

Sztuczna Inteligencja w Grafice Komputerowej - Projekt nr 1

Błażej Ejzak, Paweł Dombrzalski
313220, 318647

10 kwietnia 2025

Streszczenie

Niniejszy dokument prezentuje wyniki rozwiązania zadań Odszumianie (2.2) oraz Deblurowanie (2.3).

Spis treści

1	Odszumianie	2
1.1	Warstwy autoenkodera	2
1.2	Funkcja straty	2
1.3	Wyniki	3
1.4	Wnioski	11
2	Deblurowanie	11
2.1	Architektura modelu UNet	11
2.2	Funkcja straty	12
2.3	Trening	12
2.4	Wyniki	13
2.5	Wnioski	16

1 Odszumianie

W zadaniu został wykorzystany konwolucyjny autoenkoder.

1.1 Warstwy autoenkodera

Zbiór danych złożony jest tylko z 800 zdjęć o względnie niewielkim rozmiarze 256x256 pikseli stąd aby uchwycić modelem istotę zjawiska odszumiania nie należy używać zbyt wielu warstw. Koder składa się z następujących warstw:

- Warstwa konwolucyjna z rozmiarem jądra równym 3, krokiem równym 1 oraz dopełnieniem równym 1 z przejściem do 16 kanałów.
- Funkcja aktywacji ReLU.
- Pooling uśredniający 2x2.
- Warstwa konwolucyjna z rozmiarem jądra równym 3, krokiem równym 1 oraz dopełnieniem równym 1 z przejściem do 64 kanałów.
- Funkcja aktywacji ReLU.

Dekoder złożony jest z następujących warstw:

- Warstwa konwolucji transponowanej z rozmiarem jądra równym 3, krokiem równym 2 i dopełnieniem równym 1 z przejściem do 16 kanałów.
- Funkcja aktywacji ReLU.
- Warstwa konwolucji transponowanej z rozmiarem jądra równym 3, krokiem równym 1 i dopełnieniem równym 1 z przejściem do 3 kanałów.
- Funkcja aktywacji sigmoidalna.

1.2 Funkcja straty

Funkcja straty, która łączy błąd średniokwadratowy (**MSE**) i wskaźnik podobieństwa strukturalnego (**SSIM**), jest definiowana jako:

$$L = \alpha \cdot \text{MSE} + \beta \cdot (1 - \text{SSIM}) \quad (1)$$

gdzie:

- α, β – wagi regulujące wpływ poszczególnych składników funkcji straty.
- **MSE** (Mean Squared Error) – błąd średniokwadratowy, określony wzorem:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2)$$

gdzie y_i to rzeczywista wartość, a \hat{y}_i to przewidywana wartość modelu.

- **SSIM** (Structural Similarity Index Measure) – miara podobieństwa strukturalnego między dwoma obrazami, określona jako:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (3)$$

gdzie:

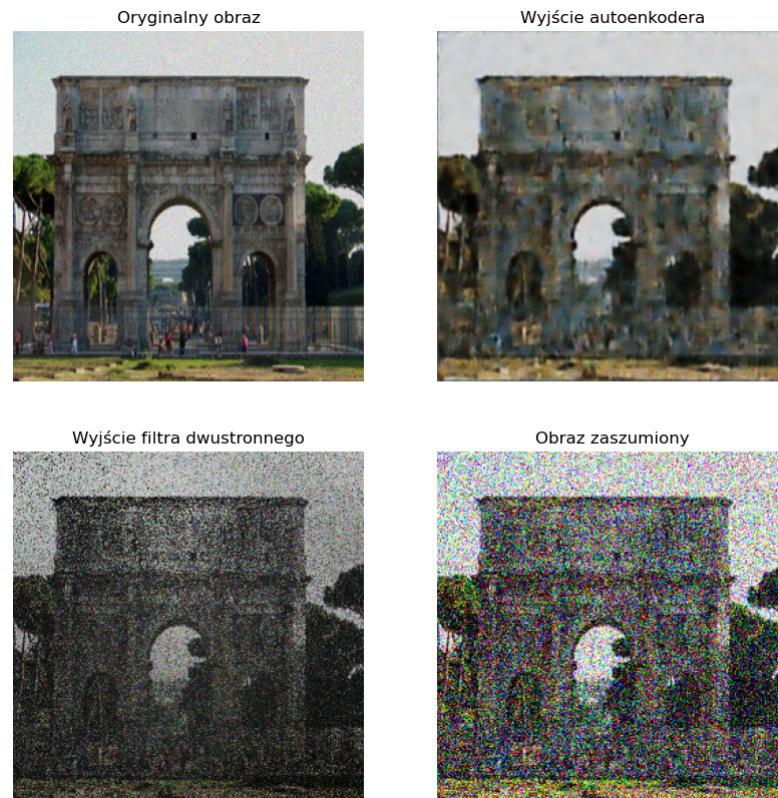
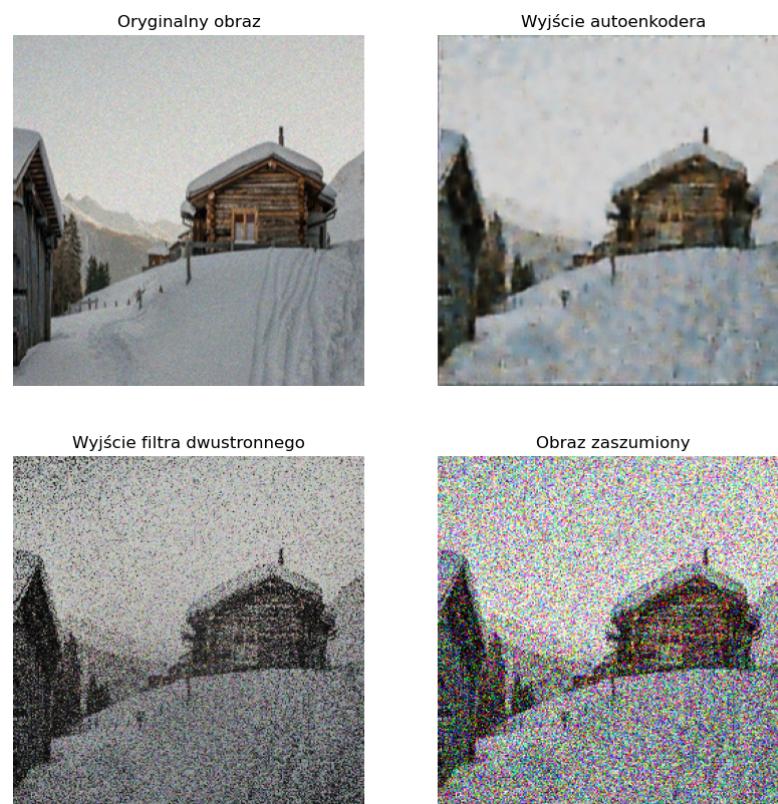
- μ_x, μ_y – średnie wartości pikseli obrazów x i y ,
- σ_x^2, σ_y^2 – wariancje obrazów x i y ,
- σ_{xy} – kowariancja obrazów,
- C_1, C_2 – stałe stabilizujące.

