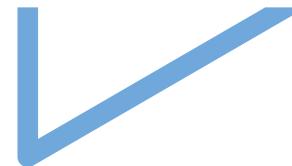




UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES



Progetto di Analisi di Immagini e Video

Professori:

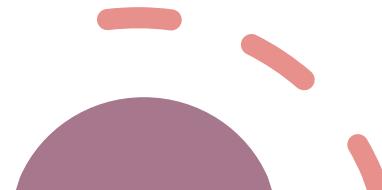
Giuseppe Manco

Francesco Sergio Pisani

Gruppo:

Andrea Zofrea 216640

Giovanni Mirante 214587





Sommario

Preprocessing del dataset

Modelli

Scelta dei parametri

Test



Preprocessing del dataset

Bilanciamento delle classi

Eliminazione delle immagine sfocate e/o omogenee

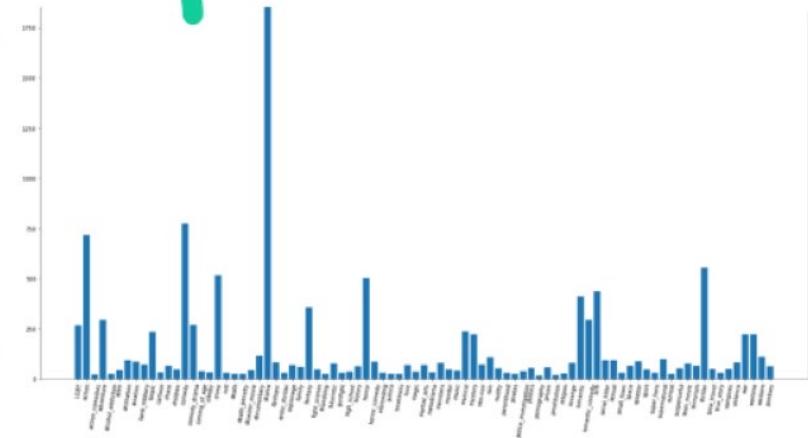
Scelta numero keyframes per trailer

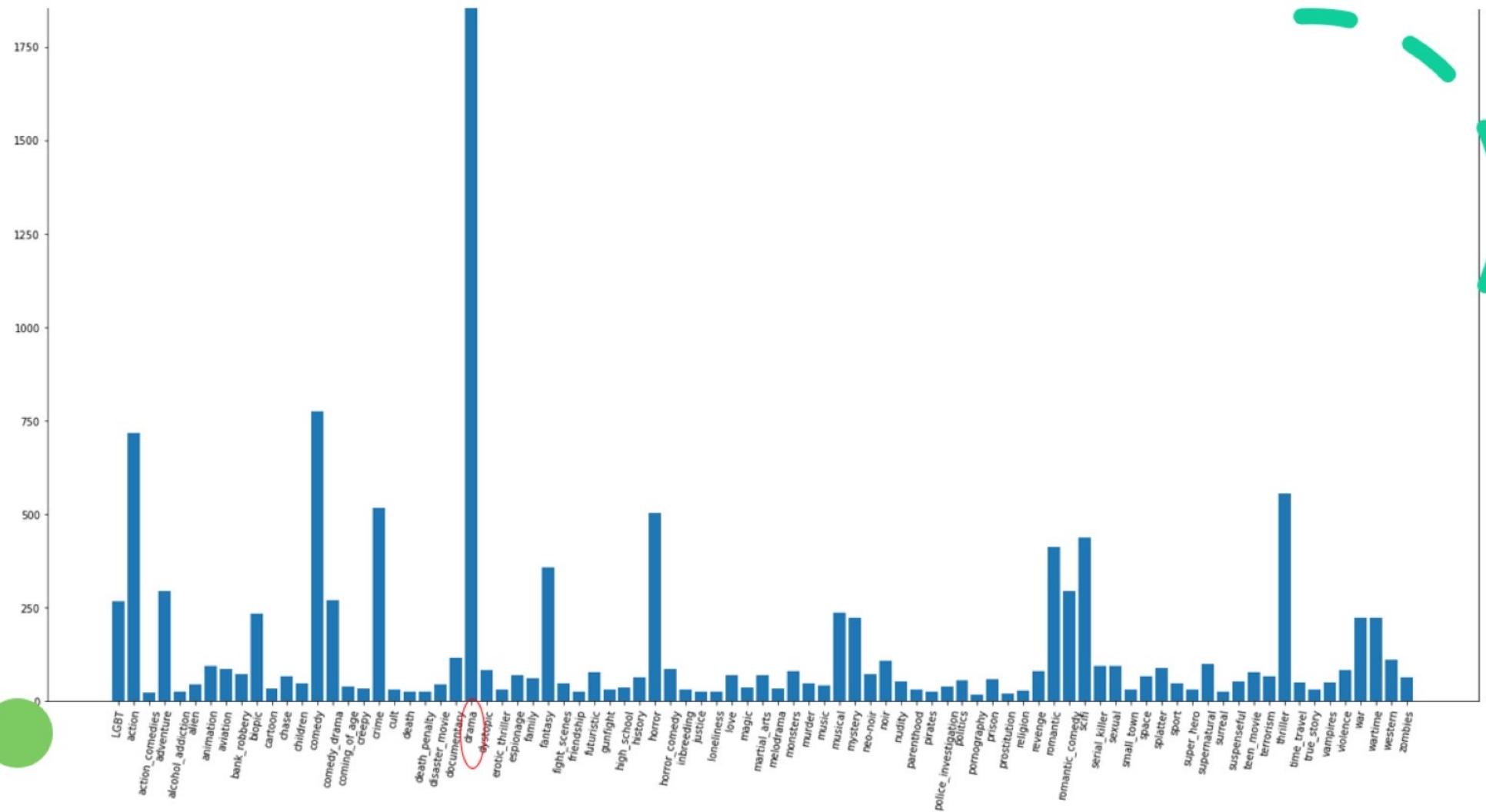
Applicazione di un algoritmo di scelta o replica nei casi in cui il numero di keyframes per trailer sia inferiore o superiore

Bilanciamento delle classi

Come si nota dal grafico abbiamo molto sbilanciamento tra le classi come ad esempio:

- 1.Drama che è assegnata a più di 2000 trailer
