Report dell'Assignment 4 di Advanced Machine Learning

Paolo Mariani 800307

Novembre 2019

1 Introduzione

Lo scopo dell'assignment 4 è quello di utilizzare la tecnica del Transfer Learning per poter effettuare classificazione di immagini su di un dataset a scelta.
Il **Transfer Learning** permette di svolgere un task specifico su di un Dataset sfruttando i parametri di una rete neurale convoluzionale complessa che viene allenata su Dataset di grandi dimensioni (ImageNet ad esempio).
Si può effettuare Transfer Learning per due principali scopi:

- Effettuare **Fine Tuning**, cioè sfruttare la conoscenza di una rete già allenata per effettuare classificazione, effettuando delle modifiche come tagliare layer e inserire nuovi strati per garantire una specializzazione della rete al nuovo task.
- Effettuare **Feature Extraction** delle immagini, utilizzando le porzioni di interesse della rete neurale complessa, in modo da ridurre le dimensioni dell'input che può essere fornito ad un classificatore, garantendo una complessità ridotta del task.

In entrambi i casi si procede a modificare la rete neurale a proprio piacimento per poter specializzare il modello per svolgere il task di classificazione, aggiungendo e tagliando strati. È fondamentale comprendere che gli strati più profondi (iniziali) della rete sono utilizzati per estrarre e codificare informazioni basilari dalle immagini come ad esempio effettuare edge detection o blob detection. Sfruttando la codifica generica che producono si può codificare una immagine (senza effettuare training dei layer considerati che sono bloccati in questa fase) coi pesi della rete originale ed eventualmente aggiungere altri strati per specializzarla nel riconoscere correttamente le immagini del dataset collegato al target specifico.
La tecnica di Transfer Learning viene utilizzata in questo Assignment per effettuare Feature Extraction delle immagini, selezionando a partire dalla rete neurale VGG16 diversi strati da cui ottenere diverse codifiche dell'input che sarà fornito ad un classificatore generico.
Il Dataset che è stato scelto è il **Sign Language Digits Dataset**, contiene 2062 immagini di cifre (da 0 a 9) rappresentate nel linguaggio dei segni. Si mostrano alcune immagini di esempio per comprendere il contenuto del Dataset:

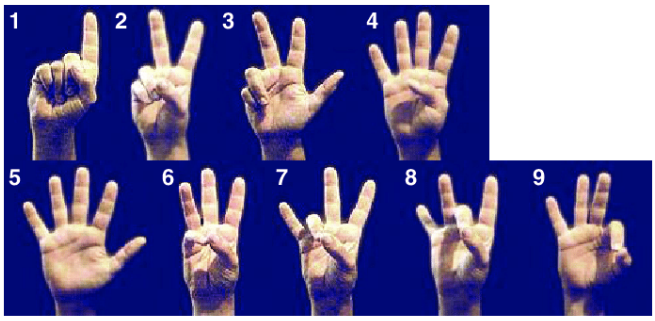

Figura 1: Codifica del linguaggio dei segni

1.1 Dataset

Sul dataset sono state effettuate alcune operazioni di Preprocessing.
Il valore delle Label associato ad ogni immagine è espresso in forma one-hot encoding, si procede tramite la funzione *np.argmax()* a convertire ogni valore rappresentato tramite un intero.
Si osserva poi il seguente risultato ottenuto stampando a schermo l'immagine e il relativo label:

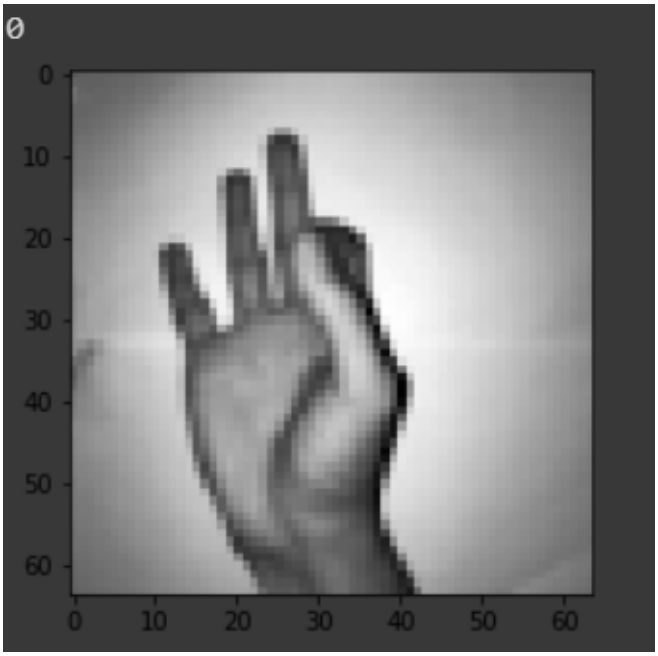

Figura 2: Immagine con label sbagliato

L'immagine che viene rappresentata si codifica con la cifra 9 nel linguaggio dei segni, ma il label associato ad essa che viene stampato a schermo è 0. Dopo aver effettuato una analisi del dataset ci si accorge che le label sono scambiate, e per garantire una coerenza tra immagine e label associato si procede ad effettuare una modifica dei valori dei label con quelli corretti.

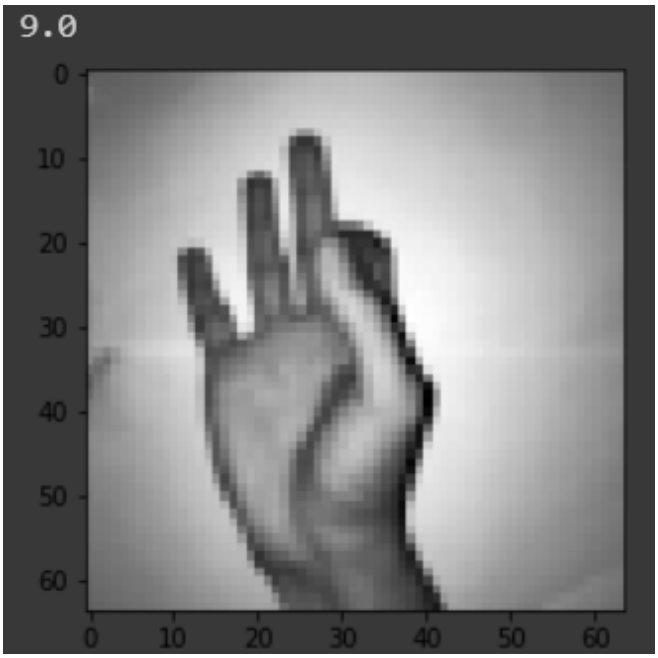

Figura 3: Immagine con label corretto

Si può osservare che il valore ora è corretto.
Ogni immagine è rappresentata in scala di grigi e possiede un solo canale che rappresenta il colore di ogni pixel, la dimensione di ogni immagine è ***64*64*1***; la CNN utilizzata per la risoluzione del task richiede che la ogni immagine possieda 3 canali relativi alla codifica RGB: per effettuare il cambiamento su ogni singola osservazione del dataset si procede ad utilizzare la funzione *cvtColor* con parametri *cv2.COLOR_GRAY2RGB* per trasformare l'immagine dalla scala di grigi in RGB.
La dimensione di ogni immagine ora è ***64*64*3***.
Si osserva la distribuzione dei valori delle immagini nelle classi:

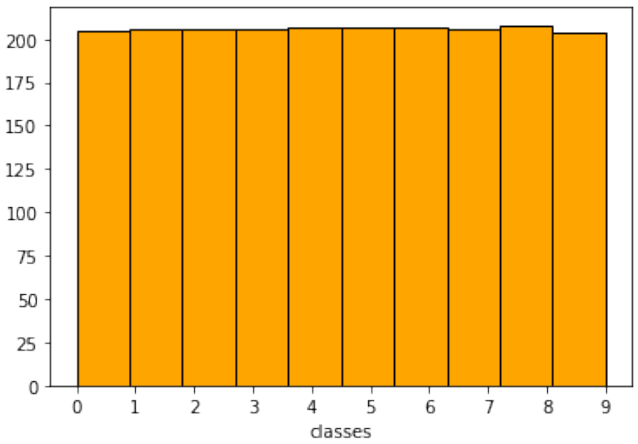

Figura 4: Istogramma delle classi

2 Train-Test Split

Si effettua uno split dei dati, da 2062 immagini si dedica il 20% dei dati al Test Set e il rimanente 80% al Training Set, il campionamento è stratificato sui valori della variabile dipendente "labels".

