Esecuzione di modelli di rilevamento di oggetti TensorFlow Lite in Python

Le cose buone arrivano nei pacchetti lite (TF)!

Harshit Dwivedi
16 apr 2020 · 8 min di lettura

Foto di Nik Shuliahin su Unsplash

Questo blog è il sesto blog della serie e un seguito al mio precedente post sul blog sull'esecuzione dei modelli di classificazione delle immagini TensorFlow Lite in Python. Se non hai letto quel post, puoi leggerlo qui:

Esecuzione di modelli di classificazione delle immagini Tensorflow Lite in Python

Le cose buone arrivano nei pacchetti lite (TF)!

heartbeat.fritz.ai

Serie Pit Stop

- **Addestramento di un modello di classificazione delle immagini TensorFlow Lite utilizzando AutoML Vision Edge**
- **Creazione di un modello di rilevamento di oggetti TensorFlow Lite utilizzando Google Cloud AutoML**
- **Utilizzo dei modelli di classificazione delle immagini di Google Cloud AutoML Edge in Python**
- **Utilizzo dei modelli di rilevamento di oggetti Edge di Google Cloud AutoML in Python**
- **Esecuzione di modelli di classificazione delle immagini TensorFlow Lite in Python**
- Esecuzione di modelli di rilevamento di oggetti TensorFlow Lite in Python (sei qui)
- **Ottimizzazione delle prestazioni dei modelli TensorFlow per il bordo**

Questo post del blog presuppone che tu abbia già un modello TFLite addestrato a portata di mano. Se non lo fai o hai bisogno di crearne uno, puoi dare un'occhiata al post sul blog che ho scritto qui:

Utilizzo dei modelli di rilevamento di oggetti Edge di Google Cloud AutoML in Python

Lavorare con modelli pronti per il bordo in un ambiente Python

heartbeat.fritz.ai

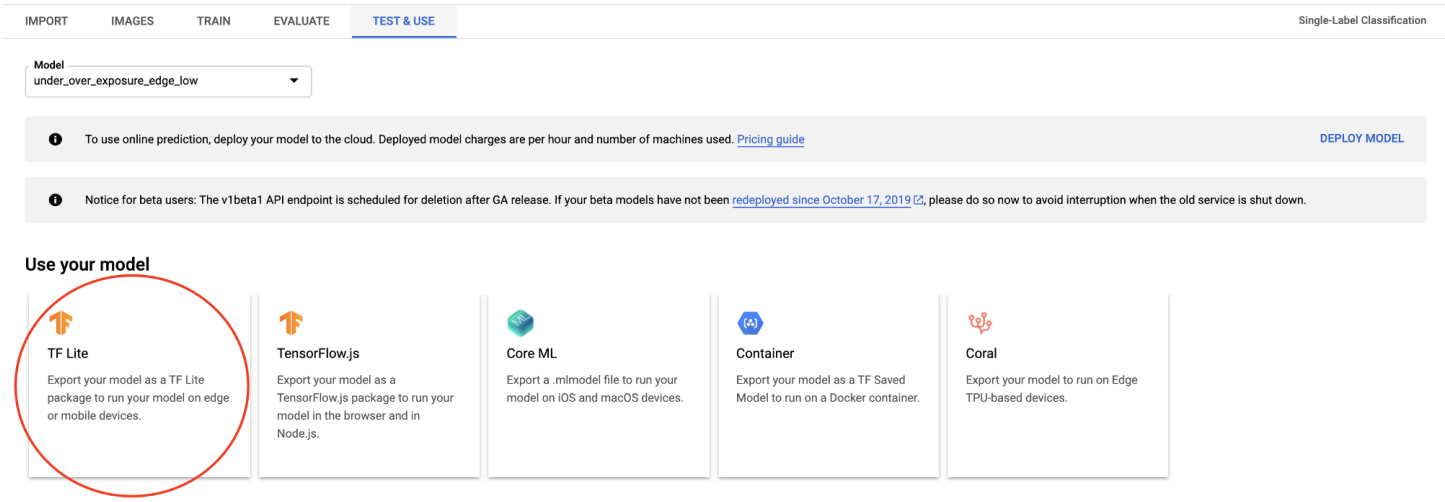
Contrariamente ai modelli di classificazione delle immagini che classificano un'immagine di input in una o più categorie diverse, i modelli di rilevamento degli

oggetti sono progettati per identificare gli oggetti target e fornire un riquadro di delimitazione attorno ad essi (per tracciarne la posizione).