- 2.Justince,Cult,death,action_comedies a meno di 10







Soluzione

Undersampling sulla classe maggioritaria (drama) che avrà 755 trailer associati a fronte dei 2000 iniziali

Come avviene l'eliminazione?

- Eliminazione di tutti i trailer che hanno associato solamente il tag "drama".
- In seguito, laddove ci siano ancora tag da eliminare si vanno ad ispezionare i trailer che tra i vari tag loro associati contengono anche il tag "drama". In questo caso, si elimina solamente il tag e non il trailer.





Eliminazione di immagini rumorose

Quali immagini sono state eliminate?

- **immagini pubblicitarie**
- **Immagini sfocate**

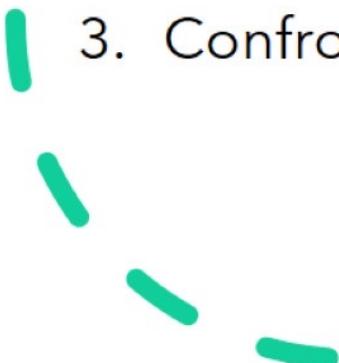




Individuazione immagini pubblicitarie

L'idea alla base dell'algoritmo è di individuare le immagini pubblicitarie riscontrando una certa omogeneità dei canali colore all'interno dell'immagine, attraverso i seguenti step:

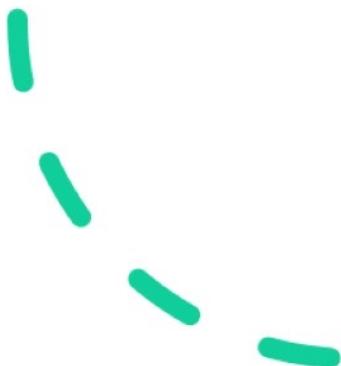
1. Calcolo delle medie dei canali colore
2. Calcolo della differenza delle medie (tra le varie coppie)
3. Confronto del risultato con un valore di soglia

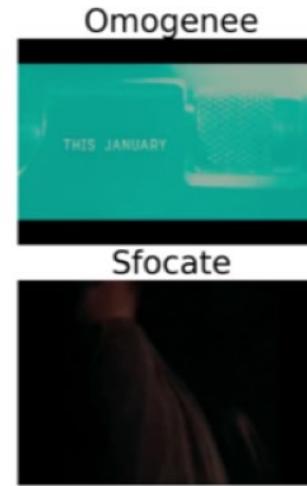
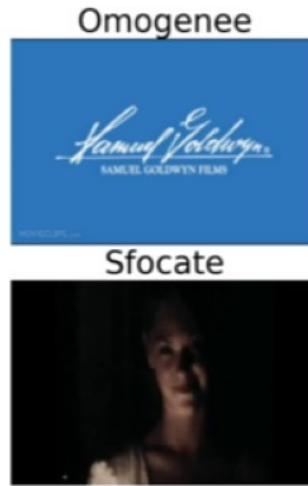
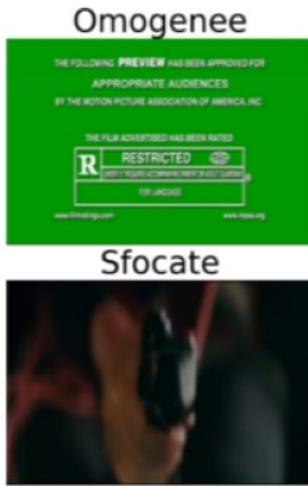
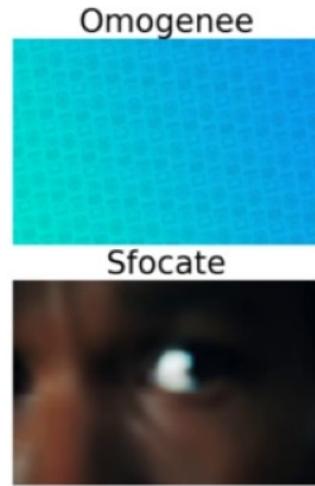




Individuazione immagini sfocate

Le immagini **sfatate**, invece, sono state individuate , utilizzando la varianza del laplaciano , poichè tramite alcune ricerche in letteratura abbiamo notato che essa ci permette di calcolare il valore di sfocatura di un'immagine.