3 Feature Extraction

Si utilizza la rete neurale VGG16 per effettuare il compito di Feature Extraction, viene importata nell'ambiente di lavoro fornendo come parametri:

- *weighting*: "ImageNet", cioè i pesi della rete salvati per il compito di classificazione sulla ImageNet
- *input_shape*: 64*64*3
- *pooling*: "avg"

- *include_top*: "False", permette di non inserire i layer di tipo Fully Connected in cima alla CNN

Saranno implementate 3 diverse soluzioni per la codifica delle immagini, utilizzando la CNN come feature extractor e considerando un numero diverso di layer ad ogni soluzione.
Si osserva la struttura della rete VGG16:

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

Figura 5: struttura della VGG16

La strategia che si vuole implementare è quella di tagliare la rete alla fine di ogni "block" i-esimo a partire dallo strato block4_pool:

3.1 Prima Soluzione

La rete convoluzionale VGG16 viene tagliata al layer *"block4_pool"*, è un layer di tipo MaxPooling2D che produce un output di dimensione *4*4*512* (4*4 è la dimensione dell'output, 512 sono i canali).

Si genera un modello ***net*** che copia i layer della rete VGG, e possiede in coda un layer *Flatten()* per poter rappresentare la codifica (con dimensione pari a 8192) con un vettore.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
=================================================================
Total params: 7,635,264
Trainable params: 0
Non-trainable params: 7,635,264
```

Figura 6: rete utilizzata per feature extraction: net

Il modello ***net*** possiede 7,635,264 parametri, il risultato dell'elaborazione di ogni immagine è una codifica-vettore che verrà passata in input ad un classificatore SVM.

Il **classificatore SVM** che si implementa possiede kernel di tipo "rbf" (Radial Basis Function), con iperparametri ottimizzati con il processo di ottimizzazione *GridSearchCV()*. Gli iperparametri ottimali risultano:

- C:50
- Gamma: 0.5

Prima di procedere ad allenare il classificatore si effettua la codifica di tutte le immagini di Training Set e Test Set attraverso la rete *net*.
Le performance ottenute vengono riassunte dal-

la seguente immagine, generata dalla funzione "classification_report":

	precision	recall	f1-score	support
0.0	1.00	0.98	0.99	41
1.0	0.98	1.00	0.99	41
2.0	0.95	0.95	0.95	41
3.0	1.00	0.98	0.99	41
4.0	0.82	0.90	0.86	41
5.0	1.00	1.00	1.00	42
6.0	0.88	0.90	0.89	42
7.0	0.94	0.80	0.87	41
8.0	0.88	0.88	0.88	42
9.0	0.91	0.95	0.93	41
accuracy			0.93	413
macro avg	0.94	0.93	0.93	413
weighted avg	0.94	0.93	0.93	413

Figura 7: Performance della prima soluzione

3.2 Seconda Soluzione

La rete convoluzionale VGG16 nella seconda soluzione viene tagliata al layer *"block3_pool"*, è un layer di tipo MaxPooling2D che produce un output di dimensione *8*8*256* (8*8 è la dimensione dell'output, 256 sono i canali).

Model: "model_2"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 64, 64, 3)	0
block1_conv1 (Conv2D)	(None, 64, 64, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 64, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 32, 64)	0
block2_conv1 (Conv2D)	(None, 32, 32, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 32, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 16, 128)	0
block3_conv1 (Conv2D)	(None, 16, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 16, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 16, 256)	590080
block3_pool (MaxPooling2D)	(None, 8, 8, 256)	0
flatten_2 (Flatten)	(None, 16384)	0

Total params: 1,735,488
Trainable params: 0
Non-trainable params: 1,735,488

Figura 8: rete utilizzata per feature extraction: net2

Si genera un modello ***net2*** che copia i layer della rete VGG, e possiede in coda un layer *Flatten()* per poter rappresentare la codifica con un vettore: in questo caso la dimensione dell'output è pari a 16384 valori. Il modello ***net2*** possiede 1,735,488 parametri.

Il secondo **classificatore SVM** che si implementa possiede kernel di tipo "rbf" con iperparametri ottimizzati con il processo di ottimizzazione *GridSearchCV()*. Gli iperparametri ottimali risultano:

- C:50
- Gamma: 0.01

Prima di procedere ad allenare il classificatore si effettua la codifica di tutte le immagini di Training Set e Test Set attraverso la rete *net2*.
Le performance ottenute vengono riassunte dalla seguente immagine, generata dalla funzione "classification_report":

	precision	recall	f1-score	support
0.0	1.00	0.98	0.99	41
1.0	0.98	1.00	0.99	41
2.0	0.97	0.93	0.95	41
3.0	1.00	0.98	0.99	41
4.0	0.89	0.98	0.93	41
5.0	1.00	1.00	1.00	42
6.0	0.95	0.98	0.96	42
7.0	0.87	0.83	0.85	41
8.0	0.90	0.90	0.90	42
9.0	1.00	1.00	1.00	41
accuracy			0.96	413
macro avg	0.96	0.96	0.96	413
weighted avg	0.96	0.96	0.96	413

Figura 9: Performance della seconda soluzione

3.3 Terza Soluzione

La rete convoluzionale VGG16 nella terza soluzione viene tagliata al layer *"block2_pool"*, è un layer di tipo MaxPooling2D che produce un output di dimensione *16*16*128* (16*16 è la dimensione dell'output, 128 sono i canali).
Si genera un modello ***net3*** che copia i layer della rete VGG, e possiede in coda un layer *Flatten()* per poter rappresentare la codifica con un vettore: in questo caso la dimensione dell'output è pari a 32768 valori. Il modello ***net2*** possiede 260160 parametri.

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
flatten_3 (Flatten)          (None, 32768)             0
=================================================================
Total params: 260,160
Trainable params: 0
Non-trainable params: 260,160
_________________________________________________________________
```

