Relazione Assignment 4

Gianmaria Balducci

19 Dicembre 2020

Transfer Learning

Goal: The assignment consists in Transfer Learning using a CNN pretrained on IMAGENET. The CNN should be used as fixed feature extractor on a new task of my choice containing a number of classes in the range from 2 to 10.
Base Model: VGG16 trained on IMAGENET.
New Task: Image classification on CIFAR-10 dataset that consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Running this task on Google Colab, the computation has exceed RAM's limit offer by Colab, so i've decided to reduce CIFAR-10 considering only 6 categoires and all the data related to these categories.
The preproccesing step is the same used for the training of VGG, through keras it is possible to download both the model and the preprocess.

Experiment

I choose Linear Support Vector Classification for transfer learning experiment considering VGG16 as feature extractor considering 3 different layers. Linear SVC is trained with C parameter (that is regularization parameter) equal to *0.1* and *max_ iteration* equal to 1000, with loss function = squared_hinge

First cut

First cut consist in a simple way to cut fully convolutional layers of VGG16 simply get the model with parameter of function *include_ top = False*. In this case there are 14,714,688 learnable parameters. Whit this kind of cut it needs to extract features from *block5pool* layer but before i've added a flatten layer in order to flat this features and give them to the classifier. So with this first model i predict with training set of CIFAR-10 and save this features in order to use the CNN as features extractors on my new task.

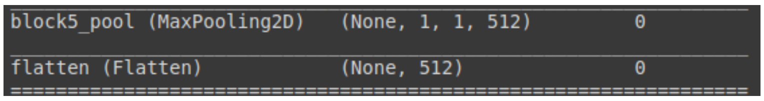
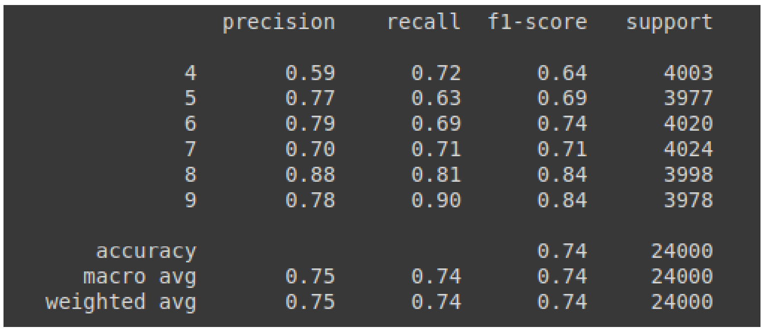

```
block5_pool (MaxPooling2D)   (None, 1, 1, 512)        0
_________________________________________________________________
flatten (Flatten)            (None, 512)              0
=================================================================
```

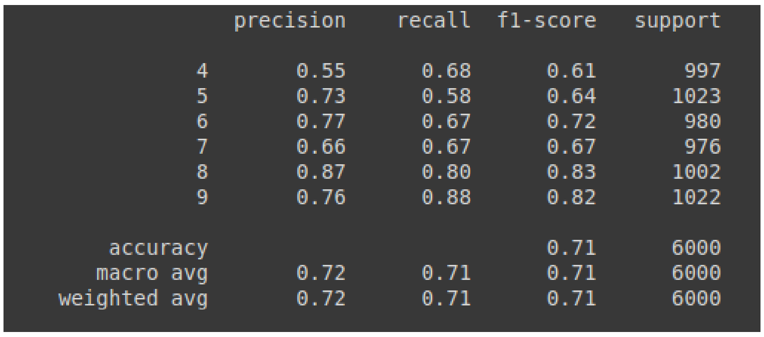
Figura 1: Output layer

With flatten layer i reshape the output dimension of block5_pool from (1,1,512) to (1, 512).
After loading this new features, i split train features to create validation set and evaluate the training. At this point I give this features in order to train new classifier Linear SVC. Below are the performances which are not exciting, maybe this cut is too near the output and the task that i choosed is not so similar to the task of imagenet. Next cut will be further away from the output.