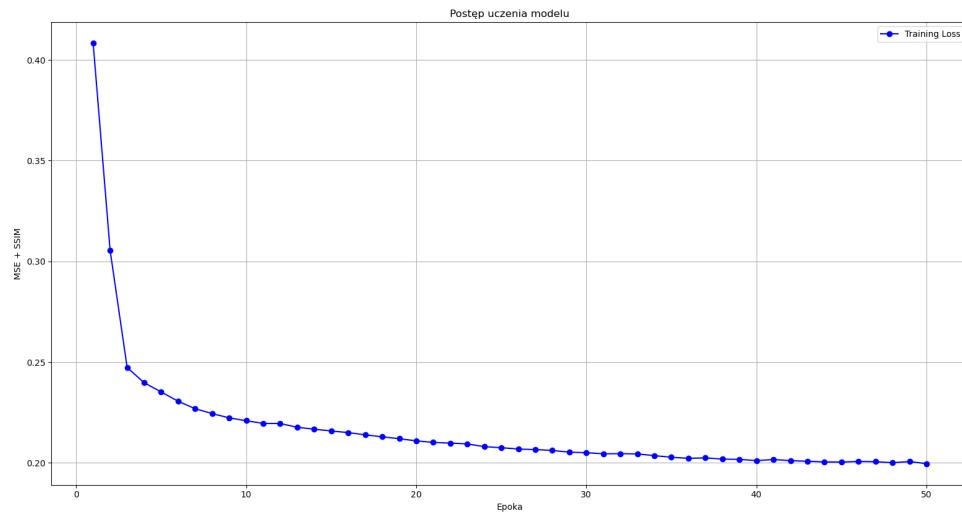
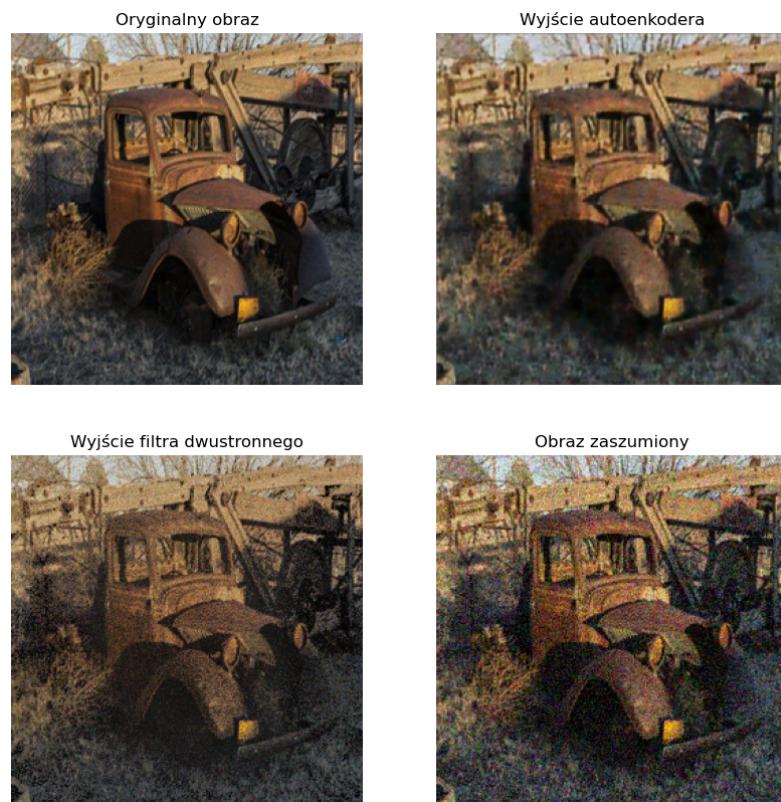
Podczas trenowania współczynniki α oraz β wynosiły 0,5. Wejściem do uczenia modelu były obrazki ze zbioru danych DIV2K_train_LR_mild przekształcone do rozmiarów 256x256.

