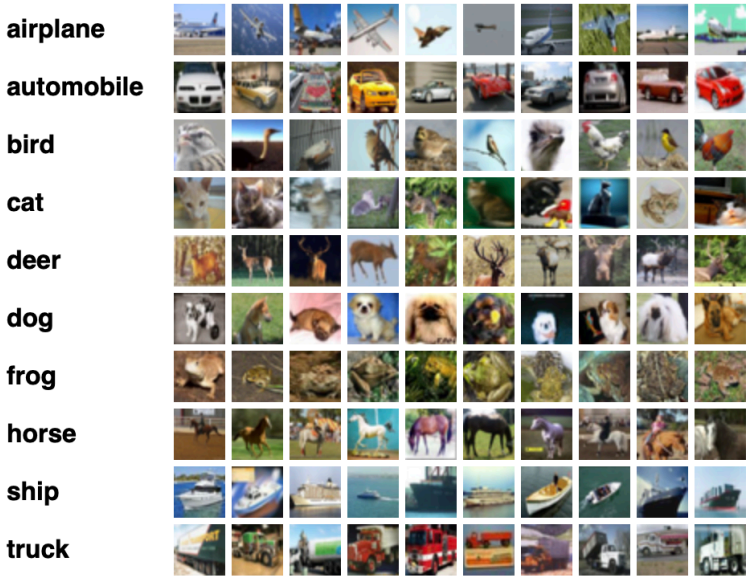
Object Detection del dataset CIFAR-10

Progetto di Machine Learning AA2021/22

Studente Massimiliano Patrizi
Professore Giuseppe Sansonetti
Professore Alessandro Micarelli
Università degli Studi Roma Tre

Questo progetto si pone l'obiettivo di classificare il dataset CIFAR-10 tramite una rete neurale. La classificazione viene effettuando allenando il modello in più modi, cambiando il numero delle epoche e applicando la tecnica nota come Data Augmentation.

Presentazione del dataset



Il dataset utilizzato è denominato CIFAR-10 (**https://www.cs.toronto.edu/~kriz/cifar.html**), è composto da **60.000 immagini** a colori 32x32 divise in 10 classi (6.000 immagini per classe), come possiamo notare nell'immagine in alto.
Ci sono 50.000 immagini per il training e 10.000 per il test.

Tecnologie utilizzate

Per la semplicità di utilizzo e la sua comodità è stato usato **Google Colab** come ambiente di lavoro (anche perchè sul mio portatile con chip M1 di Apple, sono stati riscontrati spesso problemi con l'utilizzo dei notebook Jupyter in locale).

Lo strumento che è alla base di questo progetto è **Tensorflow**, piattaforma nata per creare progetti di Machine Learning, ed è stata usata anche la sua API **Keras**.

Come librerie di Python, sono state impiegate **numpy** (per i calcoli numerici) e **matplotlib** (per creare grafici).