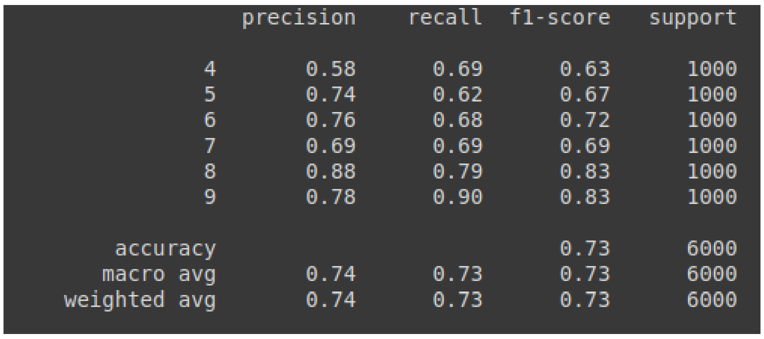
```
              precision    recall  f1-score   support

           4       0.59      0.72      0.64      4003
           5       0.77      0.63      0.69      3977
           6       0.79      0.69      0.74      4020
           7       0.70      0.71      0.71      4024
           8       0.88      0.81      0.84      3998
           9       0.78      0.90      0.84      3978

    accuracy                           0.74     24000
   macro avg       0.75      0.74      0.74     24000
weighted avg       0.75      0.74      0.74     24000
```

Figura 2: Training Performances



```
              precision    recall  f1-score   support

           4       0.55      0.68      0.61       997
           5       0.73      0.58      0.64      1023
           6       0.77      0.67      0.72       980
           7       0.66      0.67      0.67       976
           8       0.87      0.80      0.83      1002
           9       0.76      0.88      0.82      1022

    accuracy                           0.71      6000
   macro avg       0.72      0.71      0.71      6000
weighted avg       0.72      0.71      0.71      6000
```

Figura 3: Validation Performances

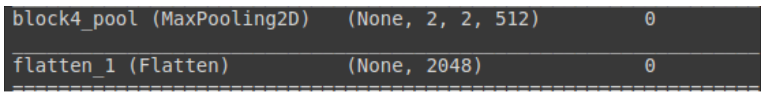


```
              precision    recall  f1-score   support

           4       0.58      0.69      0.63      1000
           5       0.74      0.62      0.67      1000
           6       0.76      0.68      0.72      1000
           7       0.69      0.69      0.69      1000
           8       0.88      0.79      0.83      1000
           9       0.78      0.90      0.83      1000

    accuracy                           0.73      6000
   macro avg       0.74      0.73      0.73      6000
weighted avg       0.74      0.73      0.73      6000
```

Figura 4: Test Performances

Second cut

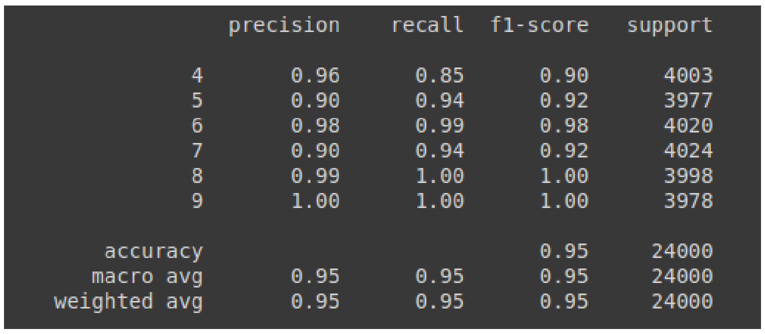
So lets see if cutting VGG16 closer to the input the accuracy of new classification task improves.
Second cut is after block4_pool layer of VGG16.
The process is the same: insert flatten layer to reshape the features , with this model (VGG16 until *block_4pool* plus flatten layer) predict on CIFAR-10 training set reduced as described and save this feature for use with SVC.



```
block4_pool (MaxPooling2D)   (None, 2, 2, 512)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)             0
=================================================================
```