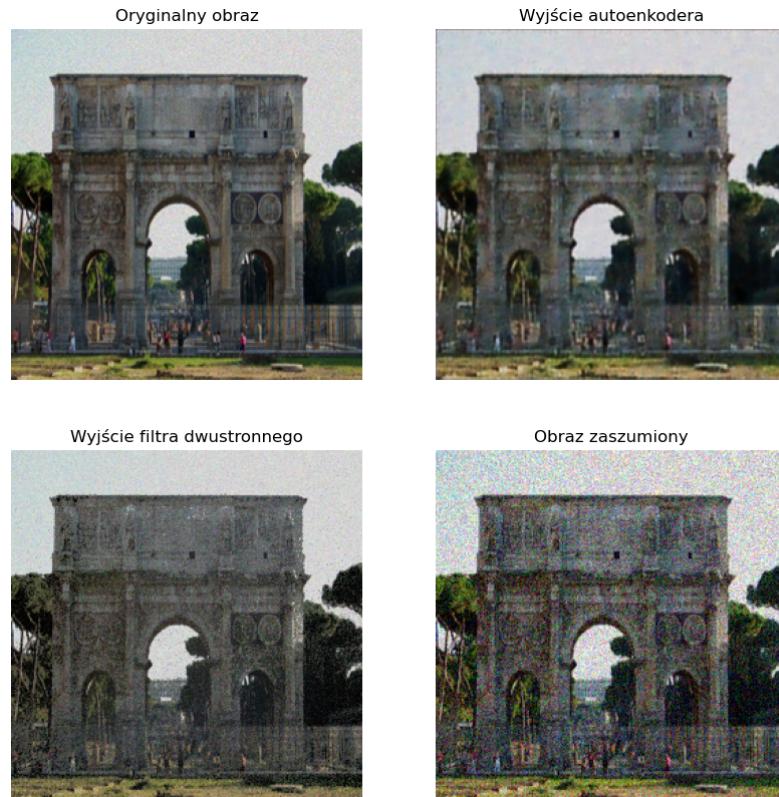
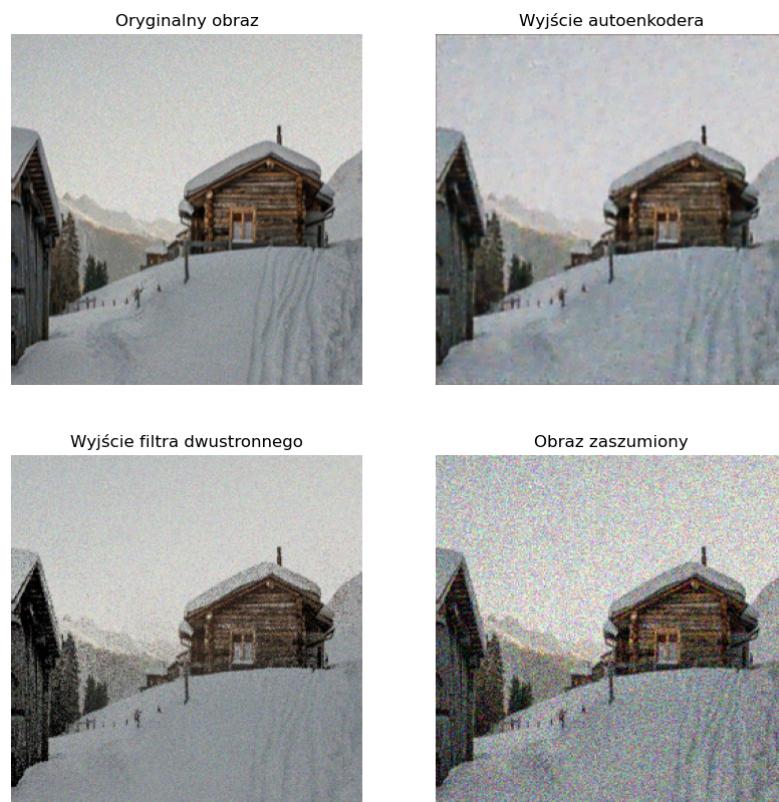
1.3 Wyniki