Figura 10: rete utilizzata per feature extraction: net3

Il terzo **classificatore SVM** che si implementa possiede kernel di tipo "rbf", con iperparametri ottimizzati con il processo di ottimizzazione *GridSearchCV()*. Gli iperparametri ottimali risultano:

- C:50
- Gamma: 0.02

Prima di procedere ad allenare il classificatore si effettua la codifica di tutte le immagini di Training Set e Test Set attraverso la rete *net3*.
Le performance ottenute vengono riassunte dalla seguente immagine, generata dalla funzione "classification_report":

	precision	recall	f1-score	support
0.0	0.98	0.98	0.98	41
1.0	0.95	0.98	0.96	41
2.0	0.84	0.88	0.86	41
3.0	1.00	0.95	0.97	41
4.0	0.85	0.95	0.90	41
5.0	0.95	1.00	0.98	42
6.0	0.95	0.83	0.89	42
7.0	0.85	0.85	0.85	41
8.0	0.97	0.90	0.94	42
9.0	0.95	0.95	0.95	41
accuracy			0.93	413
macro avg	0.93	0.93	0.93	413
weighted avg	0.93	0.93	0.93	413

Figura 11: Performance della terza soluzione

4 Risultati

Osservando i risultati ottenuti dai diversi classificatori, che possiedono ciascuno iperparametri diversi e dati codificati con reti neurali diverse in input, si deduce che la seconda soluzione (Rete Neurale *net2* e SVM con *kernel rbf*, *gamma 0.01* e *C 50*) è la migliore tra le tre: garantisce di raggiungere una *Accuracy* pari a *96%*, *Precision 96%* e *Recall 96%*. Questa soluzione impiega circa 30 minuti per effettuare la codifica delle immagini e il training della SVM utilizzando la macchina virtuale offerta da Google Colab, che è un tempo intermedio tra le altre due soluzioni testate.