Team L.A.I.E

Road Signs Classification

Referente: Fabio Narducci Tag: Apprendimento

TEAM

Link Repository GitHub: https://github.com/EDesimone12/Road_Signs_Classification.git

Eugenio De Simone

Andrea Squillante

Paolo Dello Buono

Lucia Gaeta

TECNOLOGIE UTILIZZATE:

-TensorFlow

-Keras

-Google Colab

INDICE DOCUMENTAZIONE

1. Panoramica del progetto

Perché?

In un mondo in cui l'intelligenza artificiale fa da padrone, per muoversi per strada in sicurezza e in modo intelligente, una delle attività necessarie è proprio quella di fornire un servizio che permetta di classificare i segnalatori stradali che ci circondano. Per questo motivo la nostra scelta è ricaduta sulla classificazione in particolare delle segnalazioni principali che ritroviamo tutti i giorni nelle nostre città, quali limiti di velocità, stop, semafori e segnaletica pedonale.

Cosa fa?

Il progetto consiste in un algoritmo di classificazione di immagini di cartelli stradali e semafori. Fornendo un'immagine è possibile capire l'appartenenza di quest'ultima alle classi presenti all'interno del nostro dataset.

2. PEAS

PERFORMANCE

Immagini correttamente classificate/immagini totali.

ENVIRONMENT

Dataset (apprendimento).
Collezione di immagini (applicazione).

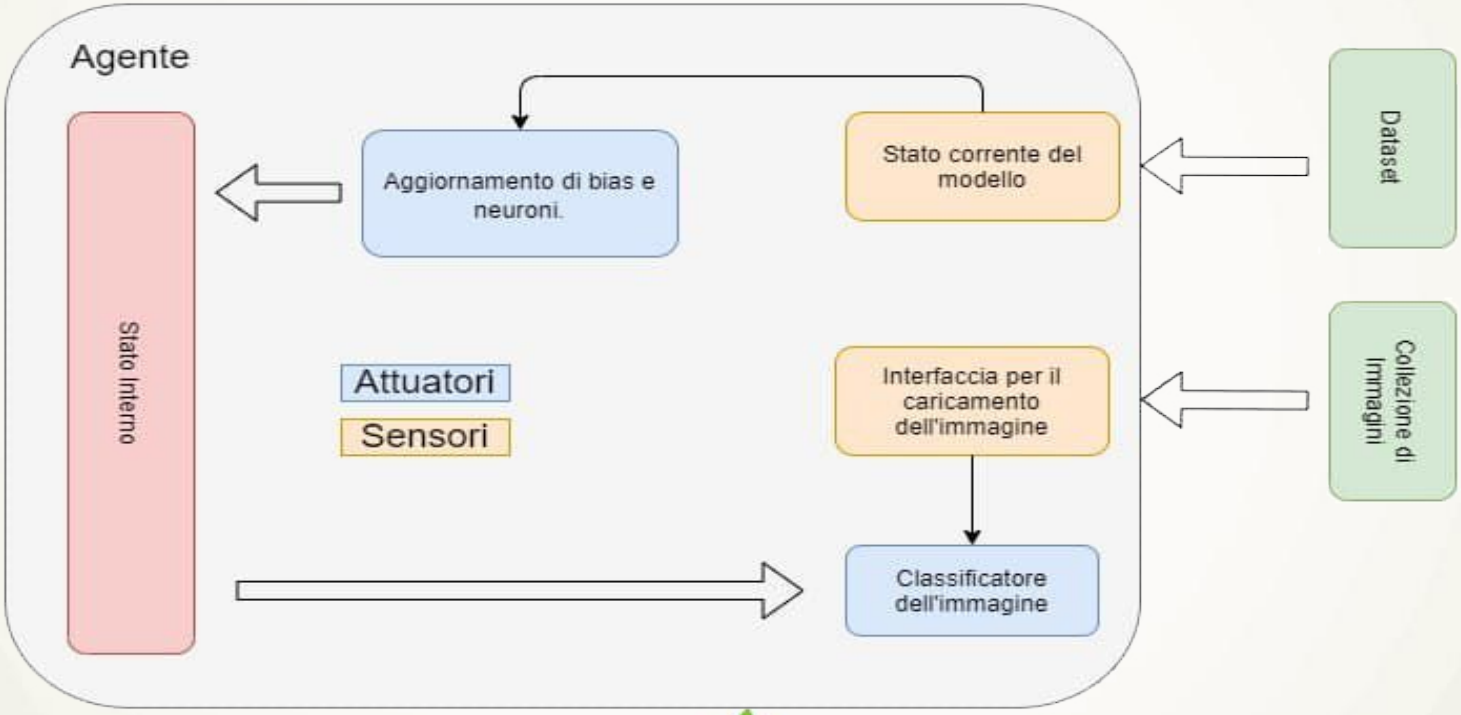
ACTUATORS

Aggiornamento di bias e neuroni.
Classificatore dell'immagine.

SENSORS

Stato corrente del modello.
Interfaccia per il caricamento dell'immagine.

3. Agente


Agente
Stato Interno
Aggiornamento di bias e neuroni.
Stato corrente del modello
Dataset
Attuatori
Sensori
Interfaccia per il caricamento dell'immagine
Collezione di immagini
Classificatore dell'immagine

3. ALGORITMO

La nostra scelta.

Abbiamo deciso di usare una rete neurale convoluzionale perché è più adatta alla classificazione di immagini.

Perché più adatta?

