UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Twitch API and Classification

Progetto di Social Media Management

Studenti:

Mazzamuto Michele (W82000176)

Isgrò Santino (1000000617)

20/02/2020

Introduzione

Sempre più frequente è il fenomeno dello streaming videoludico, ovvero dei videogiocatori che si filmano mentre giocano e condividono il tutto con altre persone che li guardano ed interagiscono in tempo reale. Tra le tante piattaforme di streaming in tempo reale esistenti Twitch.tv è sicuramente la più famosa al mondo, contando ad oggi milioni di utenti attivi.

Nata dall'idea di Justin Kan e Emmett Shear che nel 2007 crearono Justin.tv, trasformandolo poi in Twitch.tv nel 2011, è arrivata ad essere la quarta piattaforma che genera più traffico su internet negli Stati Uniti. Dal 2014, inoltre, Twitch è stato venduto ad Amazon per una cifra pari a 970 milioni di dollari e tale vendita ha portato diversi cambiamenti, come ad esempio la possibilità di collegare Amazon Prime a Twitch.
Le funzioni di un account Twitch di base sono molteplici. Oltre alla ovvia possibilità di sfogliare la galleria di giochi trasmessi in tempo reale, Twitch TV permette di:

- Scegliere chi seguire per ricevere aggiornamenti e sapere sempre quando lo *streamer* è online;
- Supportare i tuoi *streamer* preferiti con donazioni o tramite abbonamento con un costo mensile;
- Utilizzare la chat in tempo reale, comunicando sia con gli altri utenti che guardano lo *streamer* sia con lo *streamer* stesso;
- *Streammare* a tua volta.

Twitch offre delle API che permettono di ottenere numerosi dati interessanti relativi a Live, Streamers, e molto altro. Per effettuare chiamate API, è necessario un ID client, ottenibile dal pannello di sviluppo di Twitch. Le API sono state aggiornate di recente, il nuovo endpoint è /helix

Obiettivi

Gli obiettivi finali del progetto sono 2:

- il primo è quello di utilizzare le API offerte da Twitch per la creazione di un dataset. Tale dataset verrà realizzato dopo aver scaricato clip di ciascun gioco preso in esame e dopo aver estratto da esse alcuni frames.
- Il secondo è quello di allenare un classificatore utilizzando il dataset ottenuto in precedenza al fine di classificare una determinata clip, della quale non si conosce il corretto gioco di appartenenza, in uno dei giochi presi in esame. L'elenco dei giochi presi in esame è il seguente:

```
game=[
('21779', 'League of Legends'),
('509658', 'Just Chatting'),
('33214', 'Fortnite'),
('512710', 'Call of Duty: Modern Warfare'),
('18122', 'World of Warcraft'),
('32982', 'Grand Theft Auto V'),
('27471', 'Minecraft'),
('512804', 'FIFA 20')
]
```

Si può notare come tra i giochi sia presente pure 'Just Chatting' tale categoria non è propriamente un gioco, tuttavia abbiamo deciso di prenderla in esame comunque poiché rappresenta la categoria di maggior successo tra le varie presenti su Twitch. Il successo di tale categoria però potrebbe essere utilizzato impropriamente da alcuni streamer che volendo attirare spettatori, pubblicano stream su 'just chatting' portando come contenuto in realtà un gioco. Potrebbe risultare interessante quindi andare ad individuare tali comportamenti.