Esempio di immagini sfocate o omogenee

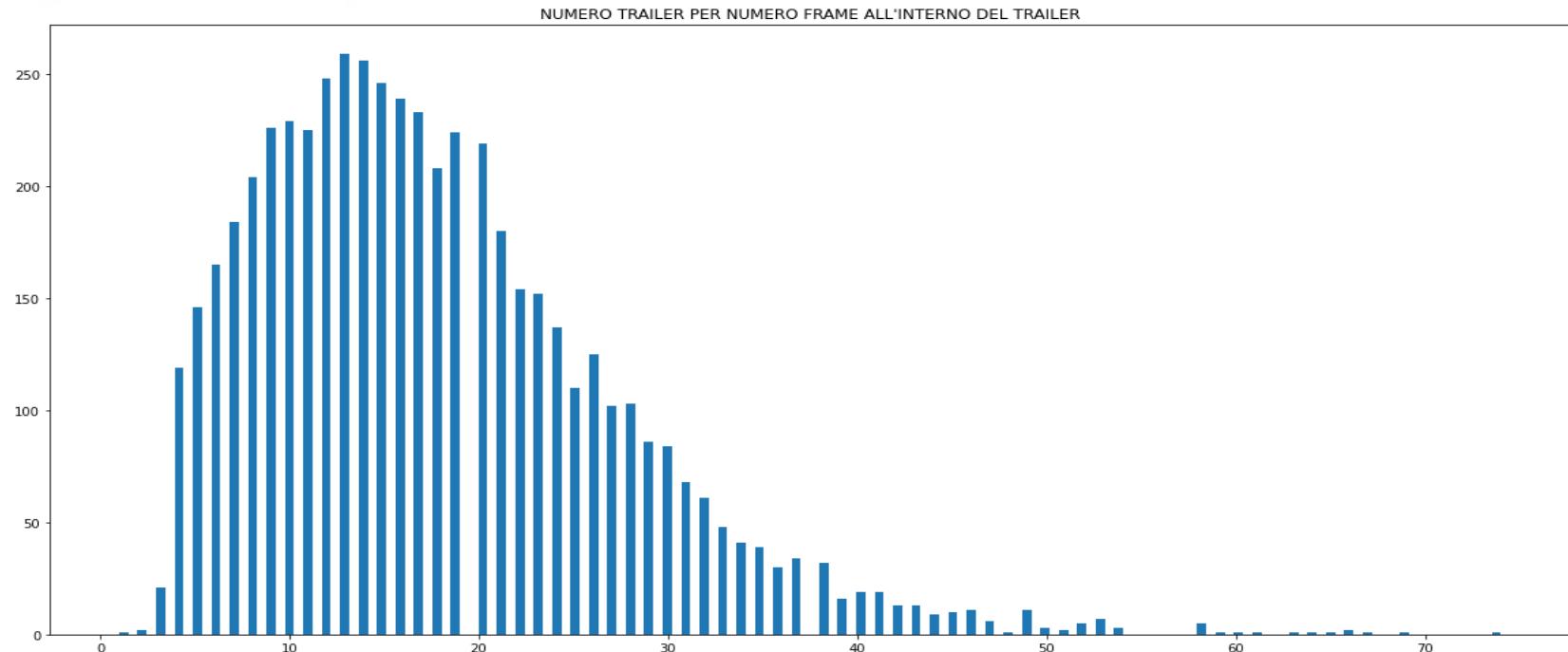
Numero immagini eliminate:

- 2387 immagini omogenee
- 1993 immagini sfocate



Scelta del numero di keyframes

Per scegliere il numero di keyframes da analizzare per ogni singolo trailer, è stata effettuata un'ispezione preliminare sul dataset, dalla quale è emerso un numero medio di Keyframes per trailer pari a 17. Nonostante ciò il numero di Keyframes scelto è stato pari a 10 al fine di velocizzare l'elaborazione in fase di apprendimento da parte della rete.



Casi limite sul numero di keyframes

Numero di
keyframes
inferiori

Numero di
keyframes
superiori



Caso limite keyframes superiori

In questo caso limite, verranno scelti i 10 keyframes migliori, attraverso l'algoritmo implementato nella funzione ritornaImmaginiMigliori() che restituisce 10 immagini migliori tra i keyframes, effettuando la scelta dei 10 frame in maniera consecutiva.

- L'algoritmo prende le prime 10 immagini e le confronta con le successive 10 immagini che sono le precedenti solamente che traslate di uno a destra. Una volta presi i due array d'immagini si calcola per ciascuno il valore di sfocatura e di omogeneità con gli algoritmi descritti precedentemente ma con soglie ancora più profonde.
- Per ciascun array si contano le immagini che superano queste soglie e l'array che contiene più immagini in questo modo viene scartato e si passa al confronto con un altro array di 10 immagini e così via fino a selezionare le 10 migliori.



Esempio





Caso limite keyframes inferiori

In questo caso limite, l'algoritmo utilizzato è **duplicalmmagini()**.

