Povećanje broja piksela fotografije primenom interpolacionih metoda

Nikola Pantoić RA135/2023

31.1.2026.

1 Definicija problema

Naš problem ogleda se u tome da pri povećanju dimenzija fotografije dolazi do pojave piksela koji nedostaju. Klasičnim skaliranjem slike se broj piksela povećava, ali detalji i oštrina slike često se gube, što rezultira zamućenim ili pikselizovanim izgledom. Cilj ovog projekta je da uz korišćenje interpolacionih metoda popuni pikesele koji fale kako bi se dobila fotografija što većeg kvaliteta. Metode koje ćemo koristiti su: Nearest Neighbor, Bilinear i Bicubic interpolacija. Merenje kvaliteta će se vršiti korišćenjem PSNR (Peak Signal-to-Noise Ratio) i SSIM (Structural Similarity Index). Poreg toga uporedjivaćemo i brzinu izvršavanja svake od metoda. Radićemo sa fotografija u bojama (PPM, JPG, PNG formati) kao i sa fotografijama u sivim tonovima. Biće moguće učitati ih lokalno sa računara i reprezentovati ih kao vektor piksela koji imaju 3 vrednosti (R,G,B) ili jednu vrednost (sivi tonovi).

2 Skup Podataka

Za potrebe ovog projekta koristićemo skup od 10 fotografija koje su slobodne za korišćenje i preuzete sa sajta Unsplash (https://unsplash.com). Fotografije će prvobitno biti i downscale-ovane na dimentzije 256x256 i 512x512 piksela kako bi se testiranje izvodilo na slikama manjih dimenzija. Bitni atributi za praćenje:

- Širina fotografija
- Visina fotografija
- Vrednosti svakog piksela (R,G,B) ili (Sivi ton)

Ovi parametri će biti korišćeni kao ulazni podaci za naše algoritme interpolacije. Za treiranje neuronske mreže koristićemo dataset BSD100 (https://www.kaggle.com/datasets/asilva1691/bsd100) koji sadrži 100 fotografija koje su već podeljenje u Train/Validation skupove (80/20).

3 Metodologija

Rešenje našeg problema predstavlja interpolaciju u 2D prostoru. Biće sprovedeni sledeći koraci:

1. **Treniranje CNN modela:** Korišćenjem PyTorch biblioteke, treniraćemo Konvolucionu Neuronsku Mrežu (CNN) na dataset-u. Mreža će biti trenirana da uči mapiranje iz niske rezolucije fotografija u visoku rezoluciju. Izlazni model će biti sačuvan u .onnx formatu za kasniju upotrebu.

2. **Učitavanje fotografije:** Fotografije će biti učitane sa lokalnog diska i konvertovane u odgovarajući format (niz piksela).

3. **Konverzija u grayscale (opcionalno):** Fotografije će biti konvertovane kako bi svi kanali (R,G,B) imali istu vrednost. Simulira kako bi se algoritam ponašao kada bi se radilo sa samo jednim kanalom.

4. **Skaliranje fotografije:** Svaki piksel u rezultujućoj fotografiji će biti mapiran na odgovarajući piksel u početnoj koristeći faktor skaliranja.

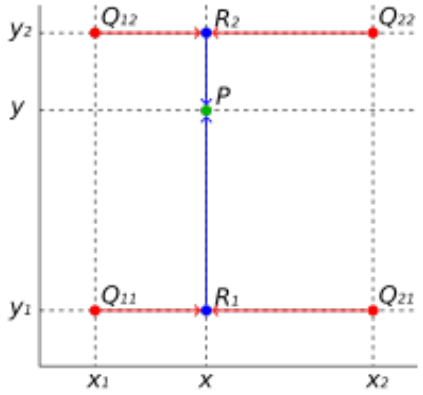
5. **Primena interpolacije:**

   (a) **Bilinear interpolation:** Metoda interpolacije funkcije dve promenljive koristeći linearnu interpolaciju po jednoj promenljivoj, a zatim po drugoj. Linearnom interpolacijom po x-osi dobijamo R_1 i R_2, a zatim linearnom interpolacijom po y-osi dobijamo konačnu vrednost piksela P kao težinski prosek R_1 i R_2.

$$R_1 = f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$R_2 = f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$P = f(x, y) = \frac{y_2 - y}{y_2 - y_1} R_1 + \frac{y - y_1}{y_2 - y_1} R_2$$



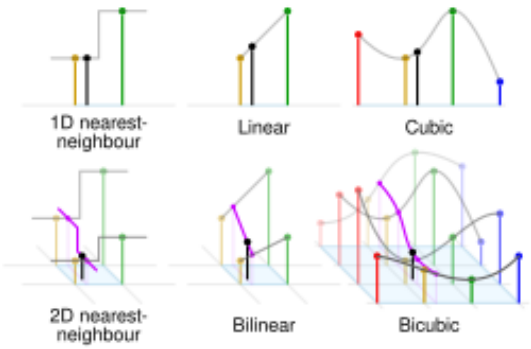
(b) **Bicubic interpolation:** Za razliku od bilinearne interpolacije koja koristi 4 piksela, bikubna interpolacija koristi 16 piksela (4x4). Ova tehnika interpolacije koristi se kada je potrebna veća preciznost, ali po cenu dužeg vremena potrebnog za izračunavanje. Vrednost piksela se računa korišćenjem kubne funkcije koja uzima u obzir vrednosti okolnih piksela.

$$I(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} I_{orig}(x_i, y_j) \cdot w(x - x_i) \cdot w(y - y_j)$$

gde je $w(x)$ kubna interpolaciona funkcija definisana kao:

$$w(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{ako } |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{ako } 1 < |x| < 2 \\ 0 & \text{inače} \end{cases}$$

gde je a obično postavljeno na -0.5 ili -0.75.