API e Creazione dataset

API

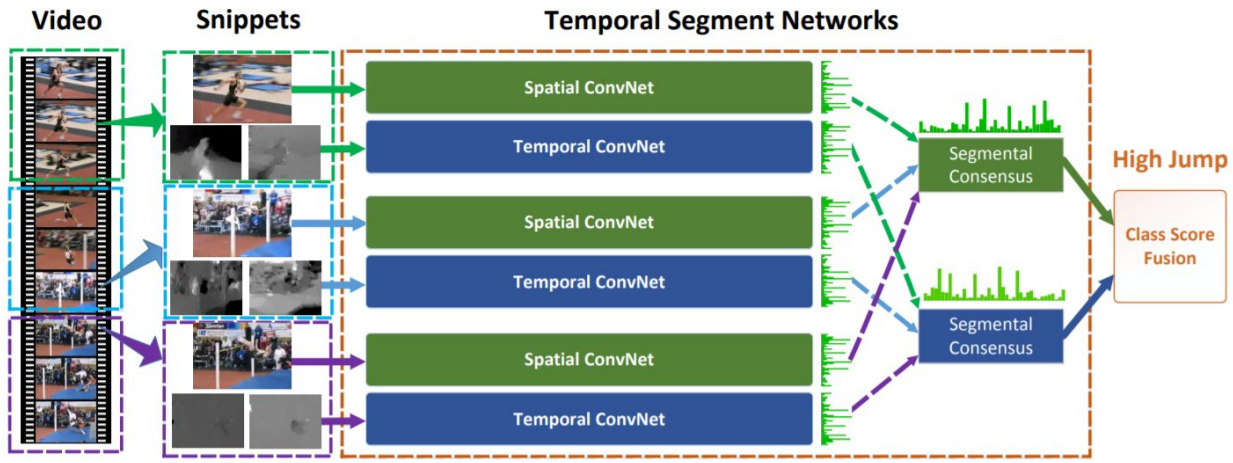
Il primo script utilizzato durante la creazione del dataset ha come scopo quello di individuare per ogni gioco gli streamer migliori, e di essi scaricare un determinato numero di clip. La parte principale dello script la il seguente:

```
for game_id in game:
   print('Game selected:     ', game_id[1])
   create_dir(game_id[1])
   streamers_list=get_top_stream_from_game_id(game_id[0],50)
   cont=0
   print('Streamer found:     ', len(streamers_list))
   for streamer in streamers_list:
       print('Streamer selected: ', streamer[1], '  id: ', streamer[0])
       clip_url_list=get_clip_from_broadcaster_id(streamer[0], game_id[0])[:1]
       print('Going to download: ', len(clip_url_list))
       for url in clip_url_list:
           download_clip(url, game_id[1].replace(' ','-'),  str(cont).zfill(4), streamer[1])
           cont += 1
           pass
```

Le API utilizzate sono quelle per ottenere ID dei giochi partendo dal nome, nickname dei top streamer di ogni gioco e scaricare le clip.

Creazione del Dataset

Dopo aver creato un dataset di clip, ovvero file .mp4 è stato necessario estrarre per ciascuna clip un determinato numero di frame. Per l'estrazione dei frame ci siamo basati sull'idea illustrata in **Temporal Segment Networks: Towards Good Practices for Deep Action Recognition** https://arxiv.org/pdf/1608.00859.pdf



In tale pubblicazione si spiega come a differenza delle immagini fisse, per analizzare azioni contenute nei video gli approcci tradizionali non siano più sufficienti.

L'innovazione introdotta è quella di considerare la correlazione spaziale dei frame, ma aggiungere ad essa una correlazione temporale. Tale correlazione temporale è ottenuta andando a dividere il video in K gruppi e prendendo da ciascun gruppo K frame.

La divisione in K gruppi permette di avere frame estrapolati dalla totalità del video ed evita sbilanciamenti dovuti ad un estrazione random dei frame, che rischierebbe per esempio di essere effettuata solo nella prima parte del video o in quella finale.

Tale ragionamento applicato al nostro dataset ha dato i seguenti risultati nella selezione dei frame:

Un esempio di frame estrapolato(Call Of Duty) è il seguente:

E’ possibile notare alcuni elementi chiave che tendenzialmente lo distinguono da una scena di Just Chatting, ovvero la webcam frontale che generalmente è rivolta sullo streamer ed è posizionata in uno degli angoli dello schermo, la scena di gioco, ed eventuali GUI del gioco.

Nel caso in cui si volesse riprodurre la creazione del dataset, basterebbe eseguire gli script in allegato seguendo il seguente ordine:

1. Get_clip.py → Che provvederà a scaricare le clip
2. Extract_Frame.py → Che effettuerà l'estrazione dei frame dalle clip
3. Split_train_test-py → Che genererà la cartella di train e di test suddividendo i frame totali in K blocchi e selezionando *x* frame per blocco. Infine verrà effettuata una resize per ciascun frame di test e di train.

