Politechnika Krakowska
im. Tadeusza Kościuszki

Ocena czasu potrzebnego na transkodowanie filmu

Dominik Majka (D/133650)

Jakub Saletra (D/136196)

Grzegorz Pach (D/133657)

Sztuczna Inteligencja
Projekt 2023

Wydział Inżynierii Elektrycznej i Komputerowej
Politechnika Krakowska

Spis treści

Wprowadzenie

Kino i filmy stanowią nieodzowny element kultury, zapewniając inspirację, rozrywkę i edukację. Wraz z rozwojem telewizji i powszechnym dostępem do Internetu, filmy stały się bardziej dostępne niż kiedykolwiek. Zachodzi więc potrzeba dostarczania filmów o jak najlepszej jakości, w jak najszybszym czasie.

Współczesne elementy przemysłu wykorzystują kamery i czujniki do szybkiej oceny jakości produktów na liniach produkcyjnych. W tym przypadku liczy się potrzebny czas. Celem niniejszej pracy jest zbadanie za pomocą narzędzi Sztucznej Inteligencji czasu potrzebnego na dekodowanie filmów, wykorzystując przy tym zbiór danych „Online Video Characteristics and Transcoding Time Dataset Data Set".

Wiedza dziedzinowa

Transkodowanie jest to proces konwersji z jednego formatu na inny, w oczekiwanej jakości i rozmiarze pliku wyjściowego. Przykładowy proces transkodowania składa się z:

1. Ekstrakcja danych wejściowych
2. Dekodowanie
3. Przetwarzanie
4. Kodowanie
5. Pakowanie

Podczas tworzenia plików multimedialnych używa się algorytmów kodowania do zamiany danych wideo i audio na specjalny format. Następnie, aby móc odtworzyć taki plik, konieczne jest rozdzielenie tych danych na osobne strumienie wideo i audio. Proces ten polega na wykorzystaniu demuksera, który pobiera plik multimedialny i dzieli go na poszczególne strumienie. Kolejnym krokiem jest przekazanie tych strumieni do odpowiednich dekoderów, które odczytują zakodowane dane i przekształcają je z powrotem na dane wideo i audio, które można odtworzyć.

Bitrate jest to miara ilości danych użytych do zakodowania jednej sekundy filmu. Generalnie rzecz ujmując, wyższy bitrate oznacza lepszą jakość obrazu, ale też rozmiar filmu jest większy.

Klatka, inaczej kadr, jest najmniejszą częścią składową filmu. Film składa się z wyświetlanych jedna po drugiej klatek.

Klatkarz (framerate) jest miarą ilości klatek wyświetlanych w ciągu sekundy.

Rozdzielczość jest to rozmiar pojedynczej klatki. Wyraża się on w formie pikseli, które definiują szerokość i wysokość klatki. Jeszcze do niedawna telewizyjnym standardem była rozdzielczość SD (640:480, proporcje 4:3), jednak wraz z rozwojem cyfryzacji i wyświetlaczy coraz lepszej jakości, zaczęto wdrażać kolejne standardy.

Kodek definiuje standard kodowania dla filmu. Może być on zaimplementowany w formie oprogramowania, czy też układu scalonego. Służy on do kodowania i dekodowania filmu. Popularne kodeki to np. *flv*, *MPEG-4*, *H.264*, *VP8*.

Intra i Inter Prediction są dwoma technikami używanymi w kompresji wideo do redukcji redundancji danych i efektywnego kodowania sekwencji wideo.

1. **Intra Prediction (przewidywanie wewnątrzklatkowe)** - technika wykorzystywana w kompresji wideo, polegającą na przewidywaniu wartości pikseli wewnątrz danej klatki na podstawie sąsiednich pikseli, co prowadzi do redukcji danych poprzez kodowanie różnic między przewidywanymi a rzeczywistymi wartościami pikseli.
2. **Inter Prediction (przewidywanie międzyklatkowe)** - technika wykorzystywana w kompresji wideo, gdzie piksele w danej klatce są przewidywane na podstawie wcześniejszych i/lub następnych klatek w sekwencji, wykorzystując informacje o ruchu obiektów, co prowadzi do kodowania różnic między przewidywanymi a rzeczywistymi wartościami pikseli w celu efektywnego kompresowania danych wideo.

Klatki I, P i B są rodzajami klatek w kompresji wideo. Klatki I są niezależnymi klatkami, klatki P są przewidywanymi klatkami na podstawie poprzednich klatek, a klatki B są przewidywanymi klatkami na podstawie zarówno poprzednich, jak i następnych klatek.

