Klaudia Anna Kiedrowska

Obliczenia naukowe – Raport

1. Analiza obrazów w skali szarości

Funkcja encode przyjmuje jako parametr stopień wielomianu. Mogłaby też przyjmować jako parametr np folder w ktorym znajdują się pliki do pobrania. Za pomocą glob wyszukuje wszystkich plików o rozszerzeniu png które znajdują się w tym samym katalogu co program z funkcją encode. Pobrane pliki png mają dodatkowy wymiar transparency który wynosi dla przykładowych plików 255. Pliki są przerabiane na rgb, a z rgb na gray. Wszystkie szare obrazki są dodawane na listę. Zawartość listy jest łączona w 1 macierz, axis =2 oznacza że ilość obrazków będzie trzecim wymiarem, np. (100,189, 60-obrazków).

Poly_coefs jest macierzą do której będą wrzucane współczynniki. W miejsce każdego piksela powstaje lista współczynników. Współczynniki są obliczane za pomocą np.polyfit. Polyfit zwraca komunikat dla użytkownika o możliwym słabym uwarunkowaniu.

Funkcja encode sprawdza czy w wyniku: poly_coefs.npy znalazły się wartości NaN lub inf które to nie będa mogły zostać odtworzone w dekompresji. Wyświetla komunikat o istnieniu lub braku.

Funkcja decode pobiera jako parametry z terminala plik zapisany przez encode poly_coefs.npy oraz ilosc obrazkow do odtworzenia. Tworzy macierz zer o wymiarach takich jakie ma poly_coefs (szerokość, wysokość) oraz trzeci wymiar to liczba obrazkow. Współczynniki kompresji zostają zamienione na wartości za pomocą np.polyval. Pętlą przechodząc po matrix wyświetlane są obrazki.

W poniższej tabeli znajdują się rozmiary plików inputowych połączonych I zapisanych jako many.npy oraz plików po kompresji, zapisanych jako współczynniki wielomianów dla **60 przykładowych obrazków.**

Many.npy - size	**poly_coefs.npy - size**	**k**
8860	8272	55, poorly conditioned
8860	7532	50, poorly conditioned
8860	6796	45, poorly conditioned
8860	6056	40, poorly conditioned
8860	5316	35, poorly conditioned
8860	4580	30, poorly condition
8860	3840	25, poorly conditioned
8860	3104	20, poorly conditioned
8860	2364	15
8860	1628	10
8860	888	5

W poniższej tabeli znajdują się wyniki błędu średniokwadratowego oraz współczynika determinacji $R^{2.}$

mse	**r2_score**	**k**
0.011256840584678214	0.7505591514049327	50, may be poorly conditioned
0.011488159918998719	0.7454333356295997	40, may be poorly conditioned
0.011738979688682675	0.7398754088095696	35, may be poorly conditioned

0.013147735064955706	0.7086587335908849	20, may be poorly conditioned
0.017152674784462682	0.6199130900249938	10
0.020149464457693275	0.5535070897331329	5

Linijki kodu wyliczającego te wartości znajdują się w pliku mse.py.

Dla przykładowych danych, współczynnik determinacji jest bardzo wysoki, nawet dla dużej kompresji (20 z 60) a współczynnik 0.7. Gdyby inne dane dały podobne wyniki, to można by wnioskować, że jest to dobra metoda kompresji.

Rozmiary dla **60 losowych** macierzy zapisanych jako .png wygenerowanych za pomocą testy2.py

Many.npy - size	**poly_coefs.np**y	**k**
8860	7532	50
8860	6056	40
8860	4580	30

mse	**r2**	**k**
0.02680111008390031	0.43271015885846287	50, may be poorly conditioned
0.027593661934391188	0.4159344875540312	40, may be poorly conditioned
0.029151236633194525	0.38296584182688986	30, may be poorly conditioned
0.031490761708366094	0.3338027587398076	20, may be poorly conditioned
0.038584465663326524	0.18373315899888865	10

Dla losowych danych współczynnik determinacji jest niski 0.43 już dla niewielkiej kompresji 50. Jeśli taka tendencja się utrzyma przy większej liczbie testów, to można się spodziewać, że kompresja nie będzie dobra dla danych zupełnie losowych dużych rozmiarów.

13 losowych macierzy o wymiarach 60x60 zapisanych jako .npy

mse	**r2_score**	**k**
830.2468937118546	0.8454669584369597	10
1659.096620789687	0.6911939701317649	8
2498.4093182340944	0.5349735193948805	6
3326.703989753334	0.3808038430374393	4

Dla danych małych rozmiarów udało się całkiem dobrze odtworzyć dane skopresowane o połowę. Dla k=6 współczynnik determinacji jest nadal bliższy 1 niż 0. O więcej niz połowe model już słabo przewiduje prawidłowe wyniki.

2. Analiza obrazów kolorowych

Program encodeCol3.py postępuje analogicznie do encode.py, tak jak poprzednio many.npy było rozmiarów (wysokość, szerokość, nr obrazka) to teraz jest (wysokość, szerokość, kolor, nr obrazka)
a poly_coefs było rozmiarów (wysokość, szerokość, k+1) a dla kolorów (wysokość, szerokość, kolor, k+1). W przypadku obrazków zapisanych w formacie z kanałem alpha, gdzie wartości = 255

czwarty kolor – transparency, nie wpłynie na zbadanie jakości predykcji więc zostaje pomijany. W pliku spr.py – sprawdzenie czy wszystkie wartości =255. Jeśli od razu jest w formacie rgb, to jest przetwarzany obraz rgb, a jeśli jest w rgba to jest przerabiany na rgb.