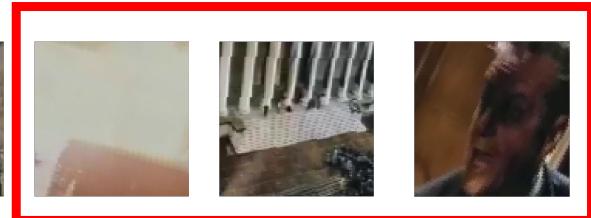
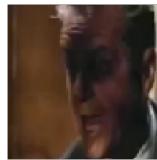
Le immagini già presenti all'interno del trailer, vengono duplicate al fine di avere 10 frame per trailer.

Come avviene la duplicazione?

Utilizzando però lo stessa tecnica descritta precedentemente, cioè replicando le immagini in sequenza in maniera tale da non perdere il senso del film. Inoltre visto che si sta effettuando una duplicazione abbiamo scelto di duplicarli in un certo modo, ovvero, attuando delle trasformazioni, del tipo una rotazione di 30 gradi e un ColorJitter().



Esempio



Modelli





Gestione input Modello

Il modello è stato costruito in maniera tale da passare un input composto da **5 dimensioni** che sono:

BatchSize,NumImgTrailer,Canali,H,W

una volta chiamato il metodo forward() del modello AzGmNet viene eseguito un reshape del tensore passato per portarlo a 4 dimensioni eseguendo questo calcolo

BatchSize*numImgTrailer,canali,H,W

una volta ottenuto il tensore a 4 dimensioni lo passiamo ai livelli che ci estrapolano le featureMap tramite la Vgg16 la ResNet50, chiamati **featureVgg,featureResNet**.

L'output ottenuto da questi livelli è un tensore di 4 dimensioni che viene trasformato in questa maniera.

BatchSize,NumImgTrailer,Feature





Gestione input Modello

Per aggregare tutte le feature prodotte per ciascun trailer abbiamo deciso di eseguire la media delle feature. Producendo così un tensore del seguente tipo

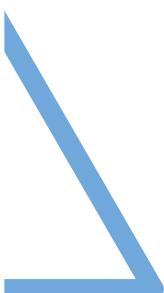
BatchSize, FeatureTotali

sia per le feature prodotte dalla Vgg16 sia per quelle della ResNet50.

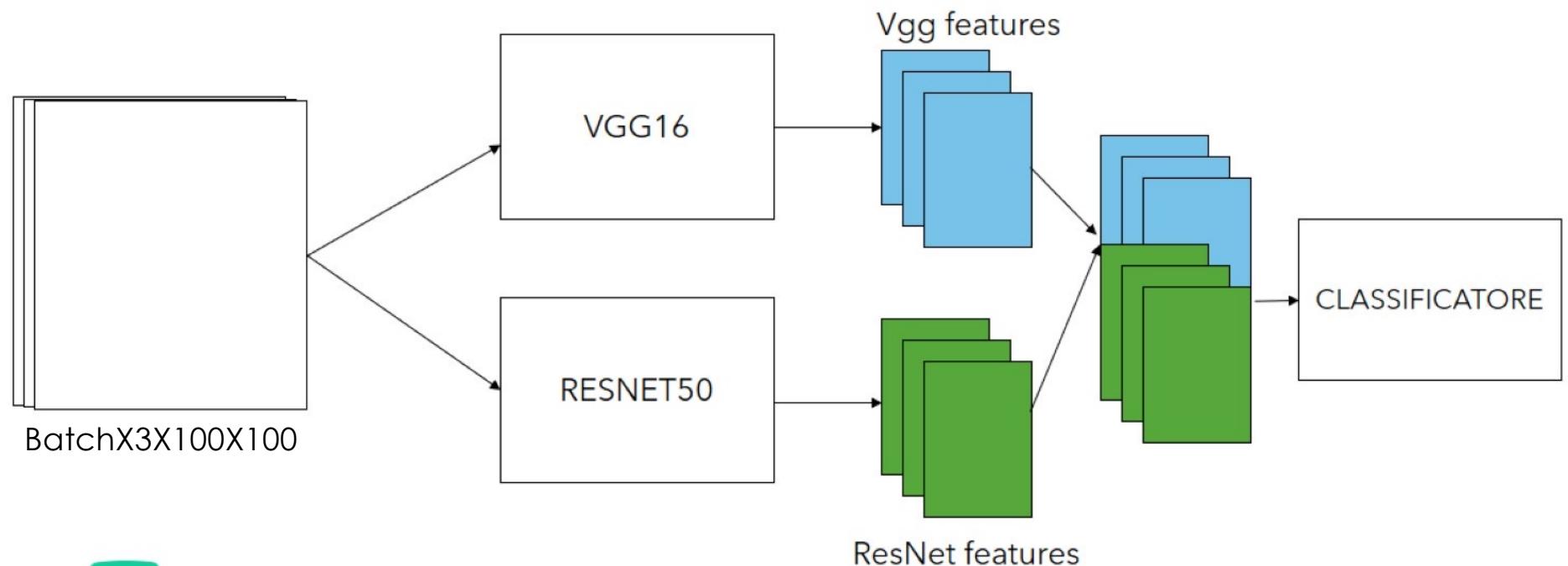
Infine una volta ottenuti i due tensori delle feature li abbiamo concatenati uno con l'altro creando un tensore unico.