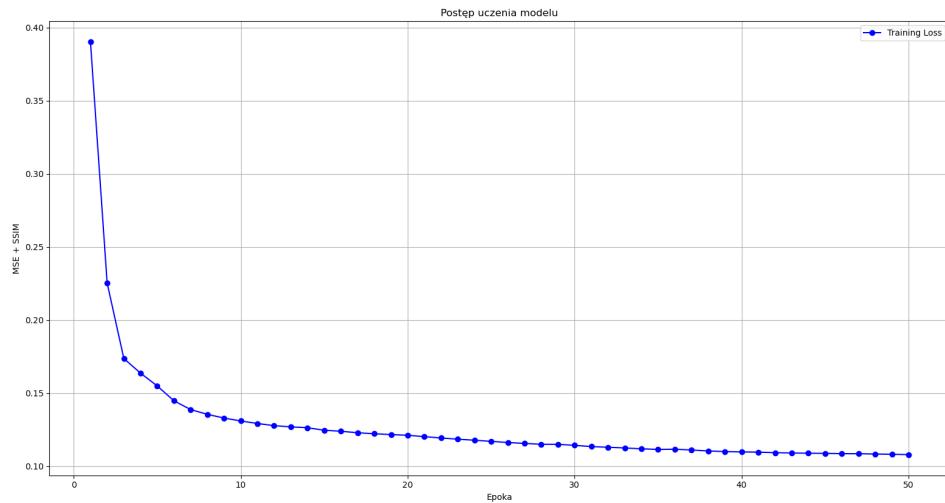


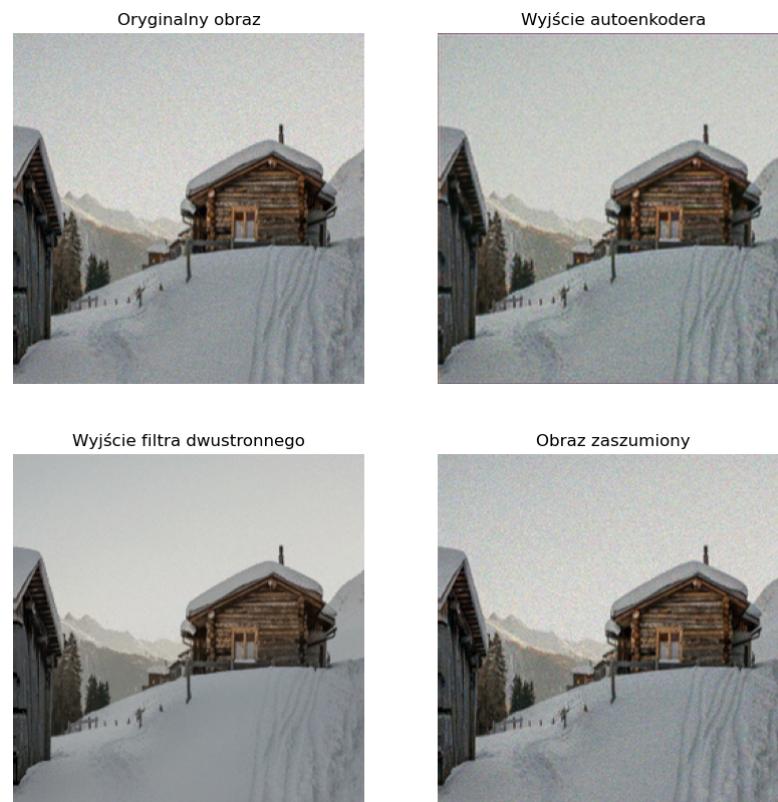
Rysunek 1: Auto $\sigma = 0,3$

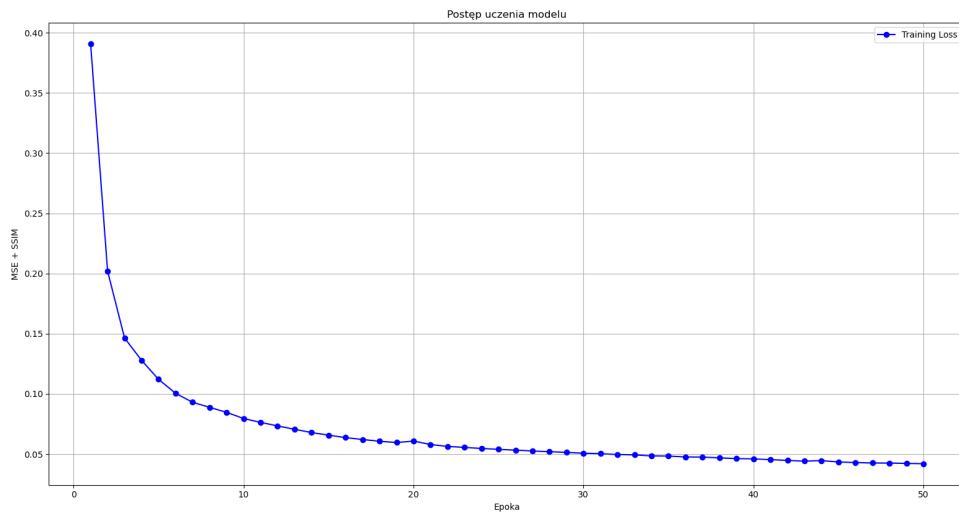