Con quel contesto stabilito, passiamo a come implementare questi modelli in un'impostazione Python.

Passaggio 1: download del modello TensorFlow Lite

Supponendo che tu abbia addestrato il tuo modello TensorFlow con Google Cloud, puoi scaricare il modello da Vision Dashboard, come mostrato nello screenshot qui:



Una volta scaricato, siamo pronti per configurare il nostro ambiente e procedere con i passaggi successivi.

> Incorporare modelli di machine learning nelle app mobili può aiutarti a scalare riducendo i costi. Iscriviti alla newsletter di Fritz AI per ulteriori informazioni su questo e altri modi in cui il ML mobile può portare benefici alla tua azienda.

Passaggio 2: installazione delle dipendenze richieste

Prima di andare avanti e scrivere qualsiasi codice, è importante che tutte le dipendenze richieste siano installate sulla nostra macchina di sviluppo.

Per l'esempio corrente, queste sono le dipendenze di cui avremo bisogno:

```
tensorflow == 1.13.1
pathlib
opencv-python
```

Possiamo usare pip per installare queste dipendenze con il seguente comando:

```
pip install nome_dipendenza
```

> *Nota: sebbene non sia obbligatorio, si consiglia vivamente di utilizzare sempre un ambiente virtuale per testare nuovi progetti. Puoi leggere ulteriori informazioni su come configurarne e attivarne uno nel link qui:*

Creazione di ambienti virtuali Python con Conda: perché e come

Un modo più semplice per installare pacchetti e dipendenze

heartbeat.fritz.ai

Passaggio 3: caricamento del modello e studio dei suoi input e output

Ora che abbiamo il modello e il nostro ambiente di sviluppo pronti, il passaggio successivo è creare uno snippet Python che ci consenta di caricare questo modello ed eseguire inferenze con esso.

Ecco come potrebbe apparire uno snippet di questo tipo:

```
1  importa  numpy  come  np
2  importa  tensorflow  come  tf
3
4  # Carica il modello TFLite e alloca i tensori.
5  interprete  =  tf.contrib.lite.Interprete ( model_path  =  "object_detection.tflite" )
```

```
interprete = tf.contrib.lite.Interpreter(model_path="object_detection.tflite")

# Ottieni tensori di input e output.
input_details = interprete.get_input_details()
output_details = interprete.get_output_details()

interprete.allocate_tensors()

# dettagli di input
print(input_details)
# dettagli di output
stampa(output_details)
```

Qui, prima carichiamo il modello scaricato e poi otteniamo i tensori di input e output dal modello caricato.

Successivamente, stampiamo i tensori di input e output ottenuti in precedenza.
Se esegui il codice, questo è l'aspetto che potrebbe avere l'output:

```
[{'name': 'normalized_input_image_tensor', 'index': 596, 'shape':
array ([1, 512, 512, 3], dtype = int32), 'dtype': <class
'numpy.uint8'>, 'quantizzazione': (0.0078125, 128)}]

[{'name': 'TFLite_Detection_PostProcess', 'index': 512, 'shape':
array ([], dtype = int32), 'dtype': <class 'numpy.float32'>,
'quantization': (0.0, 0)}, {'name': 'TFLite_Detection_PostProcess:
1', 'index': 513, 'shape': array ([], dtype = int32), 'dtype':
<class 'numpy.float32'>, 'quantization ': (0.0, 0)}, {' name ':'
TFLite_Detection_PostProcess: 2 ',' index ': 514,' shape ': array
([], dtype = int32),' dtype ': <class' numpy.float32 '>,'
quantization ': (0.0, 0)}, {' name ':' TFLite_Detection_PostProcess:
3 ',' index ': 515,' shape ': array ([], dtype = int32),' dtype ': <
class 'numpy.float32'>, 'quantizzazione': (0.0, 0)}]
```

A differenza dell'output del modello di classificazione, questo output sembra un po 'troppo da elaborare! Ma esaminiamolo comunque.

