Riconoscimento efficiente di immagini di animali tramite transfer learning e quantizzazione

ML project a.y. 2024/2025

Stefano Belli, matricola 0350116

Università degli Studi di Roma "Tor Vergata"

Agenda

Il problema da affrontare

Si richiede di progettare un classificatore di immagini di animali tramite modelli preaddestrati, sfruttando il **transfer learning** e la **quantizzazione**.

Per questo progetto, è importante tenere conto sia dell'**accuratezza** del modello che dei **costi computazionali**.

Verranno confrontati due modelli con e senza quantizzazione tenendo conto delle seguenti metriche:

- Loss e accuracy del modello
- Tempo di inferenza del modello
- Grandezza del modello ottenuto

Transfer learning

Addestrare modelli di deep learning in modo efficiente non è affatto semplice:

1. Hardware potente e costoso richiesto
2. Tempi di addestramento del modello elevati
3. Costi per l'energia eccessivi
4. Mancanza di dati per l'addestramento
5. Progettare una rete da zero

Applicazione

E' possibile sfruttare modelli preaddestrati complessi, già testati e perfettamente funzionanti e "trasferirli" al nostro problema congelandone i pesi (*"trained weights"*), lasciando "addestrabili" solo i pesi di una rete neurale densa che è il classificatore rimpiazzato dal nostro.

DNNs e dispositivi embedded

Pensiamo all'IoT e alla diffusione capillare di dispositivi embedded, ad esempio una telecamera: e se volessimo integrare una rete convoluzionale nel device stesso?

- Preserveremmo la privacy dell'utente
- Niente problemi di latenze elevate o legate al trasferimento dati
- Se la telecamera viene disconnessa da internet, la rete può continuare a svolgere il suo compito, rendendo il dispositivo più affidabile

Il problema nell'eseguire le reti neurali su tali dispositivi è ovvia: la poca potenza a disposizione impatta sui tempi di **inferenza** del modello e sulla **memorizzazione** (sia in memoria primaria che secondaria) dei pesi del modello, oltre al fatto che il dispositivo potrebbe non avere capacità di calcolo in virgola mobile (es. non ha una FPU o ISA che supporti operazioni floating point).

Quantizzazione

Una tecnica che consente di ridurre la dimensione dei parametri di un modello:

- Meno **storage** richiesto per mantenere i parametri della rete
- Reappresentare un parametro da `float` $\rightarrow$ `int8_t` significa niente operazioni floating point e quindi **tempi** di inferenza minori

La tecnica non impatta significativamente sull'accuratezza del modello originale: l'alta precisione di un `float` o `double` probabilmente non è necessaria per far si che il modello svolga bene il suo lavoro.

Quantizzazione post-training

In particolare, nel progetto viene utilizzato TFLite/LiteRT e quantizzazione post-training che permette di definire un modello Keras, addestrarlo normalmente, e solo dopo quantizzarlo.

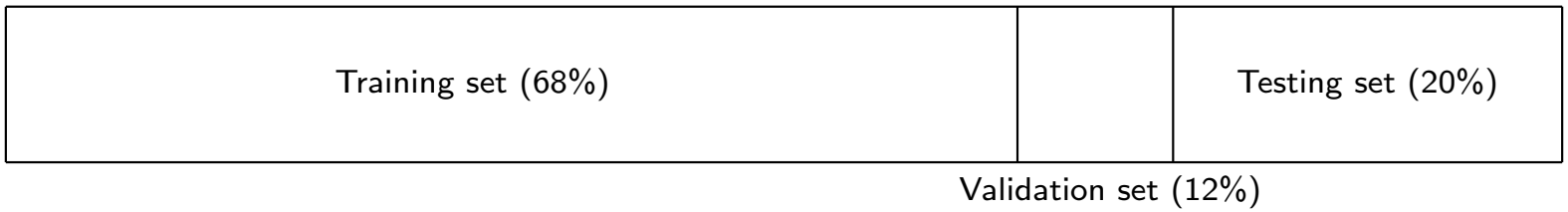
Dopo aver convertito il modello, è possibile applicare 3 livelli di quantizzazione (incrementale)

	Param. fissati	**Variabili**	**Tensori di I/O**
Dynamic range	✓	✗	✗
Float fallback	✓	✓	✗
Integer-only	✓	✓	✓