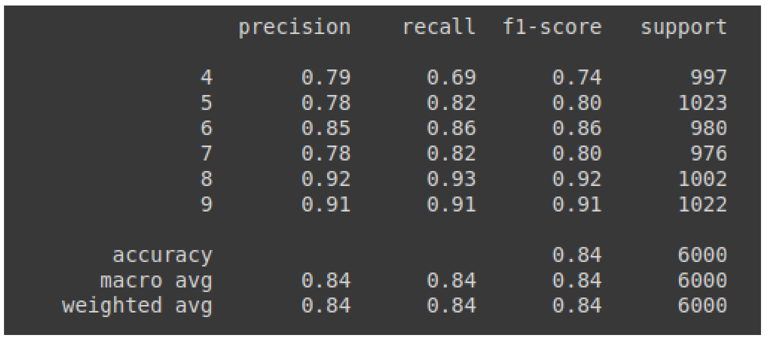
Figura 5: Output layer

With flatten layer i reshape the output dimension of block5_pool from (2,2,512) to (1, 2048).
The peformances are improved respect to the first cut and this means that taking features more generic from VGG16 this new model is capable to classify better.



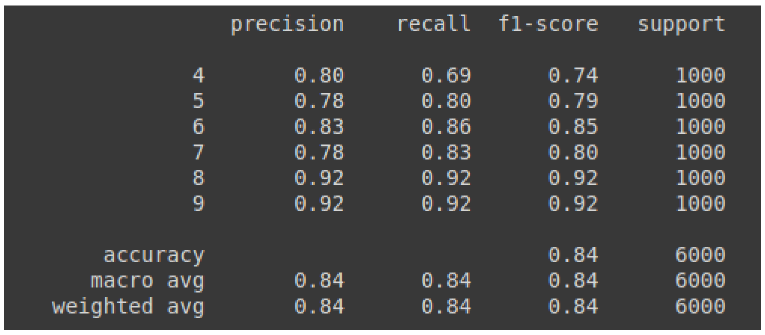
```
              precision    recall  f1-score   support

           4       0.96      0.85      0.90      4003
           5       0.90      0.94      0.92      3977
           6       0.98      0.99      0.98      4020
           7       0.90      0.94      0.92      4024
           8       0.99      1.00      1.00      3998
           9       1.00      1.00      1.00      3978

    accuracy                           0.95     24000
   macro avg       0.95      0.95      0.95     24000
weighted avg       0.95      0.95      0.95     24000
```

Figura 6: Training Performances- cut2



```
              precision    recall  f1-score   support

           4       0.79      0.69      0.74       997
           5       0.78      0.82      0.80      1023
           6       0.85      0.86      0.86       980
           7       0.78      0.82      0.80       976
           8       0.92      0.93      0.92      1002
           9       0.91      0.91      0.91      1022

    accuracy                           0.84      6000
   macro avg       0.84      0.84      0.84      6000
weighted avg       0.84      0.84      0.84      6000
```

Figura 7: Validation Performances-cut2

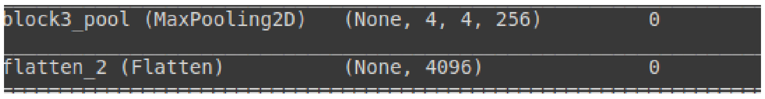


```
              precision    recall  f1-score   support

           4       0.80      0.69      0.74      1000
           5       0.78      0.80      0.79      1000
           6       0.83      0.86      0.85      1000
           7       0.78      0.83      0.80      1000
           8       0.92      0.92      0.92      1000
           9       0.92      0.92      0.92      1000

    accuracy                           0.84      6000
   macro avg       0.84      0.84      0.84      6000
weighted avg       0.84      0.84      0.84      6000
```