Tensore unico che viene passato al livello **classifier** della rete che ci permette di predire le 85 classi/tag. Restituendo un tensori di

BatchSize, 85



Modello Generale AzGmNet





Descrizione del modello

Il modello AzGmNet è composto da:

- 1 livello (feature) di una Vgg16 pre-Addestrata che ci permette di calcolare le featureMap per ogni trailer.
- 2 livello (feature) di una ResNet50 pre-Addestrata che ci permette di calcolare le featureMap per ogni trailer.
- 3 livello fully-connected composto da dei livelli lineare per effettuare la classificazione.

```
VGG(  
    (features): Sequential(  
        (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (1): ReLU(inplace=True)  
        (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (3): ReLU(inplace=True)  
        (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
        (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (6): ReLU(inplace=True)  
        (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (8): ReLU(inplace=True)  
        (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
        (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (11): ReLU(inplace=True)  
        (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (13): ReLU(inplace=True)  
        (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (15): ReLU(inplace=True)  
        (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
        (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (18): ReLU(inplace=True)  
        (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (20): ReLU(inplace=True)  
        (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (22): ReLU(inplace=True)  
        (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
        (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (25): ReLU(inplace=True)  
        (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (27): ReLU(inplace=True)  
        (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (29): ReLU(inplace=True)  
        (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    )  
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))  
    (classifier): Sequential()  
)
```

1° Livello Vgg16



2° Livello ResNet50

```
(1): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
)
(2): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
)
)
(avgnool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Sequential()
```

3° Livello Classifier

```
self.classifier = nn.Sequential(  
    nn.Linear(6656 , 1024),  
    nn.ReLU(True),  
    nn.Linear(1024 , 1024),  
    nn.ReLU(True),  
    nn.Linear(1024 , 85),  
)
```



Principali parametri scelti

Batch size: 29

Loss function: BCEWithLogitsLoss con i pesi, calcolati usando la Inverse of Square Root of Number of Samples (ISNS).

Optimizer: Adam con parametri di default;

Epoche: 6

Operazioni sulle feature: media e concatenazione delle feature estratte da vgg16 e resNet50.



Problemi Riscontrati

Poichè la fase di Train/Test è risultata essere molto pesante in termini computazionali e quindi dal punto di vista anche temporale. Abbiamo deciso di ottimizzare la fase di Train e test per ridurre i tempi di esecuzione.