Guardando il tensore di input, vediamo che ha una singola voce che accetta un'immagine RGB di dimensioni `512 x 512` come input in `index 596` .

Al contrario, il tensore di output ha 4 voci, il che significa che a differenza del caso precedente in cui abbiamo ottenuto un array a elemento singolo, qui abbiamo 4 elementi nell'array di output.

I riquadri di delimitazione per l'oggetto di cui abbiamo bisogno, insieme ai loro punteggi di confidenza, saranno in due di questi 4 elementi. In genere, gli elementi di output sono ordinati in base all'array di rettangoli seguito dall'array di punteggi per questi rettangoli.

Dopo aver utilizzato alcuni tentativi ed errori, ho identificato che l'elemento denominato `TFLite_Detection_PostProcess` contiene i miei rettangoli e l'elemento denominato `TFLite_Detection_PostProcess:2` contiene i punteggi di questi rettangoli.

L'elemento denominato `TFLite_Detection_PostProcess:3` contiene il numero totale di elementi rilevati e l'elemento `TFLite_Detection_PostProcess:1` contiene le classi per gli elementi rilevati.

Nel nostro caso attuale, la stampa dell'output di `TFLite_Detection_PostProcess:1` dovrebbe stampare un array di zeri. Tuttavia, se hai addestrato un rilevamento di oggetti a rilevare più oggetti; questo elemento potrebbe avere output diversi per te.

Ad esempio, ecco un output di esempio di questo nodo per un modello di rilevamento oggetti addestrato per rilevare 2 oggetti:

```
[
[0. 0. 0. 1 . 1. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
]
```

Qui, se un particolare indice ha il valore 0; quindi la casella e il punteggio a quel particolare indice appartengono al primo oggetto e se ha valore 1; quindi la casella e il punteggio a quell'indice appartengono al secondo oggetto.
Questi valori potrebbero aumentare se hai addestrato il tuo modello a rilevare più oggetti.

Nel passaggio successivo, passeremo un'immagine al modello e vedremo l'output di questi output in azione.

Non dovresti essere un esperto di machine learning

per sbloccarne il potenziale. Lascia questa esperienza a noi. Crea facilmente app mobili che vedono, ascoltano, percepiscono e pensano con Fritz AI.

Passaggio 4: leggere un'immagine e passarla al modello TFLite

Successivamente, useremo Pathlib per scorrere una cartella contenente immagini su cui eseguiremo l'inferenza. Quindi leggeremo ogni immagine con OpenCV, la ridimensioneremo a 512x512 e poi la passeremo al nostro modello.