Figura 8: Test Performances-cut2

Third cut

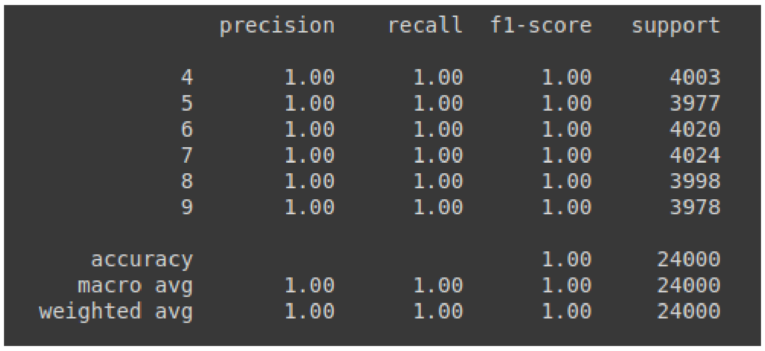
This cut is closer to the input respect the previous one, it is done after block3_pool layer of VGG16.
At this level the features are more generic and the results obtain in the previous cuts suggest that the task that i choice is very different and the closer you cut to the inputs, the better the performance, but this is not so true because i've tried to cut VGG16 model after *block2_pool* with worse perfromances on validation and test set.



```
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0

flatten_2 (Flatten)          (None, 4096)              0
```

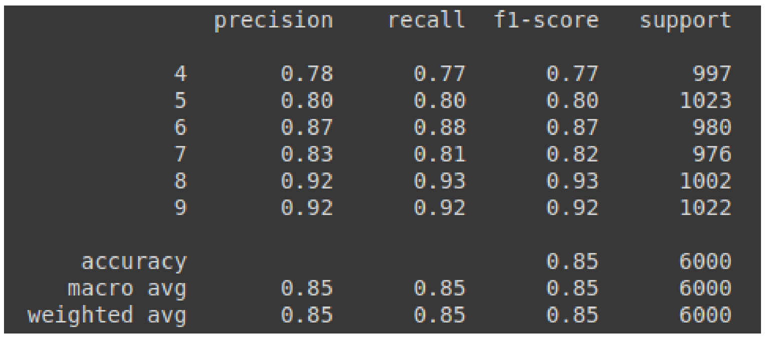
Figura 9: Output layer

With flatten layer i reshape the output dimension of block5_pool from (4,4,512) to (1, 4096).
With the same process described above the results are:



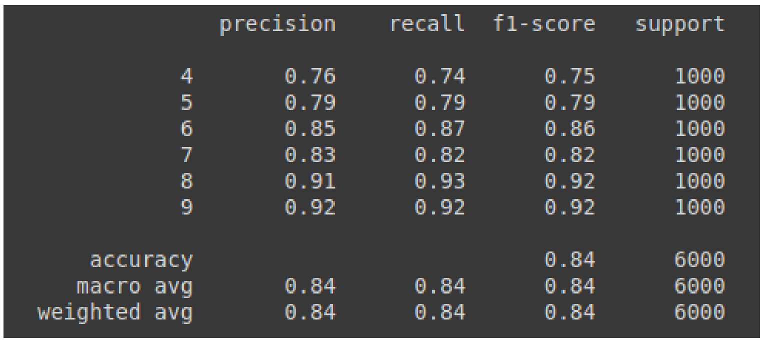
```
              precision    recall  f1-score   support

           4       1.00      1.00      1.00      4003
           5       1.00      1.00      1.00      3977
           6       1.00      1.00      1.00      4020
           7       1.00      1.00      1.00      4024
           8       1.00      1.00      1.00      3998
           9       1.00      1.00      1.00      3978

    accuracy                           1.00     24000
   macro avg       1.00      1.00      1.00     24000
weighted avg       1.00      1.00      1.00     24000
```

Figura 10: Training Performances- cut3



```
              precision    recall  f1-score   support

           4       0.78      0.77      0.77       997
           5       0.80      0.80      0.80      1023
           6       0.87      0.88      0.87       980
           7       0.83      0.81      0.82       976
           8       0.92      0.93      0.93      1002
           9       0.92      0.92      0.92      1022

    accuracy                           0.85      6000
   macro avg       0.85      0.85      0.85      6000
weighted avg       0.85      0.85      0.85      6000
```

Figura 11: Validation Performances-cut3



```
              precision    recall  f1-score   support

           4       0.76      0.74      0.75      1000
           5       0.79      0.79      0.79      1000
           6       0.85      0.87      0.86      1000
           7       0.83      0.82      0.82      1000
           8       0.91      0.93      0.92      1000
           9       0.92      0.92      0.92      1000

    accuracy                           0.84      6000
   macro avg       0.84      0.84      0.84      6000
weighted avg       0.84      0.84      0.84      6000
```