Funkcja decodeCol.py tworzy macierz zer o wymiarach takich jak poly_coefs.npy oraz ostatni wymiar to ilosc obrazkow. Teraz poly_coefs ma o 1 wymiar więcej niż w przypadko szarości. Współczynniki kompresji zostają zamienione na wartości za pomocą np.polyval. Pętlą przechodząc po matrix wyświetlane są obrazki.

Funkcja decodeBoth.py rozpoznaje czy współczynniki poly_coefs.npy pochodzą z obrazów szarych czy kolorowych.

Porównanie rozmiarów plików przed kompresją oraz po kompresji dla **60 przykładowych** obrazków kolorowych:

many.npy - size	**poly_coefs.npy - size**	**k**
26580	22592	50
26580	18164	40
26580	15948	35
26580	13736	30
26580	11520	25
26580	9304	20
26580	7088	15
26580	4876	10
26580	2660	5
26580	2216	4
26580	1772	3

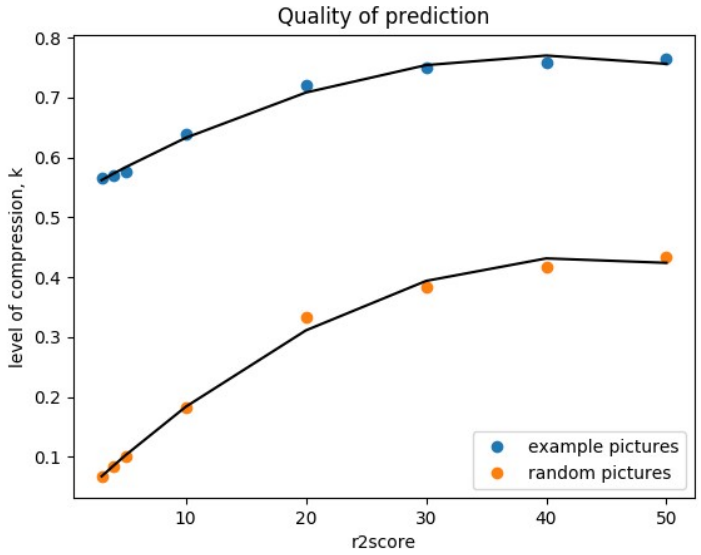
mse	**r2**	**k**
0.011485756590844484	0.7634849436422647	50, without np.clip, may be poorly conditioned
0.011484829151212557	0.7635040415079629	50, with np.clip
0.01176672438318471	0.7576992461381902	40, may be poorly conditioned
0.01219551310215033	0.7488696155231243	30, may be poorly conditioned
0.01355103594692926	0.720956647013706	20, may be poorly conditioned
0.017562946463338322	0.6383432610877803	10
0.020622740476481334	0.5753358877608507	5
0.02088486764142245	0.5699381570411903	4
0.021145776137467372	0.5645655211892809	3

Kompresja na 60 przykładowych, bardzo podobnych od siebie jest bardzo dobra, podobnie jak dla odcieni szarości. Dla dużej kompresji k=20 r2 wynosi 0.72, czyli nawet nieco lepiej niż dla odcieni szarości 0.708. Ze zwględu na komunikat iż wartości pikseli przy wyświetlaniu obrazków wykraczają poza dopuszczalny zakres [0,1] wyniki zostały policzone po zastosowaniu np.clip.

Rozmiany oraz wartości mse, r2 dla **60 losowych** macierzy kolorowych

Many.npy - size	**poly_coefs.npy - size**	**k**
26580	22592	50
26580	18164	40
26580	13736	30
26580	9304	20
26580	4876	10

mse	**r2**	**k**
0.047567265363348925	0.4333664945324407	50, may be poorly conditioned
0.04895965243625143	0.4167800214177937	40, may be poorly conditioned
0.051762999509806444	0.3833858296937803	30, may be poorly conditioned
0.05594558112525338	0.33356184118905186	20, may be poorly conditioned
0.06855617403649544	0.18334121335310327	10
0.07556700137602643	0.09982643399210289	5
0.07697452182802202	0.08305968817143472	4
0.07837080948443788	0.06642675030224154	3



Rys.1 Wykres wpływu k na jakość predykcji

Powyższy wykres ilustruje zależność r2 od k. Widać, że dla obrazów z przykładu wartość r2 jest >0.5 już od bardzo małego k = 5, I rośnie, natomiast dla losowych obrazów nawet przy niewielkiej kompresji k=50 warość jest < 0.5. Można stąd wyciągnąć hipotezę, że badana metoda kompresji bardzo dobrze nadaje sie do kompresji obrazów do siebie podobnych, np. seria zdjęć

jednego zjawiska, w których większość pikseli jest sobie równa lub niewiele się od siebie różni. Natomiast przy całkowicie losowych danych metoda ta nie radzi sobie z prawidłowym przewidzeniem oryginalnych obrazów. Podobne wartości można zaobserwoać dla obrazów w sklai szarości.

Żródła:
https://www.techopedia.com/definition/1945/alpha-channel
http://memobio2015.u-strasbg.fr/conference/FICHIERS/Documentation/doc-numpy-html/reference/generated/numpy.clip.html numpy clip, przycinanie żeby wartości nie wychodziły poza dozwolony zakres
https://www.youtube.com/watch?v=S3N-OHwCSw8&fbclid=IwAR1sztCtJCqM04iSWRkWS4gcUtwwB1g7wb7qdVpDoSfesoylwPIUfMl8ORM – polyfit, polyval

ww.appstate.edu/~marshallst/GLY3455/lectures/10_Curve_Fitting.pdf – fitting high ordered polynomials