Una volta terminato, stamperemo il nome del file e il numero di output (0 e 2) per quel file per vedere cosa significa:

```
importa tensorflow come tf
importa numpy come np
importa cv2
import pathlib

interprete = tf . contrib . lite . Interprete ( model_path = "object_detection.tflite" )

input_details = interprete . get_input_details ()
output_details = interprete . get_output_details ()

print ( input_details )
stampa ( output_details )

interprete . allocate_tensors ()

per il file in pathlib . Percorso ( "immagini" ). iterdir ():

    img = cv2 . imread ( r "{}" . format ( file . resolution ()))
    new_img = cv2 . ridimensiona ( img , ( 512 , 512 ))

    interprete . set_tensor ( input_details [ 0 ] [ 'index' ], [ new_img ])

    interprete . invocare ()
    rects = interprete . get_tensor (
        output_details [ 0 ] [ 'index' ])
```

```
26      punteggi  =  interprete . get_tensor (
27          output_details [ 2 ] [ 'index' ])
28
29      print ( "For file {}" . format ( file . stem ))
30      print ( "I rettangoli sono: {}" . format ( rects ))
```

```
Per il file 1DSCF1649 (2) -2020-01-25T15: 16: 25.462Z I
rettangoli sono: [[
  [-0.00325413 0.47174522 0.1879216 0.61565363]
  [-0.00632208 0.24867406 0.17483354 0.3669869]
  [0.00640314 0.32291842 0.1782126432 0.00536
  [0.0091842]
  0.00536 [0.0091842 ] 0.00536 [0.0091842 ] 0.00536 0.28408495
0.18565208 0.3567101]
  [0.00839117 0.22974649 0.15364154 0.33908436]
  [0.98135734 0.90441823 0.99792504 1.0007949]
  [0.9839504 0.8700926 0.9981244 0.99884427]
  [0.00121695 0.19094317 0.4290639 0.5823904]
  [0.460519 0.37304354 0.57724005 0.41105157]
  [0.9853649 0.93691933 0.99602485 0.9942064]
  [-0,00130505 0,24837694 0,02534466 0,34964582]
  [0.00229905 0.45923734 0.1424832 0.5808631]
  [0.5341829 0.54348207 0.64352083 0.57388854]
  [0.00227114 0.24248336 0.19199735 0.31510854]
  [0.98863435 0.28132698 0.9990773 0.40489325]
  [0.43845406 0.36674508 0.5567669 0.41571096]
  [0.9773656 0.8679831 0.9965315 1.0032778]
  [0.9854444 -0,02297129 ,9893,893 mila ,01,719607 millions]
  [0,9880944 0,25013754 0,9989148 0,37816945]
] ]
I punteggi sono: [
[0,671875 0,54296875 0,07421875 0,02734375 0,02734375 0,02734375
  0,0234375 0,0234375 0,0234375 0,0234375 0,01953125 0,01953125
  0,01953125 0,01953125 0,01953125 0,015625 0,015625 0,015625
  0,0256
]

...
```

A un esame più attento, vedrai che anche qui il numero di elementi nella matrice di rettangoli e la matrice di punteggi sono gli stessi!

Gli elementi nell'array dei punteggi sono tutte le probabilità per ciascuno dei rettangoli rilevati; quindi prenderemo solo quei rettangoli i cui punteggi sono maggiori di una determinata soglia, diciamo 0,5.

Ecco uno snippet di codice che applica questo filtro allo snippet di codice sopra:

```
importa tensorflow come tf
importa numpy come np
importa cv2
import pathlib

interprete = tf . contrib . lite . Interprete ( model_path = "object_detection.tflite" )

input_details = interprete . get_input_details ()
output_details = interprete . get_output_details ()

print ( input_details )
stampa ( output_details )

interprete . allocate_tensors ()

per il file in pathlib . Percorso ( "immagini" ). iterdir ():

    img = cv2 . imread ( r "{}" . format ( file . resolution ()))
    new_img = cv2 . ridimensiona ( img , ( 512 , 512 ))

    interprete . set_tensor ( input_details [ 0 ] [ 'index' ], [ new_img ])

    interprete . invocare ()
    rects = interprete . get_tensor (
        output_details [ 0 ] [ 'index' ])
    punteggi = interprete . get_tensor (
        output_details [ 2 ] [ 'index' ])

    per indice , punteggio in enumerate ( punteggi [ 0 ]):
        se punteggio > 0,5 :
            print ( "For file {}" . format ( file . stem ))
```

```
Per il file 1DSCF1649 (2) -2020-01-25T15: 16: 25.462Z, i
rettangoli sono: [-0.00325413 0.47174522 0.1879216 0.61565363]

Per il file 1DSCF1649 (2) -2020-01-25T15: 16: 25.462Z, i
rettangoli sono: [-0.00632208 0.24867406 0.17483354 0.3669869]

Per il file 1IMG_20191101_183819_154-2020-01-04T11: 20: 44.140Z i
rettangoli sono: [0.01136297 0.00234886 0.53366566 0.47188312]

Per il file 1IMG_20191101_183819_154-2020-01-04T11: 20: 44.140Z i
rettangoli sono: [0.00472282 0.6394675 0.2634614 0.9956119]
```

Come puoi vedere qui, dopo aver impostato un filtro di 0,5 sui punteggi, otteniamo un numero di rettangoli sostanzialmente ridotto (1 nella maggior parte dei casi).