Figura 12: Test Performances-cut3

Comparison and Results

Below there are the classification performances respect to differents layers where the features are been extracted. In the plots first layer is *black5_pool* that is near to the output of VGG16, so the second and the third are closer to the input. As you can see accuracy improves considering layers closer to the input of VGG16, this means that the task of classification on CIFAR-10 reduced is very different to the task in which VG16 is trained.
Therefore for CIFAR-10 reduced at six classes we needs to extract more genreal features in order to achieve better performances on this new task.

	block5_pool	**block4_pool**	**block3_pool**
Training Accuracy	0.743	0.952	1.0
Validation Accuracy	0.714	0.838	0.853
Test Accuracy	0.729	0.837	0.844

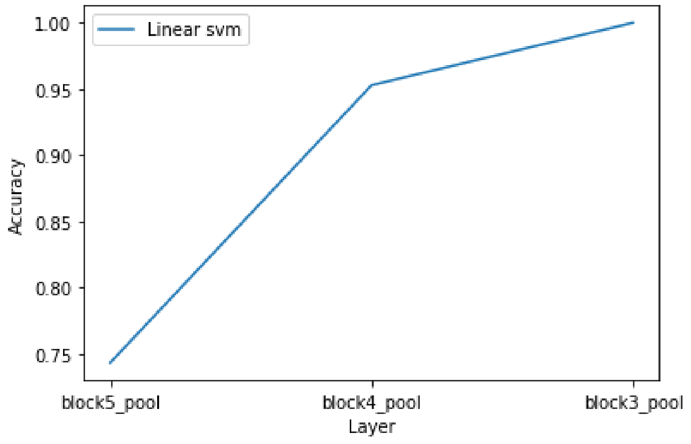


Figura 13: Training Accuracy - Layers

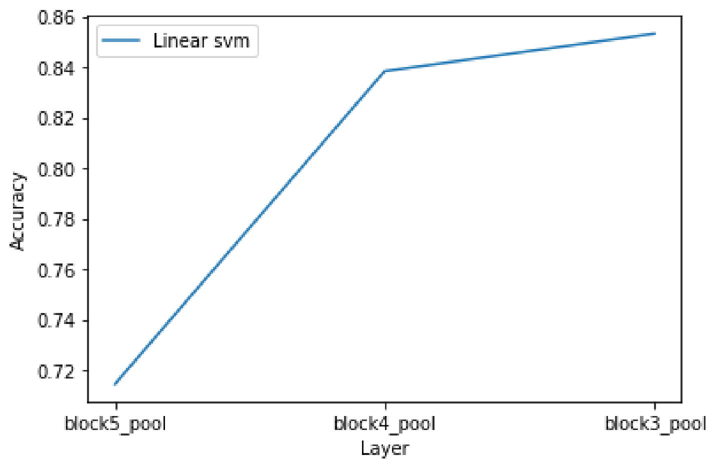


Figura 14: Validation accuracy-Layers

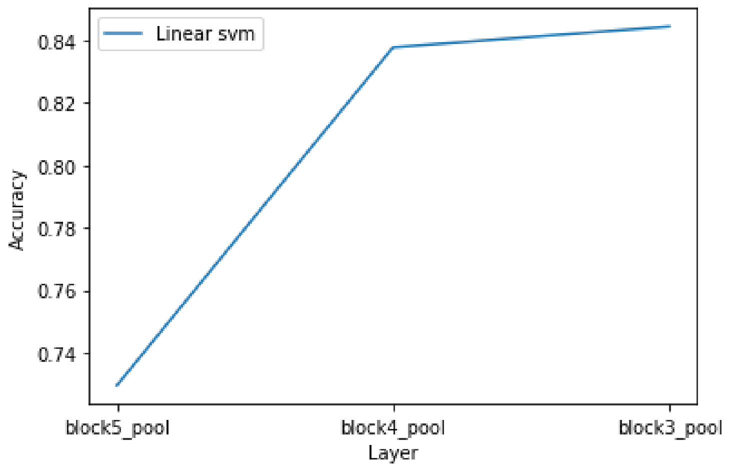


Figura 15: Test Accuracy- Layers