Metodo

Abbiamo deciso di utilizzare per il nostro task di classificazione la rete SqueezeNet. E' un modello di classificazione delle immagini con training basato sul set di dati ImageNet pubblicato nel 2016. E' un modello molto compatto e veloce: occupa pochi megabyte in memoria e può essere utilizzato su CPU non molto performanti. Le performance di SqueezeNet sono comparabili a quelle di AlexNet, che è però molto più lenta/pesante. AlexNet infatti è formata da 240 MB di parametri e SqueezeNet ne ha solo 5 MB.

Per utilizzare la rete SqueezeNet abbiamo dovuto adattare il nostro dataset in modo da avere frames di dimensione 224 x 244 a colori.

SqueezeNet1_0

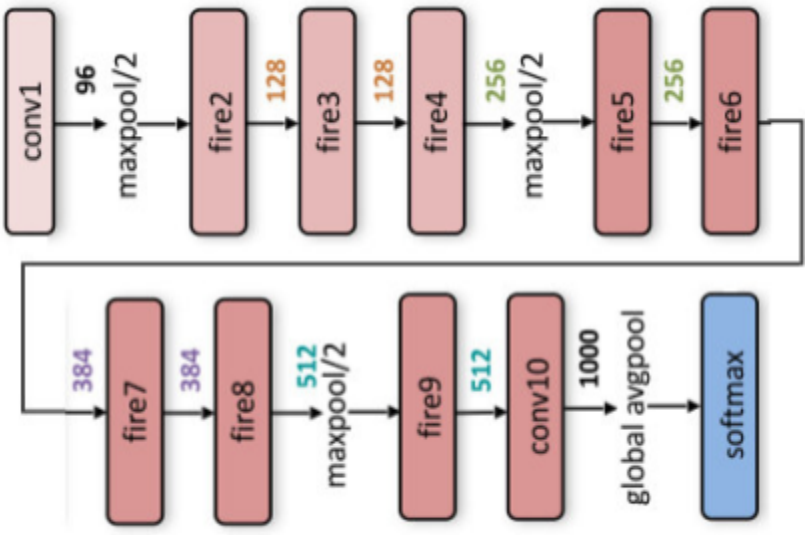


Fig. 10 Struttura SqueezeNet

Esperimenti e Risultati

Abbiamo deciso di suddividere i nostri risultati in base al K, ovvero numero di sottogruppi in cui si divide una clip.

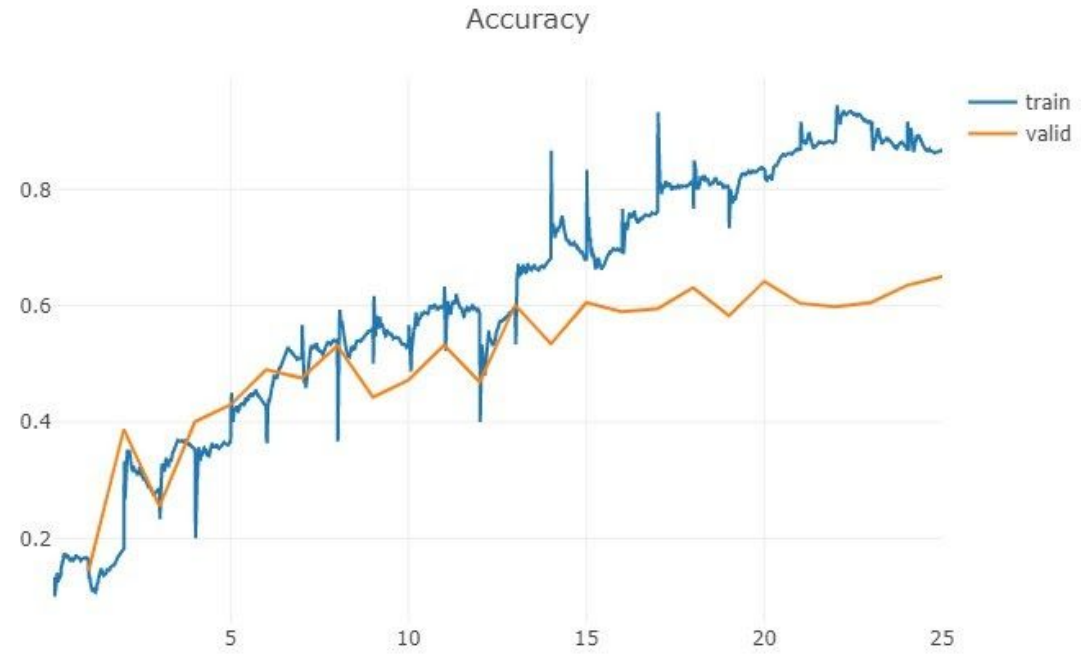
Tale scelta nasce dalla volontà di confrontare come variano i risultati al variare della correlazione temporale introdotta. Pertanto abbiamo effettuato tre test con:

1. Correlazione temporale bassa (K = 3)
2. Correlazione temporale media (K = 6)
3. Correlazione temporale alta(K=12)

Correlazione temporale bassa (K=3)

Utilizzare K=3 significa suddividere ciascuna clip in 3 insiemi. Al loro interno selezionare in maniera randomica un numero di frame, in questo caso è stato scelto sempre il numero 3.

Di seguito i risultati ottenuti :



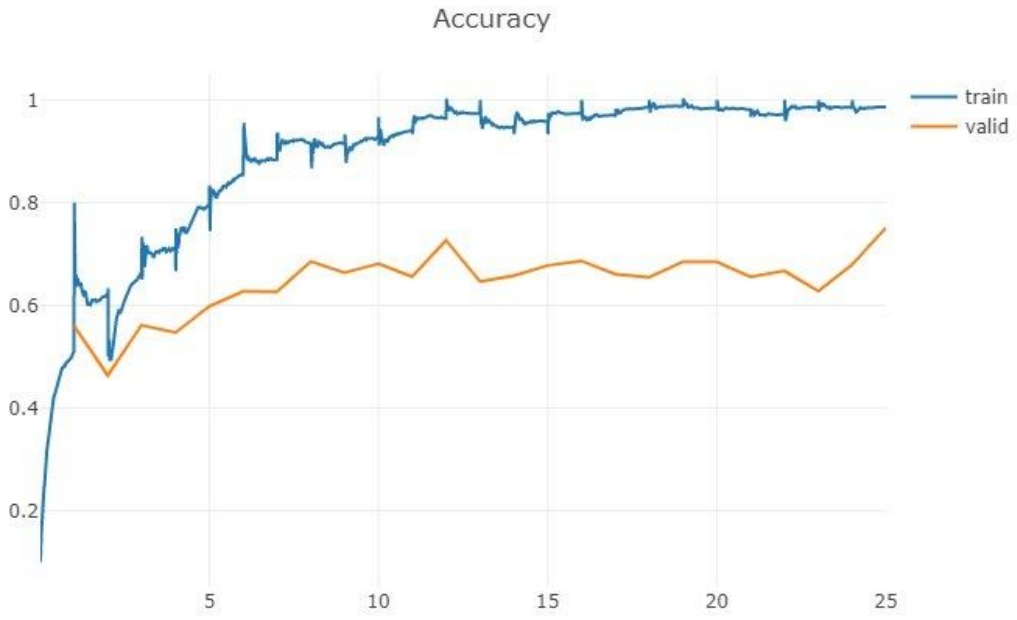
Accuarcy di training: 0.8424

Accuarcy di test: 0.6372

Correlazione temporale media (K=6)

Impostando K=6 si è cercato di introdurre più correlazione temporale all'interno della singola clip. Il numero dei frame scelti all'interno del K-esimo gruppo è rimasto uguale al precedente test, ovvero 3.