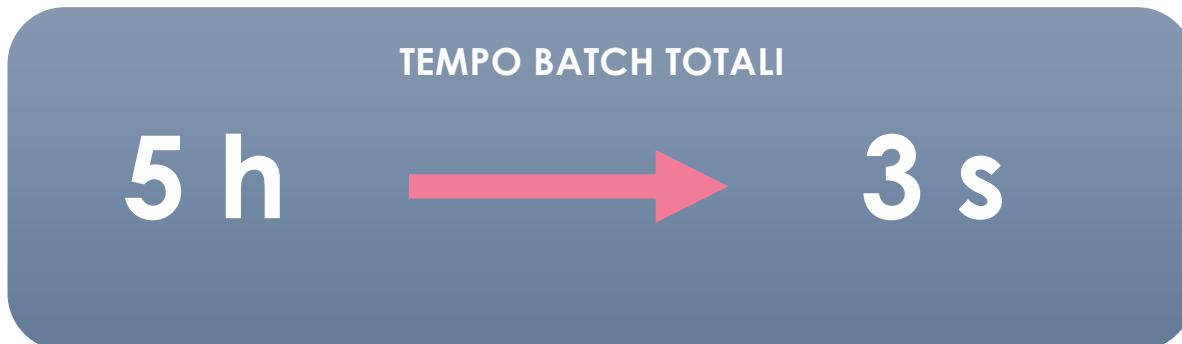
Soluzione 1°



Una prima scelta è stata ridurre i tempi di caricamento di un batch da 29 trailer.

I tempi erano molto lunghi a causa delle operazioni che bisognava fare per ciascun trailer descritte precedentemente (**ritornaImmaginiMigliori()**, **DuplicaImmagini()**, **Trasformazioni**)

Per ridurre questo tempo abbiamo deciso di creare una sorta di cache nel drive nel quale andavamo a salvare i trailer già trasformati e ottimali, in maniera tale da azzerare i tempi di presa di un batch.





Soluzione 2°

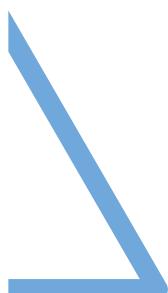
Una seconda scelta è stata ridurre i tempi di addestramento della rete AzGmNet.

I tempi erano molto lunghi a causa della complessità delle reti Vgg16 e ResNet50 per l'estrapolazione delle feature.

Visto che Vgg16 e ResNet50 non vengono addestrate e quindi estrapolano sempre le stesse feature, abbiamo deciso di creare anche qui una sorta di cache per memorizzare le feature per ogni trailer.

