Programowanie genetyczna – rekonstrukcja zdjęć

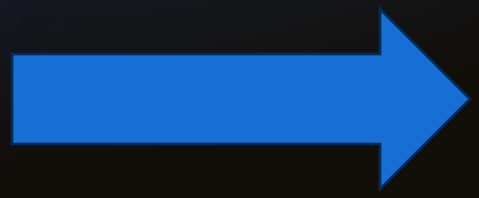
KUBA PIETRZKO

JAN STRYSZEWSKI

MATEUSZ PIĘKOŚ

Cel projektu

Celem tego projektu jest zbliżenie obrazu początkowo złożonego z losowego szumu do oryginalnego obrazu poprzez zastosowanie algorytmu ewolucyjnego. Algorytm ten będzie stopniowo poprawiał podobieństwo obrazu do wzorca, traktując każdy piksel jako osobny "gen".



Architektura algorytmu genetycznego

- **Inicjalizacja**:
  - Wczytanie obrazu docelowego.
  - Utworzenie lub wczytanie populacji osobników, zapisanych jako losowe obrazy.
- **Ewolucja populacji**:
  - **Fitness**: Ocena populacji na podstawie różnic pikseli i kary za szum.
  - **Selecja i krzyżowanie**: Wybór najlepszych osobników, krzyżowanie wybranych rodziców.
  - **Mutacja**: Wprowadzenie losowych zmian dla uniknięcia stagnacji ewolucji.
- **Zapis wyników**:
  - Zapis najlepszych obrazów co kilka generacji oraz całej populacji na koniec.

Stos technologiczny

- **Język programowania**: Python
- **NumPy**:
  - Wykorzystany do wczytywania obrazów i operacji na tablicach.
  - Konwersja obrazów do formatu zgodnego z PyTorch.
- **Pillow (PIL)**:
  - Obsługa obrazów: ładowanie, konwersja do RGB, zmiana rozmiaru, zapis wyników.
  - Konwersja wynikowych tensorów do formatu obrazu (Image.fromarray()).
- **PyTorch**:
  - Używany do operacji tensorowych i funkcji konwolucyjnych.
  - Wbudowane wsparcie dla CUDA, co znacząco przyspiesza obliczenia dla dużych populacji.
- **CUDA**:
  - Operacje równoległe na GPU umożliwiają szybsze przetwarzanie obrazów oraz oceny fitness.
  - PyTorch automatycznie alokuje dane na GPU (wywołania .cuda()), co przyspiesza kluczowe obliczenia.

Problemy

- Obrazy o wysokiej rozdzielczości i dużej liczbie pikseli są bardzo wymagające.
- Redukcja szumu jest trudna do osiągnięcia.
- Zastosowaliśmy karę za szum w funkcji oceniającej osobniki, co pomaga w uzyskaniu lepszych wyników.

Wymagania

- **Sprzęt**:
  - GPU z obsługą CUDA (dla przyspieszenia obliczeń na kartach graficznych NVIDIA).
  - Procesor o wielordzeniowej architekturze, aby poprawić wydajność w operacjach równoległych.
- **Oprogramowanie**:
  - Python 3.8 lub nowszy.
  - System operacyjny z obsługą CUDA (Linux, Windows z odpowiednim sterownikiem).
  - Zainstalowane biblioteki: torch, numpy, Pillow, CUDA Toolkit.