Successivamente, useremo OpenCV per tracciare questi rettangoli sull'immagine originale e mostrarla sullo schermo.

Passaggio 5: tracciare i rettangoli rilevati con OpenCV

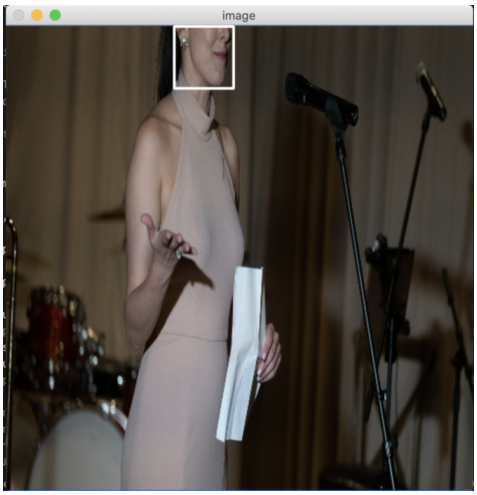
Per calcolare le coordinate x e y del rettangolo da disegnare, utilizzeremo una funzione di supporto che accetta la casella rilevata e converte i suoi elementi nelle coordinate xey corrette di un rettangolo che OpenCV può utilizzare.

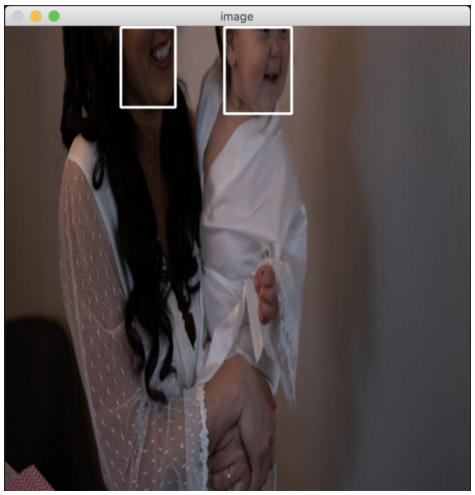
Ecco come appare:

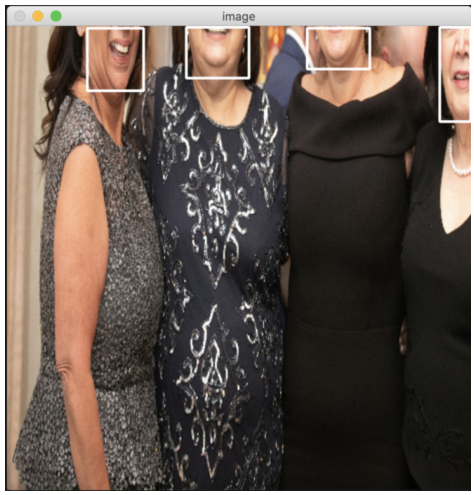
```
def  draw_rect ( immagine , scatola ):
    y_min  =  int ( max ( 1 , ( box [ 0 ] *  image . height )))
    x_min  =  int ( max ( 1 , ( box [ 1 ] *  image . width )))
    max_y  =  int ( min ( immagine . altezza , ( casella [ 2 ] *  immagine . altezza )))
    x_max  =  int ( min ( immagine . larghezza , ( box [ 3 ] *  immagine . larghezza )))

    # disegna un rettangolo sull'immagine
    cv2 . rettangolo ( immagine , ( x_min , y_min ), ( x_max , y_max ), ( 255 , 255 , 255 ), 2 )
```

main.py ospitato con ❤️ da GitHub visualizza raw

E questo è tutto! Dopo aver eseguito il codice completo, integrato con questo metodo di supporto sopra, ecco cosa dovresti vedere:







Come visto qui, il modello fa un ottimo lavoro di identificazione e rilevamento dei volti occlusi in un'immagine.