Le CNN sono progettate per riconoscere dei pattern visivi in modo diretto e non richiedono molto preprocessing o comunque ne richiedono una quantità molto limitata. Lo scopo dello strato di Convoluzione è quello di estrarre le features: sono le caratteristiche significative delle immagini, usate per poi calcolare i match tra i punti caratteristici in fase di apprendimento. Si cercano di individuare dei pattern, come ad esempio curve, angoli, circonferenze o quadrati raffigurati in un'immagine con elevata precisione.

Definizione dei layers :

```
#Layers definition

model = Sequential()
model.add(Conv2D(128, kernel_size=(3 ,3), activation='relu', input_shape=X_train.shape[1:]))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(4, activation='softmax'))

#Compilation of the model
#categorical_crossentropy(multiclass classification problems)
#Optimizer Adaptive Moment lr=0.001
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

```
(210, 224, 224, 3) (90, 224, 224, 3) (210, 4) (90, 4)
```

```
history = model.fit(X_train, y_train, batch_size=64, epochs=50, validation_data=(X_test, y_test))
```

Definizione dei layers :

Conv2D: Conv2D: Il layer di convoluzione che si occupa di estrarre le features ed attraverso il kernel(la matrice) con cui viene convoluto l'input per ottenere una Feature Map.

MaxPool2D: si occupa di calcolare il valore massimo o comunque il più grande per ciascuna feature map. Quindi crea una rappresentazione che comprenda i valori più presenti.

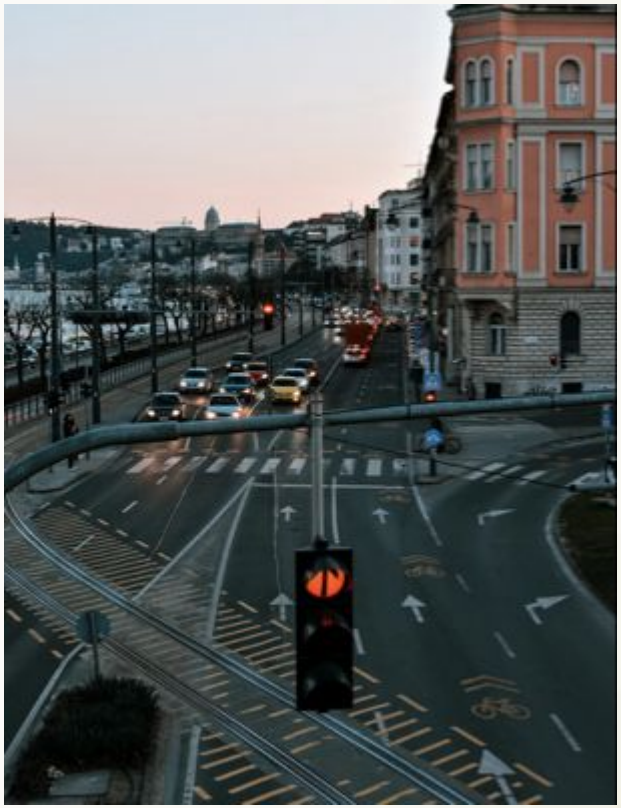
Flattern: è il layer che si occupa di rimuovere tutte le dimensioni dopo i layer di convolucione (Conv2D e MaxPool2D) tranne una.

Dropout: dei neuroni selezionati in maniera randomica vengono ignorati durante l'allenamento.

Dense: è un layer composto da n neuroni, in cui gli input vengono pesati e assieme al bias vengono trasferiti attraverso la funzione di attivazione all'output.

3.1 DATASET

Nel nostro dataset sono presenti 876 immagini.
Comprende 4 tipi di immagini: semafori, segnali di limiti di velocità, stop e segnaletica per l'attraversamento pedonale.
Il dataset presenta una maggiore presenza di immagini di segnali di limiti di velocità quindi abbiamo ridotto il numero delle immagini di questo tipo.
Questa soluzione è stata adottata per ridurre l'**overfitting**.
Ad ogni immagine è associato un file xml.



3.1 DATASET

```
<annotation>
    <folder> images </folder>
    <filename> road2.png </filename>
    <size>
        <width> 306 </width>
        <height> 400 </height>
        <depth> 3 </depth>
    </size>
    <segmented> 0 </segmented>
    <object>
        <name> trafficlight </name>
        <pose> Unspecified </pose>
        <truncated> 0 </truncated>
        <occluded> 0 </occluded>
        <difficult> 0 </difficult>
        <bndbox>
            <xmin> 144 </xmin>
            <ymin> 270 </ymin>
            <xmax> 174 </xmax>
            <ymax> 352 </ymax>
        </bndbox>
    </object>
</annotation>
```

Il tag <name> corrisponde alla categoria dell'immagine. Nel nostro dataset ci sono 4 categorie: trafficlight, stop, speedlimit e crosswalk.

Corrispondono alle coordinate in cui si trova il segnale all'interno dell'immagine.

4. DEMO DEL PROGETTO

Abbiamo realizzato una demo del progetto che permette di caricare un'immagine, la rete la classificherà e fornirà in output la classe corrispondente.
Abbiamo utilizzato come strumento per la creazione della demo Google Colab.
Di seguito il link alla cartella condivisa contenente i file necessari per l'esecuzione della demo ed un notebook.
All'interno del notebook vi saranno le istruzioni da eseguire per eseguire la demo.

Link Demo: https://drive.google.com/drive/folders/1QaQwaleBoeAiYNQp1Oby6fXdrhDrbhWK?usp=sharing

END!