Tabella: ✓ indica che avviene la quantizzazione, ✗ indica che non avviene

Split del dataset

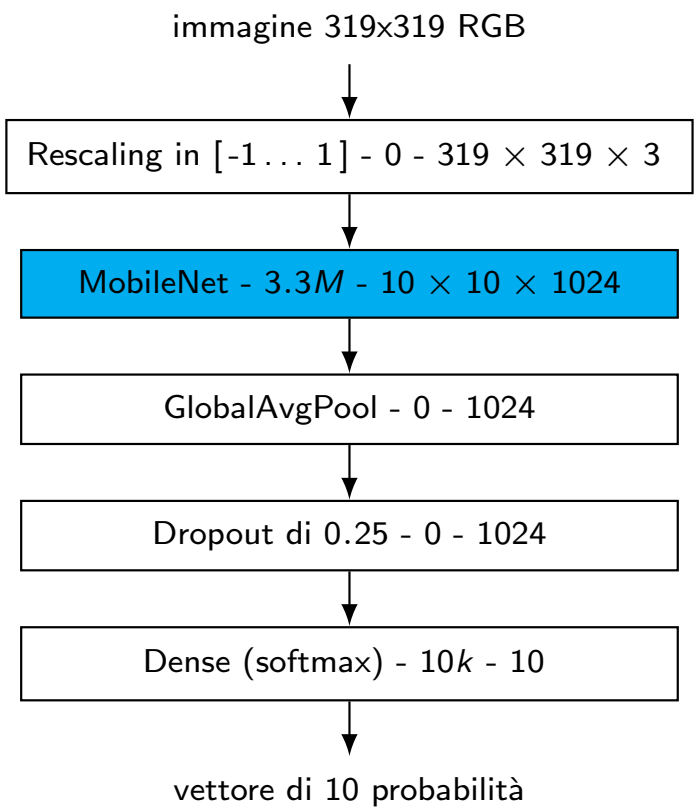
Il dataset fornito consiste in immagini di animali con etichette associate (10 classi), pronto per essere letto e splittato da `keras.utils.image_dataset_from_directory`

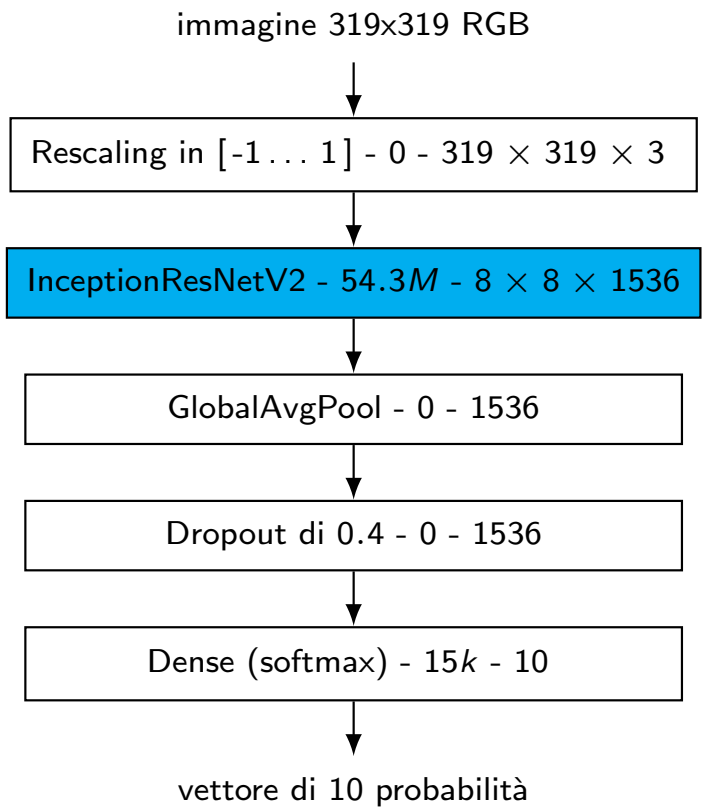


- Il **training set** viene utilizzato per l'addestramento dei modelli
- Il **validation set** viene usato ogni 3 epoche di addestramento e mostrare esempi di predizione
- Il **testing set** viene usato per effettuare le misurazioni finali

Architettura dei modelli analizzati

I due modelli pretrained scelti sono `MobileNet` e `InceptionResNetV2`, facilmente istanziati con i trained weights `imagenet` grazie a `keras.applications`





Training

Riusabilità del codice

Evaluation dei modelli

Risultati ottenuti

Conclusioni

Grazie per l'attenzione!