Se stai cercando il codice sorgente completo dell'esempio, ecco come appare:

```
importa tensorflow come tf
importa numpy come np
importa cv2
import pathlib

interprete = tf . contrib . lite . Interprete ( model_path = "object_detection.tflite" )

input_details = interprete . get_input_details ()
output_details = interprete . get_output_details ()

print ( input_details )
stampa ( output_details )

interprete . allocate_tensors ()

def draw_rect ( immagine , scatola ):
    y_min = int ( max ( 1 , ( box [ 0 ] * image . height )))
    x_min = int ( max ( 1 , ( box [ 1 ] * image . width )))
    max_y = int ( min ( immagine . altezza , ( casella [ 2 ] * immagine . altezza )))
    x_max = int ( min ( immagine . larghezza , ( box [ 3 ] * immagine . larghezza )))

    # disegna un rettangolo sull'immagine
    cv2 . rettangolo ( immagine , ( x_min , y_min ), ( x_max , y_max ), ( 255 , 255 , 255 ), 2

per il file in pathlib . Percorso ( "immagini" ). iterdir ():

    se file . suffisso ! = ".jpg" e file . suffisso ! = '.png' :
        Continua

    img = cv2 . imread ( r "{}" . format ( file . resolution ()))
    new_img = cv2 . ridimensiona ( img , ( 512 , 512 ))
    interprete . set_tensor ( input_details [ 0 ] [ 'index' ], [ new_img ])

    interprete . invocare ()
    rects = interprete . get_tensor (
        output_details [ 0 ] [ 'index' ])

    punteggi = interprete . get_tensor (
        output_details [ 2 ] [ 'index' ])

    per indice , punteggio in enumerate ( punteggi [ 0 ]):
        se punteggio > 0,5 :
          draw_rect ( new_img , rects [ 0 ] [ index ])
```

E questo è tutto! Sebbene non sia sempre la soluzione più efficace, nell'ultimo post e in questo abbiamo visto quanto sia facile caricare ed eseguire modelli TensorFlow Lite in un'impostazione basata su Python.

Se hai domande o suggerimenti su questo post, sentiti libero di lasciare un commento in basso e sarò felice di ricontattarti!

Grazie per aver letto! Se ti è piaciuta questa storia, fai ***clic sul pulsante*** 👏 ***e condividila*** *per aiutare gli altri a trovarla! Sentiti libero di lasciare un commento* 💬 *qui sotto.*

Hai un feedback? Collegiamoci ***su Twitter*** *.*

Nota del redattore: ***Heartbeat*** *è una pubblicazione online e una community guidata da collaboratori dedicati all'esplorazione dell'intersezione emergente tra lo sviluppo di app mobili e l'apprendimento automatico. Ci impegniamo a supportare e ispirare sviluppatori e ingegneri di tutti i ceti sociali.*

Editorialmente indipendente, Heartbeat è sponsorizzato e pubblicato da ***Fritz AI*** *, la piattaforma di machine learning che aiuta gli sviluppatori a insegnare ai dispositivi a vedere, ascoltare, percepire e pensare. Paghiamo i nostri collaboratori e non vendiamo annunci.*

Se desideri contribuire, vai alla nostra ***chiamata per i contributori*** *. Puoi anche iscriverti per ricevere le nostre newsletter settimanali (* ***Deep Learning Weekly*** *e* ***Fritz AI Newsletter*** *), unirti a noi su* ***Slack*** *e seguire Fritz AI su* ***Twitter*** *per tutte le ultime novità sul machine learning mobile.*

Apprendimento automatico Pitone TensorFlow Tensorflow Lite Battito cardiaco