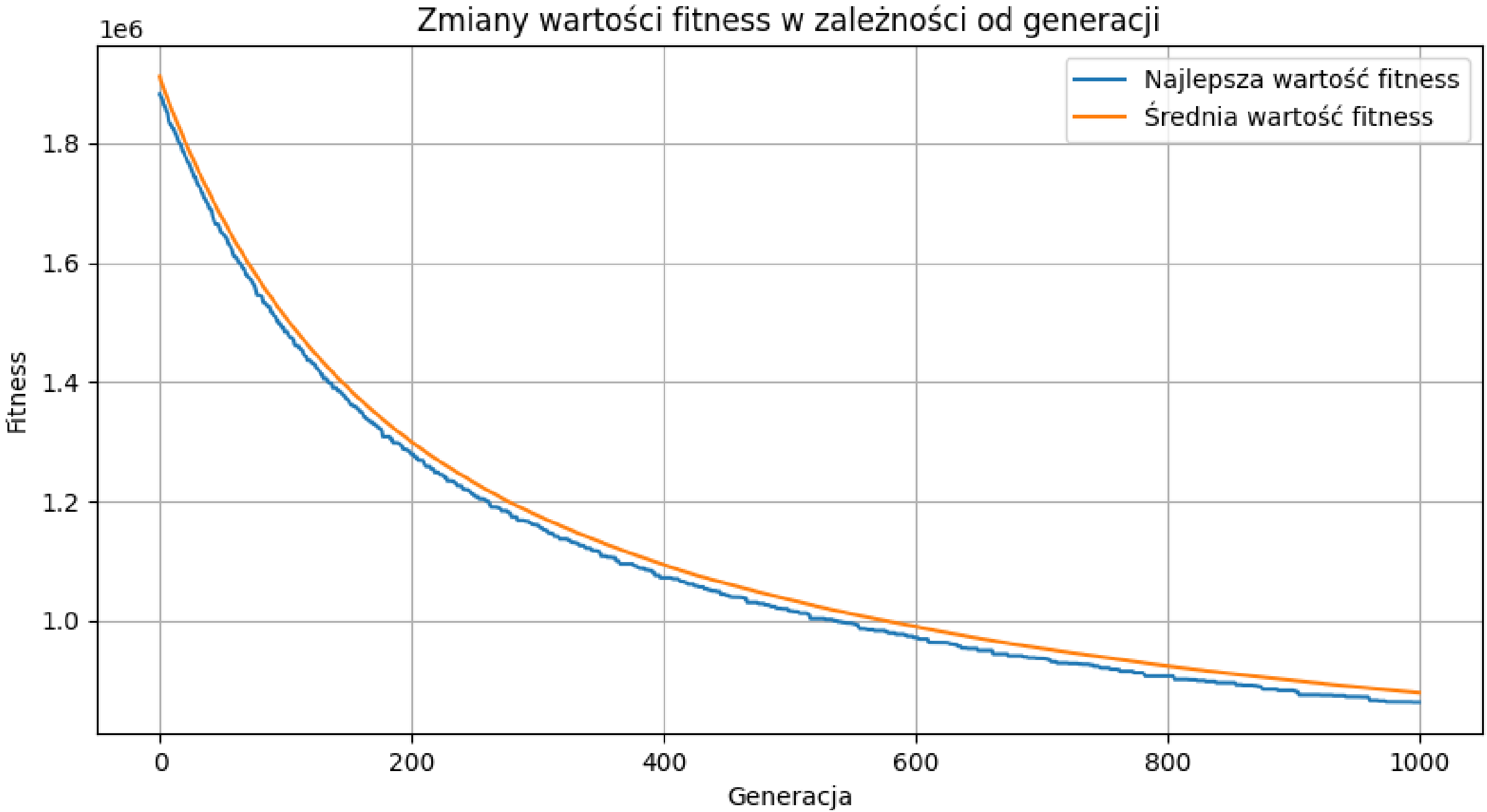
Akceleracja na GPU

- **PyTorch**:
  - Używany do operacji tensorowych i funkcji konwolucyjnych.
  - Wbudowane wsparcie dla CUDA, co znacząco przyspiesza obliczenia dla dużych populacji.
- **CUDA**:
  - Operacje równoległe na GPU umożliwiają szybsze przetwarzanie obrazów oraz oceny fitness.
  - PyTorch automatycznie alokuje dane na GPU (wywołania .cuda()), co przyspiesza kluczowe obliczenia.