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical, plot_model
```

Preparazione del dataset

Dopo aver suddiviso il dataset dividendo immagini e label corrispondenti in training e test, viene creata una **Convolutional Neural Network (CNN)**, ovvero un tipo di rete neurale artificiale usata principalmente per analizzare immagini.

All'interno di essa avvengono diverse operazioni:

- **Conv2D** : Convoluzioni 2D dell'input, ovvero vengono applicati dei filtri sugli input che effettuano operazioni matematiche per estrarre le features dai dati;
- **BatchNormalization** : normalizza l'attivazione del layer precedente;
- **MaxPooling2D** : effettua delle operazioni di downsampling per ridurre la dimensione spaziale dell'input ed il numero di parametri nel modello.

Fase di training e risultati

Una volta definita la rete e configurato il modello, inizia la fase di training. Come detto nell'introduzione, la fase di training è stata svolta in 4 modi diversi:

1. 20 epoche - fit iniziale e fit con Data Augmentation a seguire
2. 20 epoche - solo fit con Data Augmentation
3. 50 epoche - fit iniziale e fit con Data Augmentation a seguire
4. 50 epoche - solo fit con Data Augmentation

La **Data Augmentation** è una tecnica che genera ulteriori dati di training applicando trasformazioni random ai dati già esistenti, in modo ad aiutare a generalizzare il modello e a ridurre il rischio di overfitting.
Vediamo i singoli casi nel dettaglio:

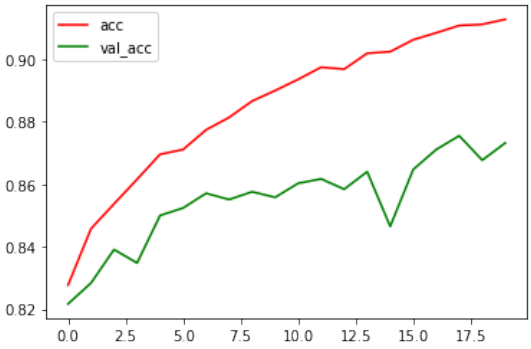
```
Epoch 1/20
1563/1563 [==============================] - 21s 8ms/step - loss: 1.2890 - accuracy: 0.5569 - val_loss: 0.9606 - val_accuracy: 0.6639
Epoch 2/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.8398 - accuracy: 0.7101 - val_loss: 0.7871 - val_accuracy: 0.7239
Epoch 3/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.6848 - accuracy: 0.7650 - val_loss: 0.7193 - val_accuracy: 0.7622
.
.
Epoch 9/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.2611 - accuracy: 0.9101 - val_loss: 0.6057 - val_accuracy: 0.8076
Epoch 10/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.2190 - accuracy: 0.9252 - val_loss: 0.6538 - val_accuracy: 0.8154
Epoch 11/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.1961 - accuracy: 0.9332 - val_loss: 0.7208 - val_accuracy: 0.8014
.
.
Epoch 18/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.1069 - accuracy: 0.9636 - val_loss: 0.6929 - val_accuracy: 0.8326
Epoch 19/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.1018 - accuracy: 0.9664 - val_loss: 0.7371 - val_accuracy: 0.8270
Epoch 20/20
1563/1563 [==============================] - 11s 7ms/step - loss: 0.0949 - accuracy: 0.9684 - val_loss: 0.7332 - val_accuracy: 0.8347
```

Fit iniziale e fit con Data Aug. a seguire **(20 epoche)**

```
Epoch 1/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.5209 - accuracy: 0.8277 - val_loss: 0.5564 - val_accuracy: 0.8217
Epoch 2/20
1562/1562 [==============================] - 32s 20ms/step - loss: 0.4617 - accuracy: 0.8458 - val_loss: 0.5250 - val_accuracy: 0.8284
Epoch 3/20
1562/1562 [==============================] - 32s 20ms/step - loss: 0.4315 - accuracy: 0.8537 - val_loss: 0.4799 - val_accuracy: 0.8391
.
.
Epoch 9/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.3347 - accuracy: 0.8866 - val_loss: 0.4477 - val_accuracy: 0.8576
Epoch 10/20
1562/1562 [==============================] - 30s 19ms/step - loss: 0.3225 - accuracy: 0.8900 - val_loss: 0.4395 - val_accuracy: 0.8558
Epoch 11/20
1562/1562 [==============================] - 30s 19ms/step - loss: 0.3130 - accuracy: 0.8936 - val_loss: 0.4290 - val_accuracy: 0.8603
.
.
Epoch 18/20
1562/1562 [==============================] - 30s 19ms/step - loss: 0.2613 - accuracy: 0.9108 - val_loss: 0.3719 - val_accuracy: 0.8755
Epoch 19/20
1562/1562 [==============================] - 32s 20ms/step - loss: 0.2556 - accuracy: 0.9111 - val_loss: 0.4121 - val_accuracy: 0.8677
Epoch 20/20
1562/1562 [==============================] - 30s 19ms/step - loss: 0.2556 - accuracy: 0.9127 - val_loss: 0.4011 - val_accuracy: 0.8732
```

Fit iniziale e **fit con Data Aug. a seguire (20 epoche)**

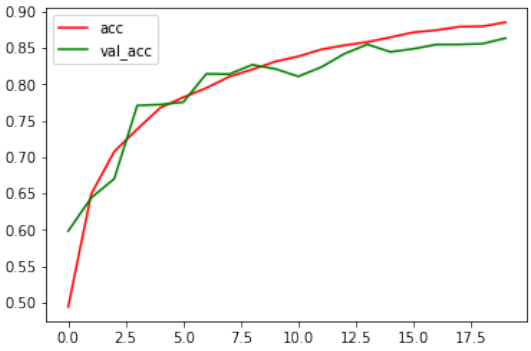
Possiamo notare come la *validation_loss* viene ridotta drasticamente dopo aver riallenato il modello con i dati trasformati, passando da 0.7332 a 0.4011, ed anche la *val_accuracy* incrementa, sebbene non dello stesso margine.



Dal grafico possiamo vedere che la *val_accuracy*, ovvero la precisione del modello rispetto a dati nuovi, tende ad aumentare all'aumentare dell'*accuracy*, per cui possiamo evincere che non vi sia presente overfitting.