(c) **CNN:** Convolutional Neural Network je neuronska mreža korišćena za obradu podataka sa prostornom hijerarhijom, kao što su slike. Sastoji se iz pet slojeva (Layer-a): Input, Convolutional, Max Pooling, Dense i Output sloj. Ulaz mreže će biti fotograije niske rezolucije iz dataset-a. Cilj je da mreža nauči da generiše fotografije visoke rezolucije. Mreža će biti trenirana korišćenjem MSE kao funkcije gubitka i optimizovana korišćenjem ADAM optimizatora. Nakon treninga, mreža će biti testirana na fotografijama koje nisu deo trening skupa kako bi se procenila njena sposobnost generalizacije. Treniranje će se vršiti u Python-u koristeći PyTorch biblioteku, dok će se primena modela na nove fotografije vršiti u C++ koristeći ONNX Runtime biblioteku.

6. **Čuvanje fotografije:** Nakon što su svi pikseli izračunati, nova fotografija će biti sačuvana na lokalnom disku u odgovarajućem formatu.

7. **Merenje kvaliteta i performansi:** Kvalitet rekonstruisane fotografije će biti izmeren korišćenjem PSNR i SSIM metrika. Takodje, vreme potrebno za izvršavanje svake metode će biti zabeleženo i uporedjeno.

4 Način evaluacije

Nakon što su fotografije skalirane, potrebno je izmeriti kvalitet rekonstruisanih fotografije u odnosu na početne. Važno je napomenuti da fotografije moraju biti istih dimenzija i da će do gubitaka doći i zbog prethodnog smanjenja slike. Koristićemo sledeće metrike:

1. **PSNR (Peak Signal-to-Noise Ratio):** Metrika koja meri odnos izmedju maksimalne moguće snage signala i snage šuma koji utiče na vernost njegovog predstavljanja. Viša vrednost PSNR-a ukazuje na bolji kvalitet slike.

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

gde je MAX_I maksimalna vrednost piksela (255 za naše slike), a MSE je srednja kvadratna greška izmedju originalne i rekonstruisane slike:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_{orig}(i,j) - I_{new}(i,j)]^2$$

primer MSE za jedan kanal (Grayscale)

Vrednosti PSNR za 8-bitne slike se tumače na sledeći način:

- PSNR = 48.131 dB: maksimalni kvalitet

- PSNR >35 dB: vrlo dobar kvalitet
- 25 dB<PSNR<35 dB: dobar kvalitet
- PSNR<25 dB: loš kvalitet

2. **SSIM (Structural Similarity Index):** Metrika koja meri sličnost izmedju dve fotografije uzimajući u obzir promene u strukturi, osvetljenju i kontrastu.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

gde su μ_x i μ_y prosečne vrednosti fotografija x i y, σ_x^2 i σ_y^2 su varijanse, a σ_{xy} je kovarijansa izmedju fotografija. C_1 i C_2 se koriste za stabilizaciju formule kada su imenioci mali. Vrednosti SSIM se tumače na sledeći način:

- SSIM = 1: savršena sličnost
- 0.8 <SSIM <1: veoma dobra sličnost
- 0.6 <SSIM <0.8: dobra sličnost
- SSIM <0.6: mala sličnost
- SSIM = 0: nema sličnosti
- SSIM = -1: potpuno različite slike

3. **Uporedjivanje brzine:** Merenje i uporedjivanje vremena potrebnog za izvršavanje svakog algoritma.

5 Tehbnologije

- Programski jezik: C++, Python
- Biblioteke: (C++: STL, stb_image.h, stb_image_write.h, opencv, dnn), (Python: NumPy, OpenCV, PyTorch)
- Kompajler: Visual C++ 2022 00482-90000-00000-AA141
- Interpreter: Python 3.12.6

6 Gotova rešenja

Postoje brojna gotova rešenja za interpolaciju fotografija, ali većina njih je implementirana u okviru većih biblioteka za obradu fotografija. Neki od najpoznatijih alata i biblioteka koje nude ove funkcionalnosti su:

OpenCV: https://opencv.org/

Pillow: https://pillow.readthedocs.io/en/stable/

7 Literatura

- https://en.wikipedia.org/wiki/Channela_$digital_image$
- https://en.wikipedia.org/wiki/Grayscale
- https://medium.com/featurepreneur/understanding-the-concept-of-channels-in-an-image-6d59d4dafaa9
- https://unsplash.com
- https://en.wikipedia.org/wiki/Bilinear_interpolation
- https://en.wikipedia.org/wiki/Bicubic_interpolation
- https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio
- https://www.emergentmind.com/topics/peak-signal-to-noise-ratio-psnr
- https://en.wikipedia.org/wiki/Structural_similarity_index_measure
- https://medium.com/coinmonks/review-srcnn-super-resolution-3cb3a4f67a7c
- https://cs229.stanford.edu/proj2020spr/report/Garber_Grossman_Johnson-Yu.pdf