Rysunek 2: Łuk triumfalny $\sigma = 0,3$ Rysunek 3: Łuk triumfalny $\sigma = 0,3$

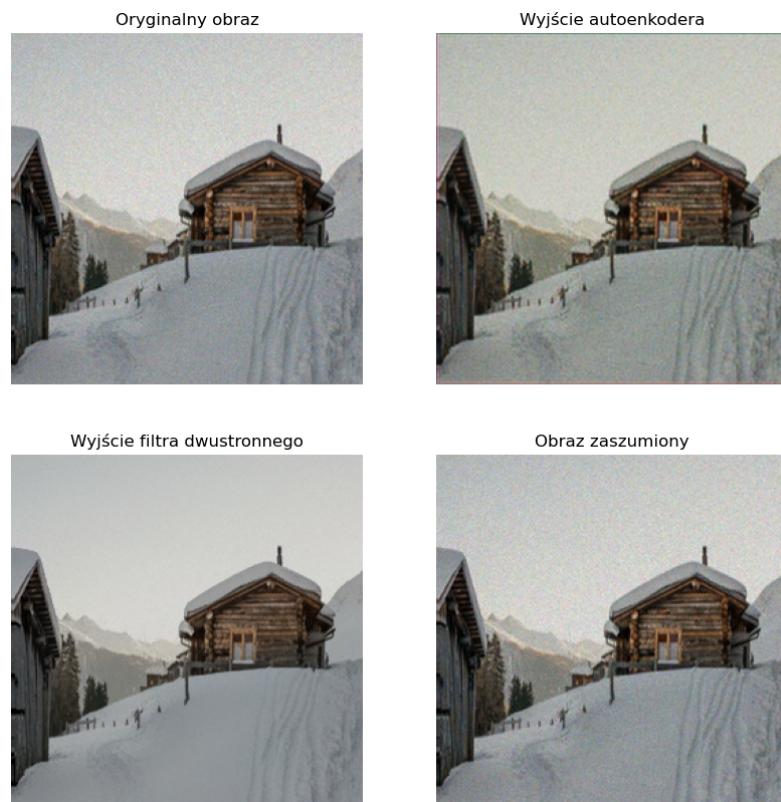
Rysunek 4: Trening $\sigma = 0,3$ Rysunek 5: Auto $\sigma = 0,1$