```
Epoch 1/20
1562/1562 [==============================] - 35s 20ms/step - loss: 1.4559 - accuracy: 0.4941 - val_loss: 1.2110 - val_accuracy: 0.5985
Epoch 2/20
1562/1562 [==============================] - 32s 20ms/step - loss: 1.0037 - accuracy: 0.6493 - val_loss: 1.1159 - val_accuracy: 0.6442
Epoch 3/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.8556 - accuracy: 0.7077 - val_loss: 0.9428 - val_accuracy: 0.6703
.
.
Epoch 9/20
1562/1562 [==============================] - 30s 19ms/step - loss: 0.5241 - accuracy: 0.8206 - val_loss: 0.5343 - val_accuracy: 0.8271
Epoch 10/20
1562/1562 [==============================] - 32s 20ms/step - loss: 0.4953 - accuracy: 0.8315 - val_loss: 0.5352 - val_accuracy: 0.8214
Epoch 11/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.4728 - accuracy: 0.8386 - val_loss: 0.5784 - val_accuracy: 0.8110
.
.
Epoch 18/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.3503 - accuracy: 0.8794 - val_loss: 0.4544 - val_accuracy: 0.8550
Epoch 19/20
1562/1562 [==============================] - 31s 20ms/step - loss: 0.3462 - accuracy: 0.8799 - val_loss: 0.4458 - val_accuracy: 0.8560
Epoch 20/20
1562/1562 [==============================] - 29s 19ms/step - loss: 0.3324 - accuracy: 0.8855 - val_loss: 0.4185 - val_accuracy: 0.8636
```

Solo fit con Data Aug. (20 epoche)



Da questo secondo grafico possiamo tirare fuori le stesse conclusioni del caso precedente, avendo un andamento dell'accuracy molto simile, nonostante il modello sia stato allenato solamente una volta.

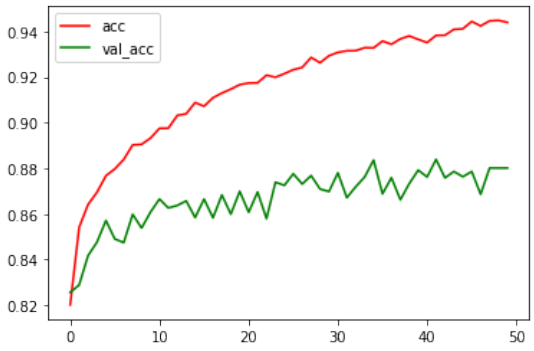
Vediamo ora i casi in cui sono stati effettuati dei training di 50 epoche:

```
Epoch 1/50
1563/1563 [==============================] - 12s 7ms/step - loss: 1.3216 - accuracy: 0.5470 - val_loss: 0.9964 - val_accuracy: 0.6533
Epoch 2/50
1563/1563 [==============================] - 11s 7ms/step - loss: 0.8568 - accuracy: 0.7042 - val_loss: 0.8507 - val_accuracy: 0.7044
Epoch 3/50
1563/1563 [==============================] - 15s 10ms/step - loss: 0.7009 - accuracy: 0.7583 - val_loss: 0.8044 - val_accuracy: 0.7304
.
.
Epoch 24/50
1563/1563 [==============================] - 12s 8ms/step - loss: 0.0861 - accuracy: 0.9720 - val_loss: 0.8280 - val_accuracy: 0.8323
Epoch 25/50
1563/1563 [==============================] - 11s 7ms/step - loss: 0.0790 - accuracy: 0.9737 - val_loss: 0.7939 - val_accuracy: 0.8379
Epoch 26/50
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0746 - accuracy: 0.9754 - val_loss: 0.8179 - val_accuracy: 0.8352
.
.
Epoch 48/50
1563/1563 [==============================] - 12s 7ms/step - loss: 0.0400 - accuracy: 0.9868 - val_loss: 0.9903 - val_accuracy: 0.8185
Epoch 49/50
1563/1563 [==============================] - 12s 8ms/step - loss: 0.0444 - accuracy: 0.9857 - val_loss: 1.0815 - val_accuracy: 0.8365
Epoch 50/50
1563/1563 [==============================] - 13s 8ms/step - loss: 0.0394 - accuracy: 0.9870 - val_loss: 0.9908 - val_accuracy: 0.8305
```

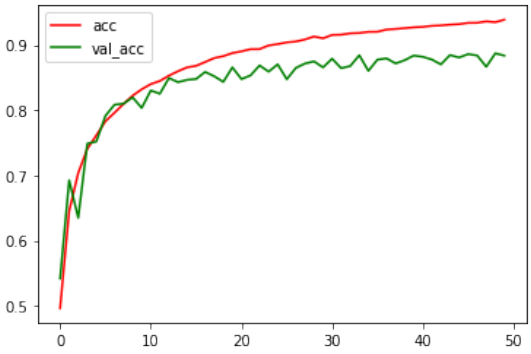
Fit iniziale e fit con Data Aug. a seguire **(50 epoche)**

```
Epoch 1/50
1562/1562 [==============================] - 40s 25ms/step - loss: 0.5600 - accuracy: 0.8201 - val_loss: 0.5368 - val_accuracy: 0.8256
Epoch 2/50
1562/1562 [==============================] - 37s 24ms/step - loss: 0.4461 - accuracy: 0.8541 - val_loss: 0.5176 - val_accuracy: 0.8288
Epoch 3/50
1562/1562 [==============================] - 37s 24ms/step - loss: 0.4086 - accuracy: 0.8641 - val_loss: 0.4996 - val_accuracy: 0.8417
.
.
Epoch 24/50
1562/1562 [==============================] - 34s 22ms/step - loss: 0.2330 - accuracy: 0.9199 - val_loss: 0.4132 - val_accuracy: 0.8739
Epoch 25/50
1562/1562 [==============================] - 34s 22ms/step - loss: 0.2255 - accuracy: 0.9215 - val_loss: 0.4111 - val_accuracy: 0.8725
Epoch 26/50
1562/1562 [==============================] - 35s 22ms/step - loss: 0.2240 - accuracy: 0.9233 - val_loss: 0.3952 - val_accuracy: 0.8776
.
.
Epoch 48/50
1562/1562 [==============================] - 34s 22ms/step - loss: 0.1680 - accuracy: 0.9447 - val_loss: 0.4284 - val_accuracy: 0.8801
Epoch 49/50
1562/1562 [==============================] - 33s 21ms/step - loss: 0.1639 - accuracy: 0.9449 - val_loss: 0.4193 - val_accuracy: 0.8801
Epoch 50/50
1562/1562 [==============================] - 33s 21ms/step - loss: 0.1649 - accuracy: 0.9440 - val_loss: 0.4474 - val_accuracy: 0.8801
```

Fit iniziale e **fit con Data Aug. a seguire (50 epoche)**



Qui si notano già delle differenze: la validation accuracy, dopo un incremento iniziale, tende ad appiattirsi intorno al valore di 0.87-88 già dopo sole 25 epoche.
Questo ci fa pensare che il numero di epoche così elevato sia superfluo in quanto si genera overfitting, in quanto l'accuracy tende ad aumentare mentre la validation accuracy si assesta.
Risultato analogo si può notare vedendo i risultati del training con Data Augmentation di 50 epoche:



(Nel codice, è possibile testare il modello facendo stampare un'immagine casuale del test set con relativo predicted label e verificandolo con l'expected label)

Conclusioni

Con questo caso studio abbiamo messo in luce alcuni fattori sul funzionamento ed efficienza di un modello di Machine Learning:

- Avere dei dati migliori crea una grande differenza in termini di qualità dei risultati ottenuti;
- Allenare più a lungo un modello può portare solo a spreco di risorse, generando overfitting che non migliora i risultati.