breve spiegazione, ma prima una rapida contestualizzazione: la confidence (confidenza) in un modello di classificazione rappresenta la probabilità stimata che il modello attribuisce a una determinata classe per un dato campione.

Per cominciare, viene eseguito il file Model.py, ottenendo il seguente output):

```
Inizio dell'addestramento del modello vehicles_model.pth...
Epoch 1, Loss: 0.9512101478046842
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.9512
Epoch 2, Loss: 0.39246153831481934
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.3925
Epoch 3, Loss: 0.1485812986890475
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.1486
Epoch 4, Loss: 0.10977022722363472
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.1098
Epoch 5, Loss: 0.10149558757742246
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.1015
Epoch 6, Loss: 0.0775538769480255
Modello migliorato e salvato in 'best_vehicles_model.pth' con Loss: 0.0776
█
```

gs  CyberCoder Improve Code Search Error Share Code Link Ln 8, Col 44



Difficoltà riscontrate

Ci sono state diverse difficoltà nella realizzazione di questo progetto, la principale delle quali è stata sicuramente la necessità di alternarsi nel lavoro e organizzare delle sessioni di briefing, riuscendo a conciliare il lavoro con lo studio di entrambi. Nonostante ciò, siamo riusciti a completare il progetto utilizzando le risorse esplorate durante le lezioni e le conoscenze acquisite lungo il percorso. Una difficoltà iniziale è stata quella di approcciarsi correttamente a questo progetto, trattandosi di una materia del tutto nuova per noi, ma grazie all'impegno e alla disciplina siamo riusciti a portarlo a termine.