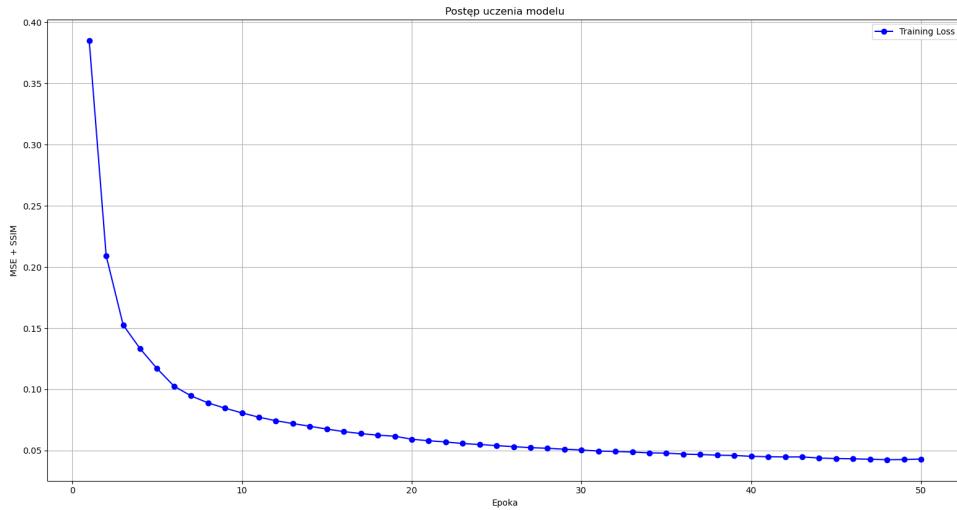
Rysunek 6: Łuk triumfalny $\sigma = 0, 1$ Rysunek 7: Łuk triumfalny $\sigma = 0, 1$

Rysunek 8: Trening $\sigma = 0,1$ Rysunek 9: Auto $\sigma = 0,02$

Rysunek 10: Łuk triumfalny $\sigma = 0,02$ Rysunek 11: Łuk triumfalny $\sigma = 0,02$

Rysunek 12: Trening $\sigma = 0,02$ Rysunek 13: Auto $\sigma = 0,01$

Rysunek 14: Łuk triumfalny $\sigma = 0,01$ Rysunek 15: Łuk triumfalny $\sigma = 0,01$

Rysunek 16: Trening $\sigma = 0,01$

σ	SNE	PSNR	SSIM	LPIPS
0.3	12347.3606	0.7069	0.0185	0.3893
0.1	1805.6389	0.8397	0.0246	0.1292
0.02	77.9951	0.9320	0.0285	0.0359
0.01	20.3513	0.9473	0.0292	0.0246

Tabela 1: Porównanie metryk dla autoenkodera

1.4 Wnioski

Dla małych wartości σ filtr dwustronny radzi sobie nieznacznie lepiej z odszumianiem - autoenkoder daje w wyniku trochę mniej wyraźne krawędzie. Wraz ze wzrostem σ coraz trudniej dobrać odpowiednie parametry funkcji filtra dwustronnego i generalnie przefiltrowany obraz jest optycznie złą jakości. Za to autoenkoder daje obrazek, który zachowuje kolory i kształty.

2 Deblurowanie

W zadaniu został wykorzystany model UNet.

2.1 Architektura modelu UNet

Do zadania wykorzystano zredukowaną architekturę UNet - z mniejszą ilością warstw oraz kanałów w warstwach splotowych. Model składa się z następujących warstw:

- Wejściowa warstwa konwolucyjna.
- 2 warstwy enkodera, składające się z warstwy poolingu maksymalnego o rozmiarze jądra 2 i parametrze kroku 2, oraz 2 warstw konwolucyjnych.
- 2 warstwy konwolucyjne w środku sieci, po każdej z których następuje warstwa dropoutu.

- 2 warstwy dekodera, składające się z warstwy upsample, korzystającej z algorytmu bilinear, skalującej obraz dwukrotnie, oraz 2 warstw konwolucyjnych.
- Wyjściowa warstwa konwolucyjna, przywracająca obraz do 3 kanałów.

