Alessandro Rodeghero - Code Challenge (AI Developer Junior)

Classificazione - Gatti vs Cani

ho sviluppato il test sia come notebook che come progetto python.

repo GitHub: https://github.com/gitaler/cat_dog_classification
dataset utilizzato: https://www.microsoft.com/en-us/download/details.aspx?id=54765

Pipeline Generale

1. download dataset
2. unzip dataset nella cartella 'dataset'
3. raccolta informazioni sulle immagini (nomi file, risoluzione, file corrotti)
4. calcolo distribuzione delle classi
5. calcolo risoluzione media
6. rimozione file corrotti
7. creazione cartelle train e test set
8. split del dataset in train e test set (85% train, 15% test)
9. definizione dei generatori di dati (normalizzazione, trasformazioni applicate alle immagini, validation set, batch size ecc.)
10. creazione modello
11. definizione Early Stopping (per evitare overfitting)
12. addestramento
13. valutazione curve di addestramento
14. calcolo metriche di valutazione (accuracy, precision ecc.)

Python Notebooks

- `notebooks/proto.ipynb` : notebook usato per l'esplorazione del dataset e la prototipizzazione.

- `notebooks/final.ipynb` : progetto completo nella forma di notebook python. Eseguire tutte le celle in ordine (provvede anche al download del dataset).

L'addestramento del modello l'ho fatto con ***final.ipynb***. Non avendo a disposizione una GPU dedicata, mi sono affidato alle GPU T4 che mette a disposizione Google con il servizio Colaboratory.

Python Project

Per completezza, ho sviluppato il test anche nella forma di progetto python. Eseguire `main.py` per lanciare l'intera pipeline (provvede anche al download del dataset).

Risultati

1. Primo test:

Nella prima prova ho voluto capire il comportamento dell'addestramento e valutare le prestazioni iniziali raggiungibili.
Vista la varietà nelle risoluzioni delle immagini, queste vengono ridimensionate alla risoluzione media di 381 x 351 pixel.
25 epoche totali, batch size di 64 samples.

Nel generatore (ImageDataGenerator) di train e validation set ho specificato delle trasformazioni randomiche da applicare alle immagini:

```
        rotation_range=15,
        rescale=1. / 255,      # normalizzazione immagini
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        width_shift_range=0.1,
        height_shift_range=0.1,
        validation_split=0.15
```

In questo modo il modello incontra campioni più variabili durante il training, migliorando la generalizzazione.