Eksperymenty

Eksperymenty zostały przeprowadzone na różnych rozdzielczościach oraz różnych przestrzeniach barw.

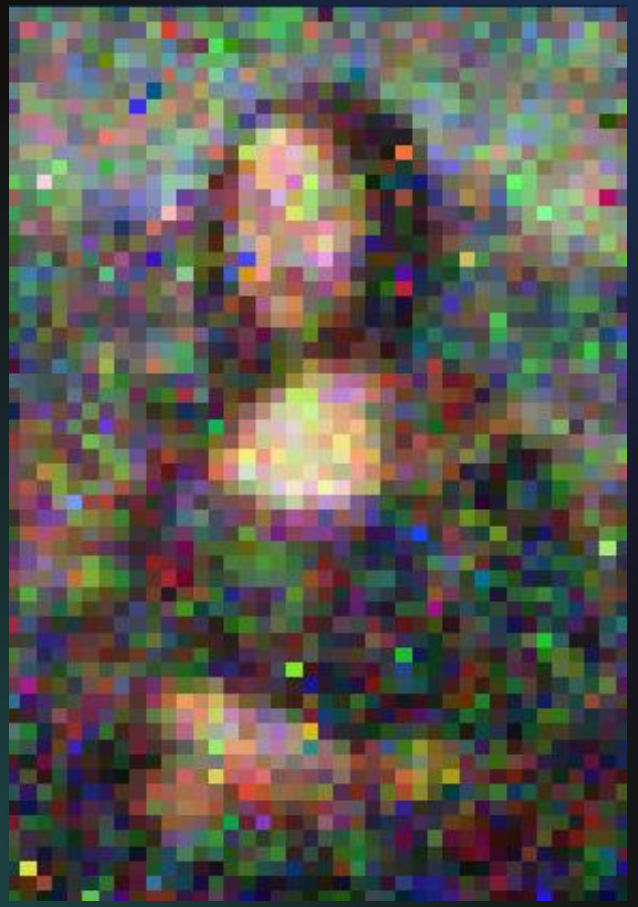
- Przetrzenie barw: RGB, HSV, monochromatyczny, redukcja przestrzeni barw do kolorów orginalnie występujących
- Rozdzielczość: 40x60, 66x100, 100x150
- Populacja od 1000 do 10000


Zmiany wartości fitness w zależności od generacji
1e6
1.8
1.6
1.4
1.2
1.0
Fitness
0
200
400
600
800
1000
Generacja
Najlepsza wartość fitness
Średnia wartość fitness

Skala Szarości

- Ilość generacji: 3000
- Wielkość populacji: 5000
- Rozdzielczość: 100x150

RGB



Ilość generacji: 20000
Wielkość populacji: 2000
Rozdzielczość: 40x60

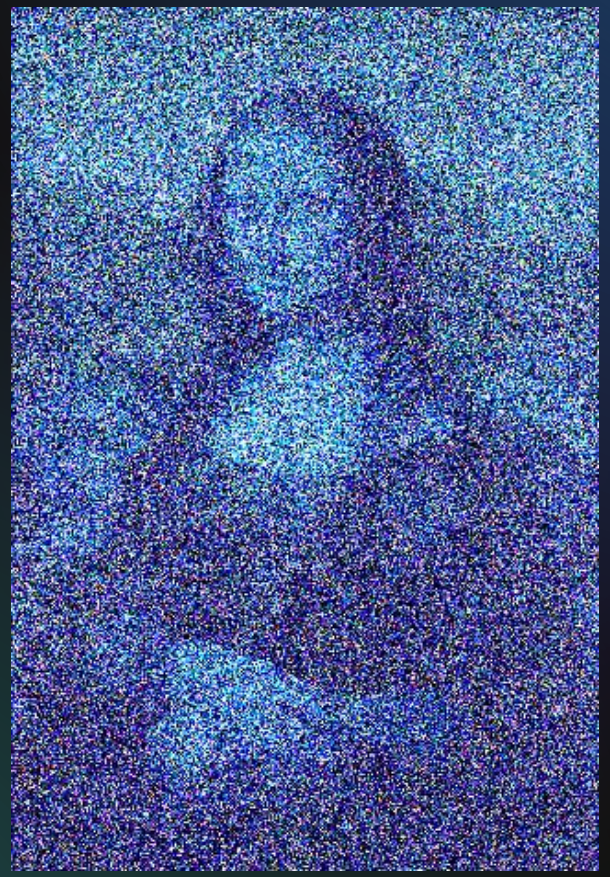
Ilość generacji: 2000
Wielkość populacji: 4000
Rozdzielczość: 100x150