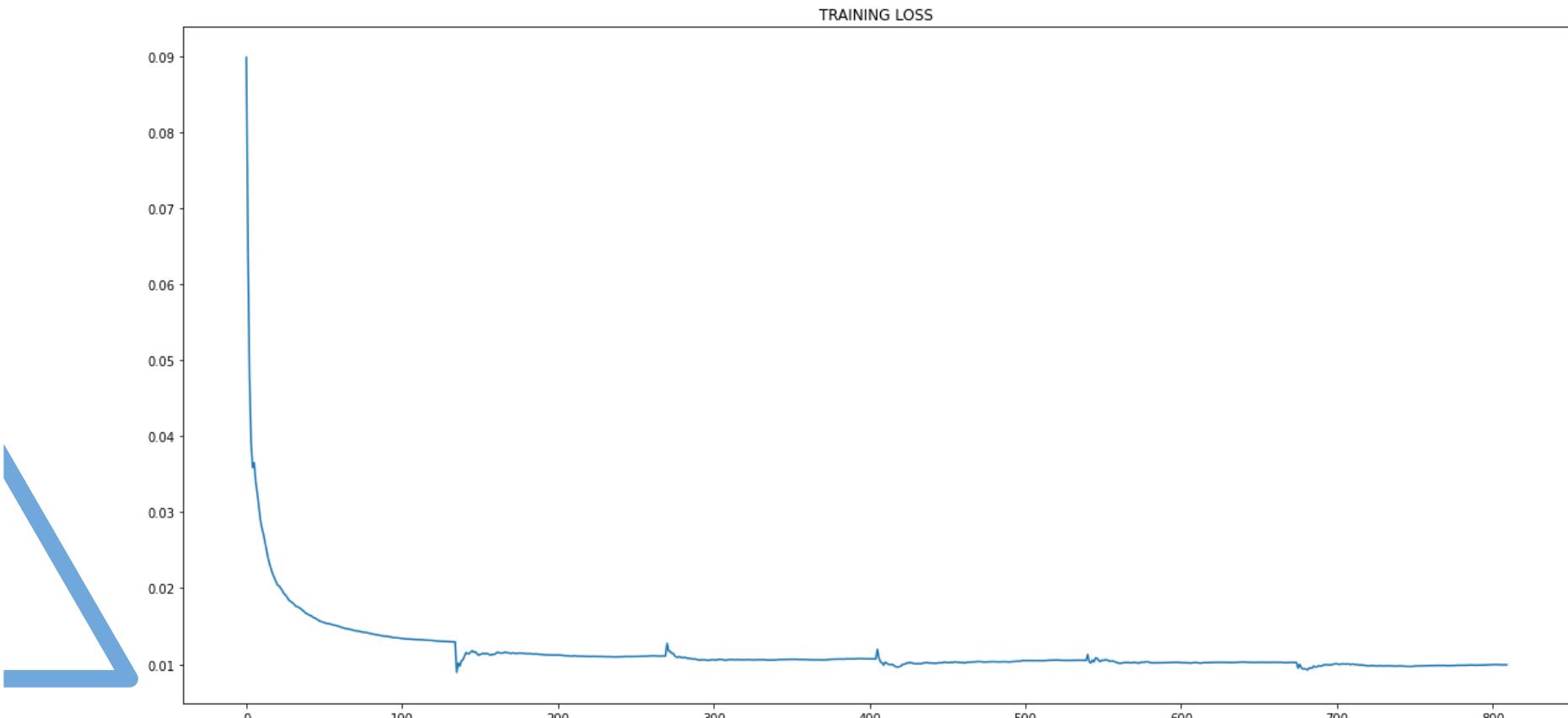
TEMPO TOTALE PER 1 EPOCA

12 h → 30 s





TRAINING LOSS



Report

Report

police_investigation	0.00	0.00	0.00	10					
politics	0.02	0.23	0.04	13					
pornography	0.00	0.00	0.00	4					
prison	0.00	0.00	0.00	15					
prostitution	0.00	0.00	0.00	5	true_story	0.00	0.00	0.00	10
religion	0.00	0.00	0.00	7	vampires	0.00	0.00	0.00	12
revenge	0.00	0.00	0.00	19	violence	0.00	0.00	0.00	20
romantic	0.11	0.80	0.20	99	war	0.08	0.71	0.15	58
romantic_comedy	0.19	0.41	0.26	86	wartime	0.15	0.51	0.23	49
scifi	0.17	0.77	0.27	111	western	0.15	0.42	0.22	33
serial_killer	0.00	0.00	0.00	23	zombies	0.00	0.00	0.00	15
sexual	0.04	0.22	0.07	23					
small_town	0.00	0.00	0.00	9					
space	0.05	0.38	0.09	16					
splatter	0.00	0.00	0.00	25	micro avg	0.17	0.60	0.27	3167
sport	0.00	0.00	0.00	12	macro avg	0.05	0.19	0.07	3167
super_hero	0.00	0.00	0.00	9	weighted avg	0.18	0.60	0.26	3167
supernatural	0.33	0.04	0.07	25	samples avg	0.17	0.64	0.26	3167
surreal	0.00	0.00	0.00	8					
suspenseful	0.00	0.00	0.00	14					
teen_movie	0.00	0.00	0.00	24					
terrorism	0.00	0.00	0.00	14					
thriller	0.14	0.95	0.25	139					
time_travel	0.00	0.00	0.00	12					



Grazie per
l'attenzione