Regresja liniowa jest to jeden ze sposobów nadzorowanego uczenia maszynowego. Na podstawie parametrów wejściowych, dopasowywane są wartości wyjściowe. Regresja ta pozwala na przewidywanie związków pomiędzy danymi. Linia trendu pozwala na stwierdzenie przewidywanej wartości wyjściowej. Technika ta jest jedną z wielu.

Zbiór danych

Abstract: The dataset contains a million randomly sampled video instances listing 10 fundamental video characteristics along with the YouTube video ID.

Data Set Characteristics:	Multivariate	**Number of Instances:**	168286	**Area:**	Computer
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	11	**Date Donated**	2015-05-19
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	57612

Rys. 1. Uproszczona charakterystyka zbioru danych (https://archive.ics.uci.edu/)

Dane zawierają następujące atrybuty:

Nazwa atrybutu	Opis atrybutu
Id	Identyfikator filmu w serwisie YouTube
Duration	Czas trwania filmu
Codec	Kodek wejściowy filmu
Width	Wejściowa szerokość klatki
Height	Wejściowa wysokość klatki
Bitrate	Bitrate
Framerate	Wejściowa ilość klatek na sekundę
I	Ilość klatek typu I
P	Ilość klatek typu P
B	Ilość klatek typu B
Frames	Całkowita ilość klatek
I_size	Rozmiar klatek typu I
P_size	Rozmiar klatek typu P
B_size	Rozmiar klatek typu B
size	Całkowity rozmiar klatek (I, P i B)
O_codec	Kodek wyjściowy
O_bitrate	Bitrate wyjściowy
O_framerate	Wyjściowa ilośc klatek na sekundę
O_width	Wyjściowa szerokość klatki
O_height	Wyjściowa szerokość klatki
Umem	Ilość pamięci na transkodowanie filmu
Utime	Czas potrzebny na transkodowanie filmu

Tab. 1. Charakterystyka zbioru danych

Filmy w zbiorze dane były transkodowane komputerem z procesorem Intel i7-3720QM.

Przygotowanie danych

Z danych wejściowych zostały usunięte niektóre parametry, gdyż w przypadku tego zadania nie wnosiły one wartości dodanej.

Out[146]:

	duration	width	height	bitrate	framerate	frames	size	o_bitrate	o_framerate	o_width	o_h
count	68784.000000	68784.000000	68784.000000	6.878400e+04	68784.000000	68784.000000	6.878400e+04	6.878400e+04	68784.000000	68784.000000	68784.0(
mean	286.413921	624.934171	412.572226	6.937015e+05	23.241321	6641.708377	2.502294e+07	1.395036e+06	21.190862	802.336357	503.8
std	287.257650	463.169069	240.615472	1.095628e+06	7.224848	6153.342453	5.414402e+07	1.749352e+06	6.668703	609.959797	315.9
min	31.080000	176.000000	144.000000	8.384000e+03	5.705752	192.000000	1.918790e+05	5.600000e+04	12.000000	176.000000	144.0(
25%	106.765000	320.000000	240.000000	1.343340e+05	15.000000	2417.000000	2.258222e+06	1.090000e+05	15.000000	320.000000	240.0(
50%	239.141660	480.000000	360.000000	2.911500e+05	25.021740	5628.000000	7.881069e+06	5.390000e+05	24.000000	480.000000	360.0(
75%	379.320000	640.000000	480.000000	6.529670e+05	29.000000	9232.000000	1.977335e+07	3.000000e+06	25.000000	1280.000000	720.0(
max	25844.086000	1920.000000	1080.000000	7.628466e+06	48.000000	310129.000000	8.067111e+08	5.000000e+06	29.970000	1920.000000	1080.0(

Rys. 2. Dane przed normalizacją

Średnie parametrów nie są zbliżone do wartości 0. Z tego powodu autorzy zdecydowali się na przeprowadzenie normalizacji danych.

Przebieg programu

Program zawiera analizę danych i budowę modeli regresji liniowej oraz sieci neuronowej do przewidywania czasu dekodowania filmów. Oto opis najważniejszych fragmentów:

1. Wczytanie danych:
    - W pierwszych linijkach kodu importowane są potrzebne biblioteki, takie jak pandas, numpy, matplotlib i seaborn.
    - Następnie dane są wczytywane z pliku TSV za pomocą funkcji pd.read_table() i przypisywane do zmiennej df.
    - Wywołanie df.head() wyświetla pierwsze pięć wierszy danych.