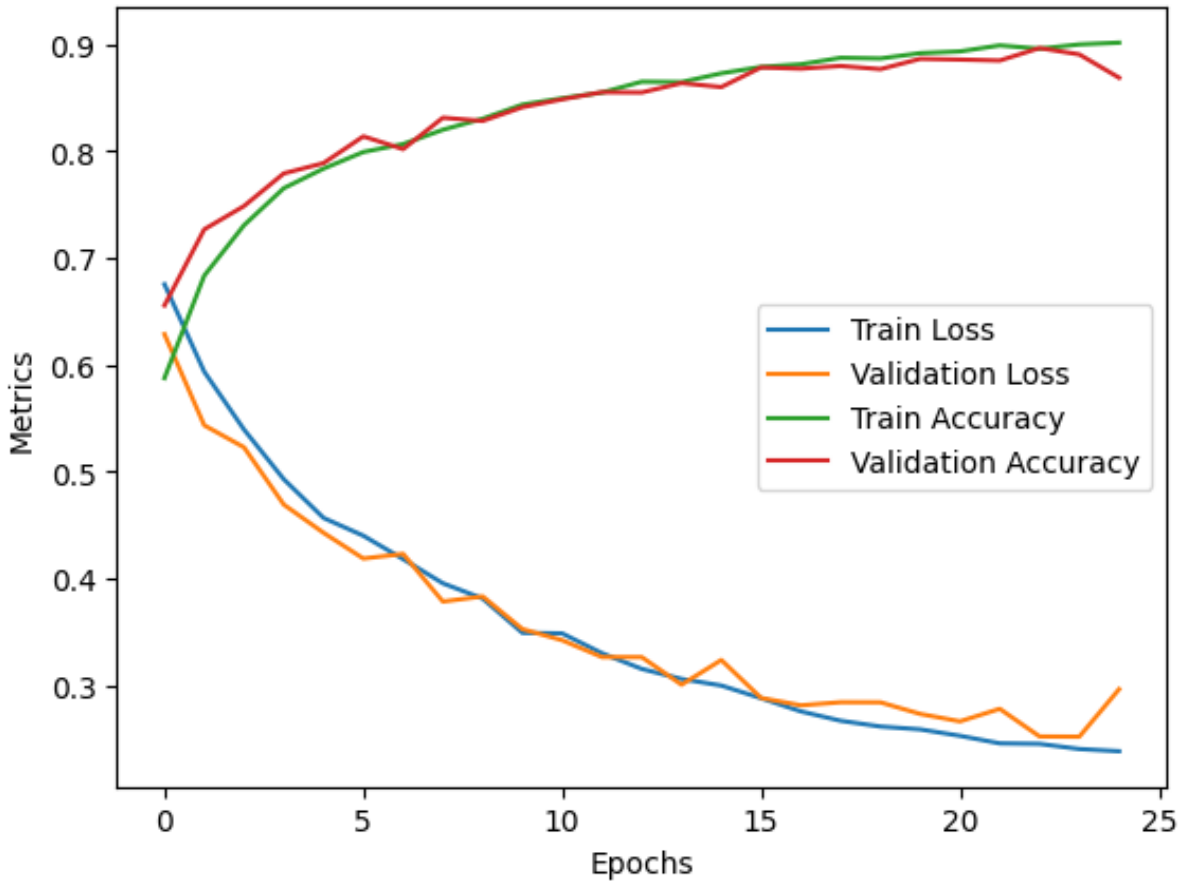
Architettura Modello CNN: 3 layer convolutivi seguiti da un layer fully-connected.

```
input_shape=(381,351,3)
Conv2D(filters=32, kernel_size=(3, 3), activation='relu')
MaxPooling2D((2, 2))

Conv2D(64, (3, 3), activation='relu'))
MaxPooling2D((2, 2))

Conv2D(128, (3, 3), activation='relu')
MaxPooling2D((2, 2))

Flatten()
Dense(512, activation='relu')
Dropout(0.5)
```

```
Dense(1, activation='sigmoid')

optimizer='adam'
loss='binary_crossentropy'
```

`Total params: 123 957 313 (472.86 MB)`

Test Accuracy: `0.9006`



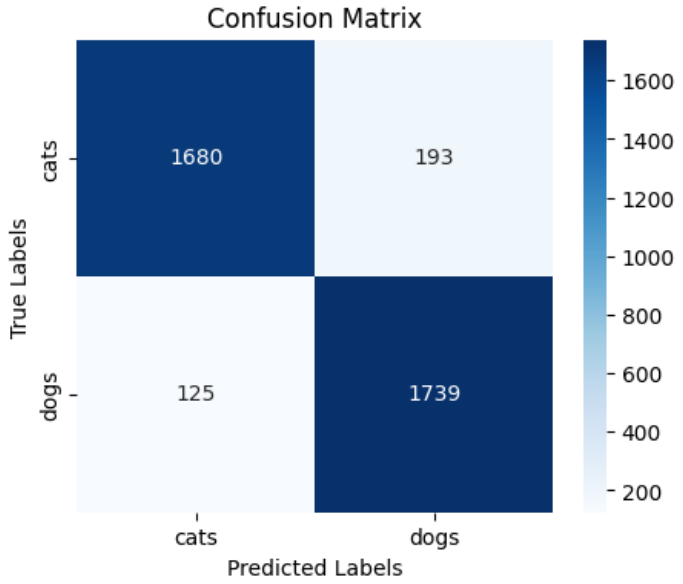
2. Secondo Test - ottimizzazione:

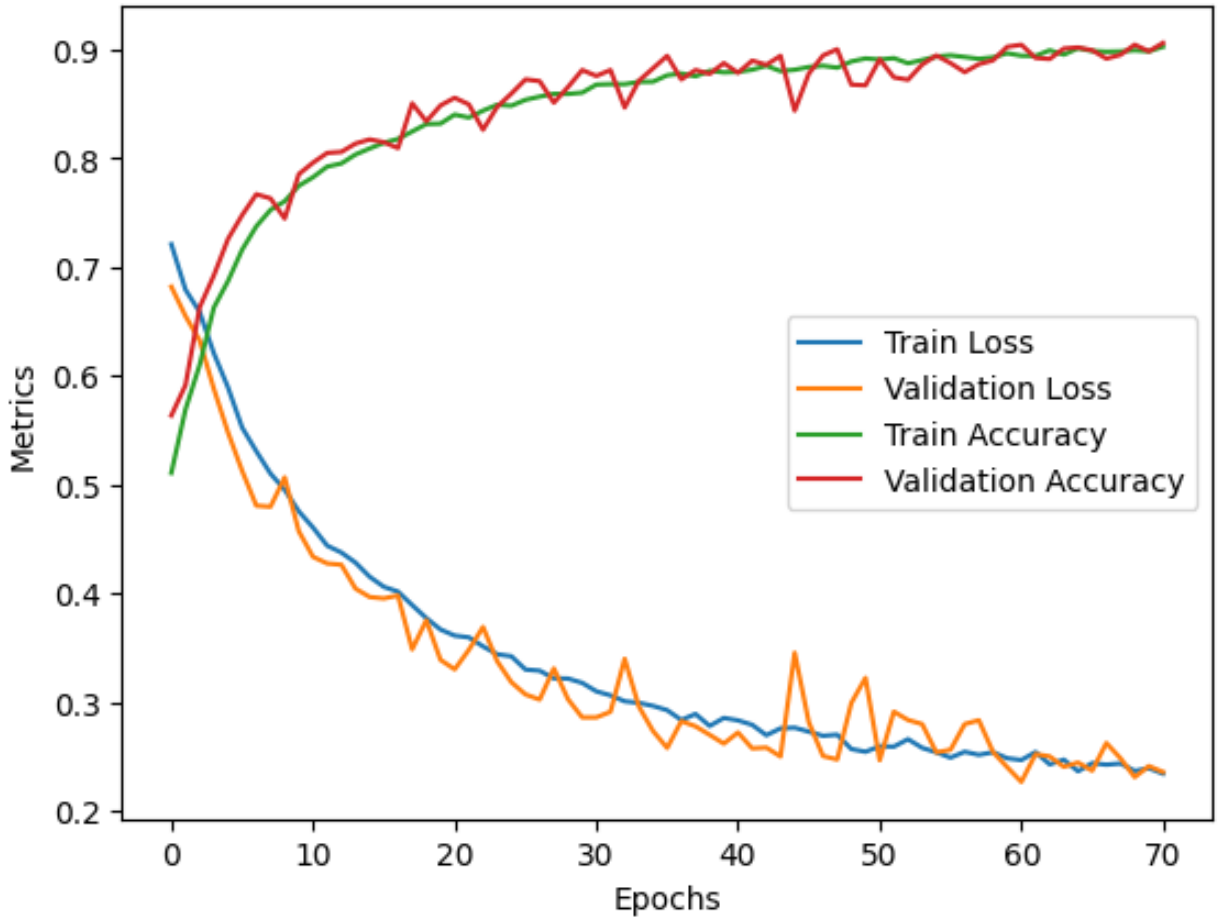
migliorie apportate:

- limite alzato a 75 epoche
- aggiunta di Dropout(0.25) anche nei layer di convoluzione.
- aggiunta di Early Stopping in controllo sulla validation loss (`patient = 10`).
- Vista la lentezza dell'addestramento precedente (2.5 ore per 25 epoche), ho ridimensionato le immagini ad una risoluzione di (128 x 128). Questo porta ad un addestramento più veloce e ad un netto calo dei parametri totali del modello, circa un decimo del modello precedente. (`Total params: 12 939 329 (49.36 MB)`)

L'addestramento si è fermato all'epoca 71 (2.5 ore) per mancato calo della validation loss nelle precedenti 10 epoche.

Test Accuracy: 0.9149
Test Precision: 0.9001
Test Recall: 0.9329
Test F1-score: 0.9162





Riproducibilità dei risultati:

Allego nella cartella results:

- curve di loss (`history.csv`)
- l'istanza di modello prodotto (`best_classifier.h5`)
- nomi dei file presenti nel test set durante la seconda prova (file `.pickle`)

Per verificare la veridicità dei risultati: in `main.py` commentare `main()` e decommentare l'ultima riga (`check_performance.py`).

Verranno letti i nomi delle immagini utilizzate nel test set per ricostruire lo stesso data_generator della seconda prova.
Viene anche caricato `best_classifier.h5` come modello.
Vengono quindi calcolate le metriche richieste dal task.