Unikalne kolory ze zdjęcia wejściowego

Ilość generacji: 2000
Wielkość populacji: 2000
Rozdzielczość: 66x100

Inne obrazy

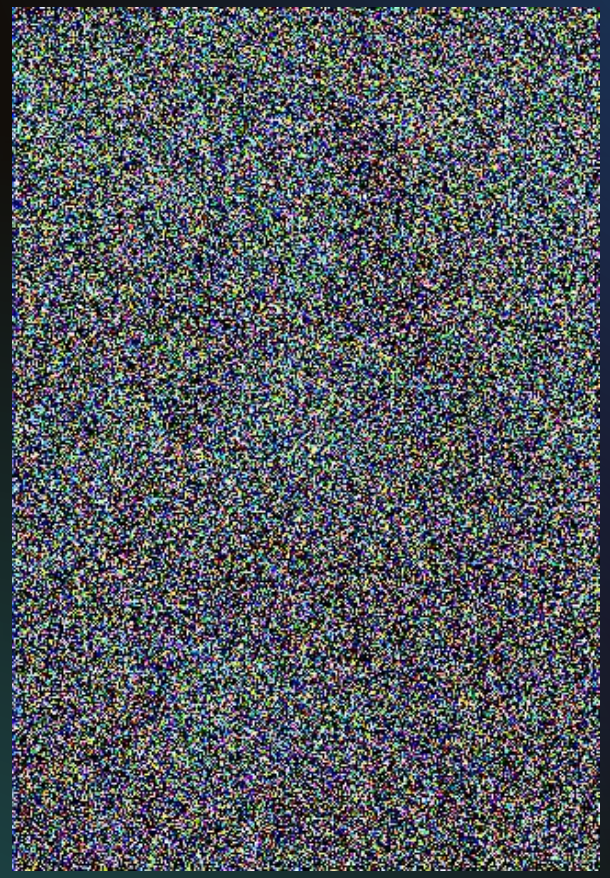
Dalsze eksperymenty

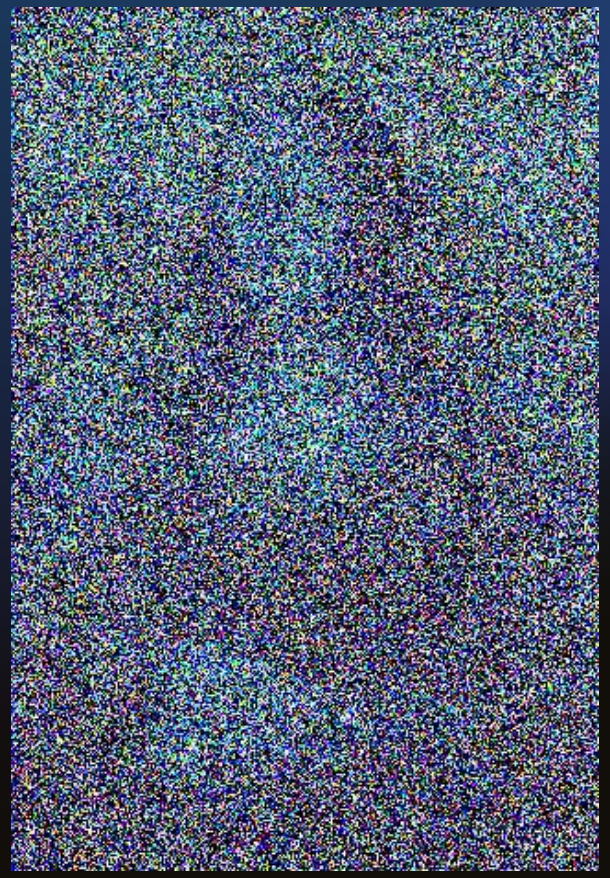


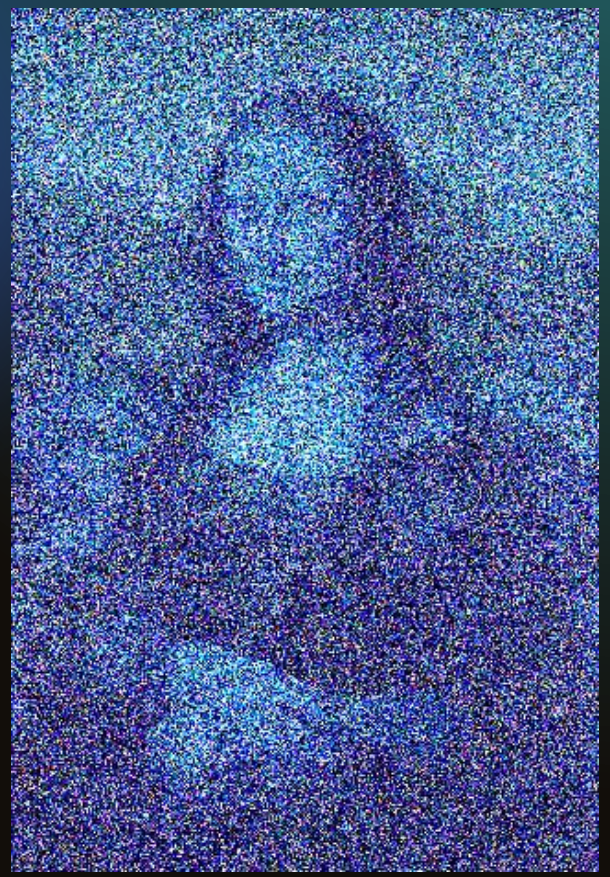
Konwersja kolorów do przestrzeni LAB
L - jasność, normalizacja [0,1]
A, b- kolory

Ilość generacji: 5000
Wielkość populacji: 5000
Rozdzielczość: 320x480

Obraz przesunięty kolorystyczne w kierunku koloru niebieskiego, czyli kanału B
Możliwe przyczyny:
Konwersja RGB -> LAB (zbyt mała kara na kanale B)







SSIM- funkcja straty

Ocenia jakośc obrazu na podstawie percepcyjnej podmności strukturalnej, skupia się na jasności, kontraście i strukturze.

Miara	Uwzględnienie jasności	Uwzględnienie kontrastu	Uwzględnienie struktury	Skala wyniku
MSE	Nie	Nie	Nie	[0, inf)
SSIM	Tak	Tak	Tak	[-1,1]

Implementacja tej metryki znajduję się w bibliotece skimage

Literatura

- Algorytmy Genetyczne. Kompendium. Tom 1 , Gwiazda Tomasz D.
- Kramer, O., & Kramer, O. (2017). Genetic algorithms (pp. 11-19). Springer International Publishing.
- Forrest, S. (1996). Genetic algorithms. ACM computing surveys (CSUR), 28(1), 77-80.
- https://rogerjohansson.blog/2008/12/07/genetic-programming-evolution-of-mona-lisa/
- https://medium.com/@sebastian.charmot/genetic-algorithm-for-image-recreation-4ca546454aaa