[83]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_table('/Users/jakub/Downloads/si/transcoding_mesurment.tsv')
df.head()
```

[83]:

	id	duration	codec	width	height	bitrate	framerate	i	p	b	...	p_size	b_size	size	o_codec	o_bitrate	o_framerate	o_width
0	04t6-jw9czg	130.35667	mpeg4	176	144	54590	12.0	27	1537	0	...	825054	0	889537	mpeg4	56000	12.0	176
1	04t6-jw9czg	130.35667	mpeg4	176	144	54590	12.0	27	1537	0	...	825054	0	889537	mpeg4	56000	12.0	320
2	04t6-jw9czg	130.35667	mpeg4	176	144	54590	12.0	27	1537	0	...	825054	0	889537	mpeg4	56000	12.0	480
3	04t6-jw9czg	130.35667	mpeg4	176	144	54590	12.0	27	1537	0	...	825054	0	889537	mpeg4	56000	12.0	640
4	04t6-jw9czg	130.35667	mpeg4	176	144	54590	12.0	27	1537	0	...	825054	0	889537	mpeg4	56000	12.0	1280

5 rows × 22 columns

Rys. 3. Dane zapisane w formacie TSV, wczytane oraz wyświetlone w środowisku Jupyter wykorzystując Python 3.10

2. Przygotowanie danych:
    - W celu przygotowania danych do analizy, usuwane są niepotrzebne kolumny, takie jak 'umem', 'id', 'i', 'p', 'b', 'i_size', 'p_size' i 'b_size', za pomocą metody df.drop().
    - Kopia danych jest tworzona i przypisywana do zmiennej data, a kopia kolumny z czasem dekodowania jest przypisywana do zmiennej target.
    - Dane są konwertowane na tablice numpy za pomocą metody to_numpy().

[84]:
```
# drop umem, id
df = df.drop(['umem', 'id', 'i', 'p', 'b', 'i_size', 'p_size', 'b_size'],axis=1)


df.head()
```

[84]:

	duration	codec	width	height	bitrate	framerate	frames	size	o_codec	o_bitrate	o_framerate	o_width	o_height	utime
0	130.35667	mpeg4	176	144	54590	12.0	1564	889537	mpeg4	56000	12.0	176	144	0.612
1	130.35667	mpeg4	176	144	54590	12.0	1564	889537	mpeg4	56000	12.0	320	240	0.980
2	130.35667	mpeg4	176	144	54590	12.0	1564	889537	mpeg4	56000	12.0	480	360	1.216
3	130.35667	mpeg4	176	144	54590	12.0	1564	889537	mpeg4	56000	12.0	640	480	1.692
4	130.35667	mpeg4	176	144	54590	12.0	1564	889537	mpeg4	56000	12.0	1280	720	3.456

Rys. 4. Wyświetlone dane z pominięciem odpowiednich kolumn.

3. Przygotowanie modelu regresji liniowej:
    - Tworzony jest obiekt LinearRegression().
    - Następnie dane są dzielone na zbiór treningowy i testowy przy użyciu train_test_split().
    - Model regresji liniowej jest trenowany na danych treningowych za pomocą metody fit().
    - Używając mean_squared_error() i r2_score(), wyliczane są miary błędu i oceny modelu na danych testowych.

```
[107]: from sklearn.linear_model import LinearRegression

       linear_regression = LinearRegression()
       linear_regression.fit(train_data, train_target)
```

```
[107]: ▾ LinearRegression
       LinearRegression()
```

```
[108]: from sklearn.metrics import mean_squared_error
       print("Mean squared error of a learned model: %.2f" %
             mean_squared_error(test_target, linear_regression.predict(test_data)))
```

```
Mean squared error of a learned model: 185.61
```

```
[109]: from sklearn.metrics import r2_score
       print('Variance score: %.2f' % r2_score(test_target, linear_regression.predict(test_data)))
```

```
Variance score: 0.30
```

```
[110]: from sklearn.model_selection import cross_val_score
       scores = cross_val_score(LinearRegression(), scaled_data, target, cv=4)
       print(scores)
```

```
[0.28506885 0.2882786  0.28821557 0.27474836]
```

```
[141]: id=2946
       linear_regression_prediction = linear_regression.predict(test_data[id,:].reshape(1,-1))
```