Di seguito i risultati ottenuti:



Accuarcy di training: 0.9931

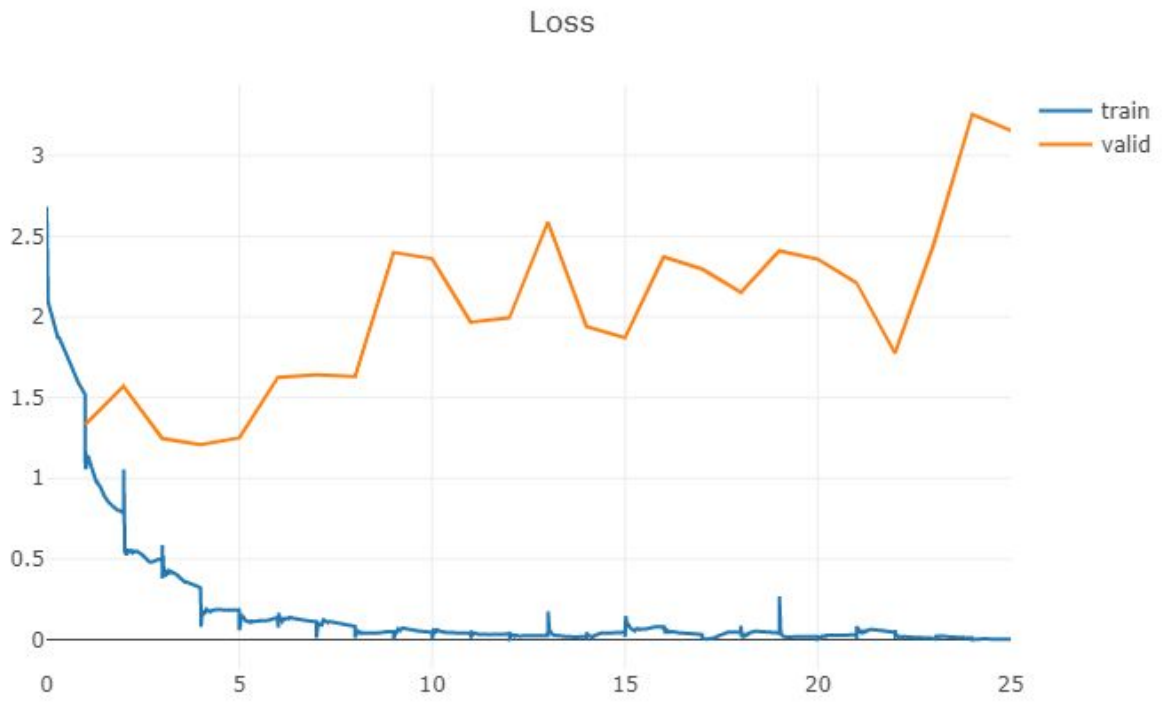
Accuarcy di test: 0.7510

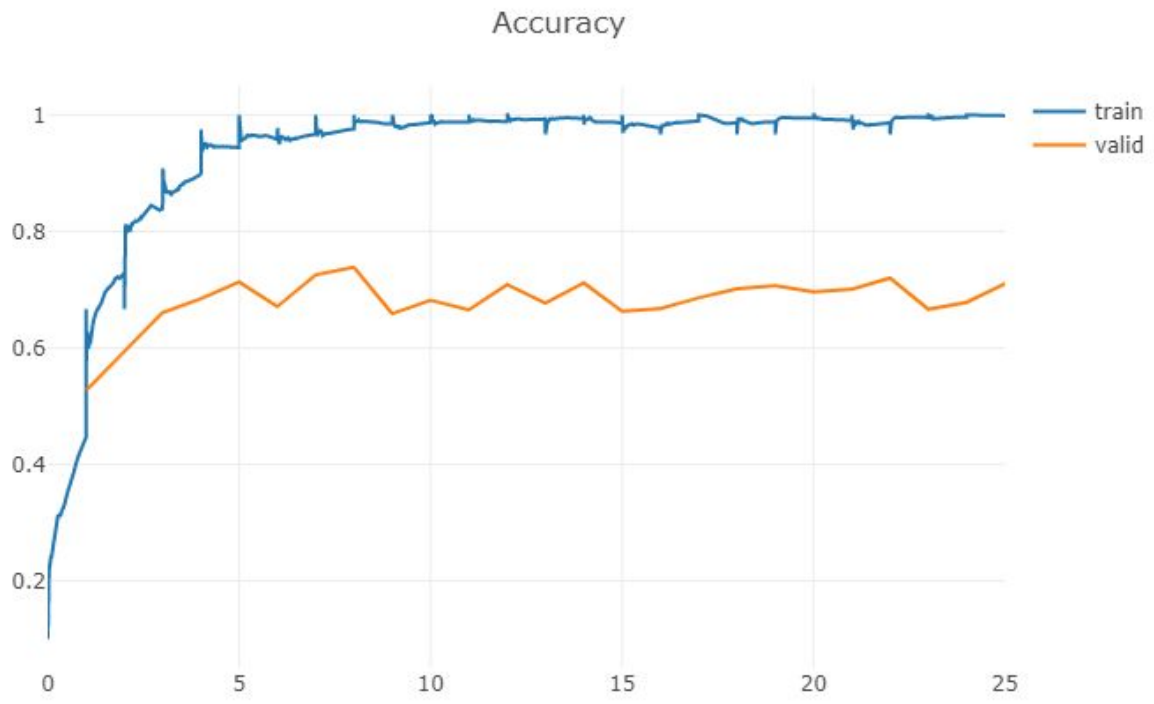
Si può notare come con K=6 i risultati di entrambe le accuracy siano migliorati. Tuttavia il 99% di accuracy di train induce a pensare che il modello faccia overfitting. Questo potrebbe essere dovuto al fatto che 50 clip per gioco per questo determinato task non rappresentino un numero sufficientemente alto.

Correlazione temporale media (K=12)

Abbiamo infine deciso di svolgere un test aumentando sensibilmente sia il numero K che il numero di frame scelti all'interno del k-esimo gruppo, portando tale numero a 5.

Di seguito i risultati ottenuti:





Accuarcy di training: 0.9981

Accuarcy di validation: 0.6788

Riepilogo risultati

Group (K)	*Frame per K*	*Accuracy sul test*
3	**3**	**63%**
6	**3**	**75%**
12	**5**	**68%**

Conclusione

Gli scopi principali del progetto assegnatoci erano due: Sfruttare le API di Twitch per la creazione del dataset ed allenare un classificatore su di esso che fosse in grado di riuscire a distinguere 8 diverse classi di giochi ti tendenza su Twitch.

Anche se i risultati ottenuti potrebbero sembrare buoni/soddisfacenti abbiamo notato di come la selezione delle clip originali che compongono il dataset influenzi tutto il progetto.