```
[142]: print("Model predicted for video {0} value {1}".format(id, linear_regression_prediction))
```

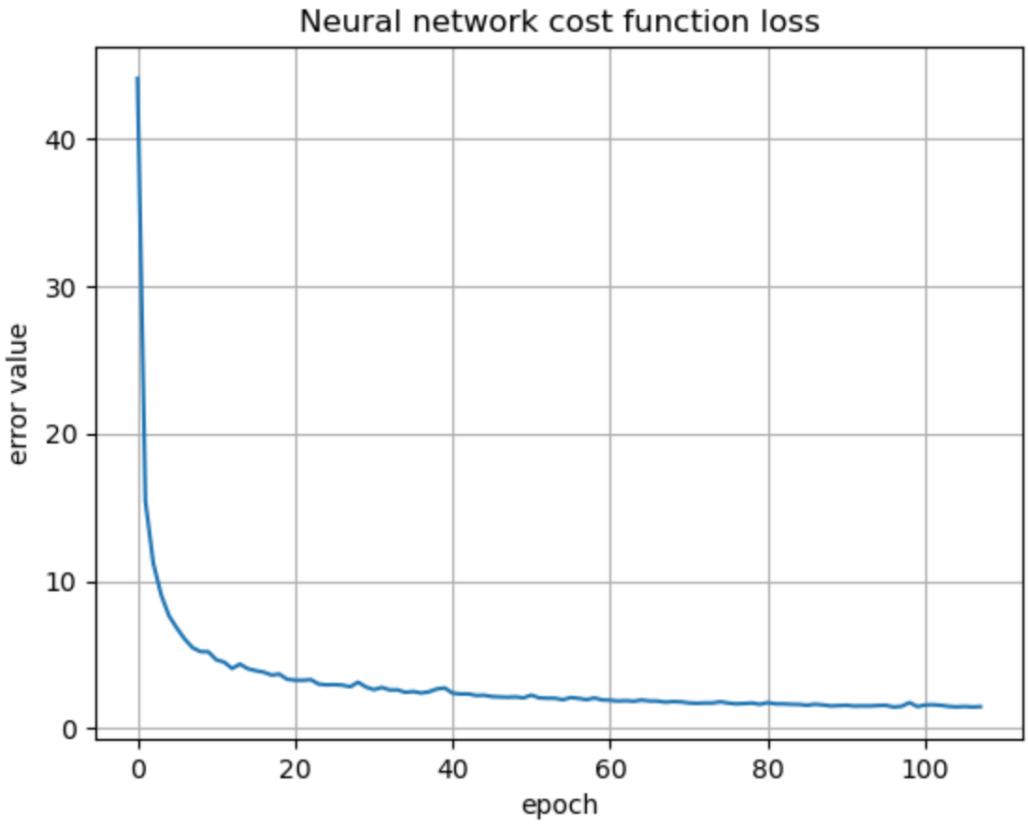
```
Model predicted for video 2946 value [14.55185525]
```

```
[113]: print("Real value for video \"{0}\" is {1}".format(id, test_target[id]))
```

```
Real value for video "5" is 18.189
```

Rys. 5. Przygotowanie modelu regresji liniowej oraz wyświetlenie przewidywalnej i rzeczywistej wartości.

4. Przygotowanie modelu sieci neuronowej:
    - Tworzony jest obiekt MLPRegressor() z odpowiednimi parametrami, takimi jak liczba warstw i neuronów, funkcja aktywacji, solver, momentum itp.
    - Model sieci neuronowej jest trenowany na danych treningowych za pomocą metody fit().
    - Ponownie używając mean_squared_error() i r2_score(), wyliczane są miary błędu i oceny modelu na danych testowych.
5. Wizualizacja wyników:
    - Wykres funkcji kosztu sieci neuronowej jest rysowany za pomocą plt.plot().
    - Wykorzystując plt.title(), plt.xlabel(), plt.ylabel() i plt.grid(), dodawane są tytuły osi i siatka.



Rys. 6. Wartość błędu w zależności od epoki

6. Dobieranie odpowiednich parametrów (grid search):
    - Tworzony jest słownik parameters zawierający różne kombinacje parametrów dla sieci neuronowej.
    - Wykorzystując GridSearchCV(), przeprowadzane jest dobieranie najlepszych parametrów dla modelu (grid search).
    - Model sieci neuronowej jest trenowany na danych treningowych przy użyciu znalezionych optymalnych parametrów.
    - Wykorzystując grid_search.predict(), przewidywany jest czas dekodowania dla konkretnego przypadku testowego.

```
[136]: id=2946
neural_network_prediction = grid_search.predict(test_data[id,:].reshape(1,-1))
print("Model predicted for video {0} value {1}".format(id, neural_network_prediction))
print("Real value for video \"{0}\" is {1}".format(id, test_target[id]))

Model predicted for video 2946 value [20.57568799]
Real value for video "2946" is 21.753
```

Rys. 7. Wyświetlenie przewidywalnej i rzeczywistej wartości na podstawie sieci neuronowej

Ten kod łączy analizę danych, budowę modeli regresji liniowej i sieci neuronowej oraz wizualizację wyników.

Wyniki

W obu przypadkach porównane zostały wybrane losowo dane.

Dla modelu sieci neuronowej

Identyfikator próby	Rzeczywisty czas dekodowania	Przewidziany czas dekodowania
5	18.189	16.47913432
7	21.445	24.00592938
734	1.524	1.9074943
2946	21.753	20.57568799

Tab 2. Wyniki dla modelu sieci neuronowej.

Dla modelu regresji liniowej

Identyfikator próby	Rzeczywisty czas dekodowania	Przewidziany czas dekodowania
5	18.189	14.08537912
7	21.445	13.37187832
734	1.524	2.26575737
2946	21.753	14.55185525

Tab 3. Wyniki dla modelu regresji liniowej.

Wnioski ogólne

1. Model sieci neuronowej wykazuje większą dokładność w przewidywaniu czasów dekodowania w porównaniu do modelu regresji liniowej.
2. Oba modele mają swoje mocne i słabe strony w przewidywaniu czasów dekodowania.
3. Przy dalszej analizie i doskonaleniu modeli, należy skoncentrować się na zidentyfikowaniu czynników, które wpływają na niezgodności między przewidywanymi a rzeczywistymi czasami dekodowania.
4. Jakość danych ma wpływ na modele

Wnioski szczegółowe

Przeprowadzone badanie skupiało się na analizie czasu dekodowania materiałów filmowych za pomocą narzędzi bazujących na Sztucznej Inteligencji. Wykorzystany został zbiór danych o nazwie „Online Video Characteristics and Transcoding Time Dataset Data Set", zawierający szczegółowe informacje dotyczące parametrów filmów oraz czasu ich dekodowania. Wykorzystując te dane, zaimplementowane zostały modele regresji liniowej i sieci neuronowej w celu prognozowania czasów dekodowania.

Analizując wyniki, można wysnuć kilka wniosków ogólnych. Model sieci neuronowej wykazywał większą dokładność w przewidywaniu czasów dekodowania w porównaniu do modelu regresji liniowej. W większości przypadków przewidywane czasy dekodowania były zbliżone do rzeczywistych czasów, co wskazuje na skuteczność modelu sieci neuronowej. W przypadku modelu regresji liniowej, przewidywane czasy dekodowania były mniej precyzyjne, często odbiegające od rzeczywistych czasów. Warto zauważyć, że żaden z modeli nie był w stanie dokładnie przewidzieć czasów dekodowania we wszystkich przypadkach. Istnieją różnice między przewidywanymi a rzeczywistymi czasami, które mogą wynikać z wielu czynników, takich jak różnorodność filmów, jakość danych wejściowych czy inne niewidoczne czynniki wpływające na proces dekodowania. Przewidywanie czasów dekodowania filmów może być zadaniem skomplikowanym i wymaga dalszych badań i doskonalenia modeli. Jednym z możliwych kierunków dalszych prac jest analiza czynników, które wpływają na niezgodności między przewidywanymi a rzeczywistymi czasami dekodowania.

Bibliografia

1. Wykłady i zajęcia z przedmiotu Sztuczna Inteligencja – mgr inż. Kazimierz Kiełkowicz
2. https://archive.ics.uci.edu/ml/datasets/Online+Video+Characteristics+and+Transcoding+Time+Dataset
3. https://cloudinary.com/guides/video-formats/video-transcoding-a-practical-guide
4. https://blog.cyfrowe.pl/standardy-rozdzielczosci-wideo-o-co-w-tym-chodzi/
5. https://www.lepszafotka.pl/co-to-jest-bitrate/
6. https://pandas.pydata.org/docs/reference/api/
7. https://ottverse.com/i-p-b-frames-idr-keyframes-differences-usecases/
8. https://www.sztucznainteligencja.org.pl/kurs/sztuczna-inteligencja-dla-poczatkujacych/jak-maszyna-sie-uczy/algorytm-regresji-liniowej/
9. https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

Dodatki

Kod źródłowy: https://github.com/amtlib/video-transcoding