Abbiamo infatti testato la rete allenata con frame mai visti dal classificatore provenienti da stream live. Tali frame, seppur appartenenti a categorie presenti del dataset, mostravano scene nuove (sia a livello di colori presenti, sia alla disposizione dei soggetti nella scena) e non presenti nel dataset. Siamo dunque giunti alla conclusione che al fine di migliorare la risposta della rete ad input nuovi sia necessario ampliare sensibilmente il dataset, in modo tale da comprendere quanti più possibili scenari di ciascun gioco.

A tale scopo ad esempio l'estrazione delle clip potrebbe essere fatta con cadenza periodica e in periodi della giornata differenti, per diversificare quanto più possibile le scene catturate. Inoltre sarebbe preferibile al fine di migliorare la qualità del dataset, di estrarre la clip dallo streamer attualmente in live mentre riproduce un determinato contenuto, piuttosto che dalle clip generate dai suoi followers. Infatti, molte di queste clip contengono momenti divertenti ma che la maggior parte delle volte non si riferiscono al gioco effettivo ma piuttosto ad azioni/eventi ripresi dalla camera dello streamer.

Dai risultati ottenuti dalla fase di train, pur avendo un dataset ridotto si evince che dividere la clip in troppe/poche porzioni K risulta essere meno efficiente poiché si avranno "pochi" frame o al contrario troppi frame molto simili all'interno del dataset. Come si evince dai risultati, le prestazioni migliori derivano da una giusta selezione del valore K rappresentante una via di mezzo che non sia ridotta o eccessiva.