Każda warstwa konwolucyjna ma rozmiar jądra 3×3 oraz parametry kroku i dopełnienia równe 1. Po każdej warstwie splotowej, oprócz wyjściowej, następuje normalizacja pakietowa i funkcja aktywacji *LeakyReLU*. Po blokach enkodera zapisywane są ich wyjścia, które następnie są dodawane do odpowiadających bloków dekodera. Do tego końcowy obraz jest uzyskiwany poprzez zsumowanie wyjścia sieci ze zdjęciem początkowym.

2.2 Funkcja straty

Funkcja straty łączy błąd średniokwadratowy (**MSE**) i stratę percepcyjną **LPIPS** wykorzystującą model VGG.

$$\mathcal{L} = \lambda_{MSE} \mathcal{L}_{MSE}(\tilde{Y}, Y) + \lambda_{LPIPS} \mathcal{L}_{LPIPS}(\tilde{Y}, Y) \quad (4)$$

Wartości poszczególnych komponentów funkcji straty są przeskalowane przez współczynniki λ .

2.3 Trening

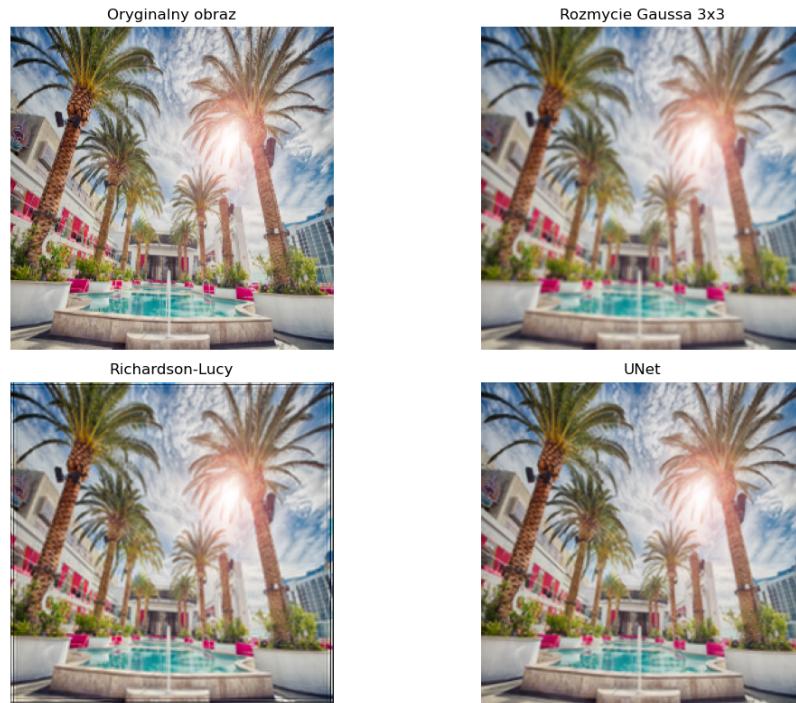
Podczas treningu modelu współczynnik λ_{MSE} był ustawiony na 1, a λ_{LPIPS} na 0,5. Model był trenowany na zbiorze danych DIV2K_train_LR_bicubic o czterokrotnie zredukowanej rozdzielcości. Obrazy były przekształcone do rozmiarów 256×256 . Model był trenowany w 2 konfiguracjach - dla szumu Gaussa o rozmiarze jądra 3×3 i 5×5 . W obu przypadkach wartość σ rozmycia była ustawiona na losową wartość z przedziału [1,5, 2,5]. Trening trwał godzinę z wykorzystaniem GPU Nvidia RTX 3060.



Rysunek 17: Zmiana wartości f. straty podczas treningu dla modelu 3×3 . Oś pionowa - wartość f. straty, Oś pozioma - krok treningu

Wartość funkcji straty z początku szybko zmalała, natomiast w dalszej części treningu zmiany są mniej gwałtowne. Wykres funkcji starty wygląda podobnie dla modelu 5×5

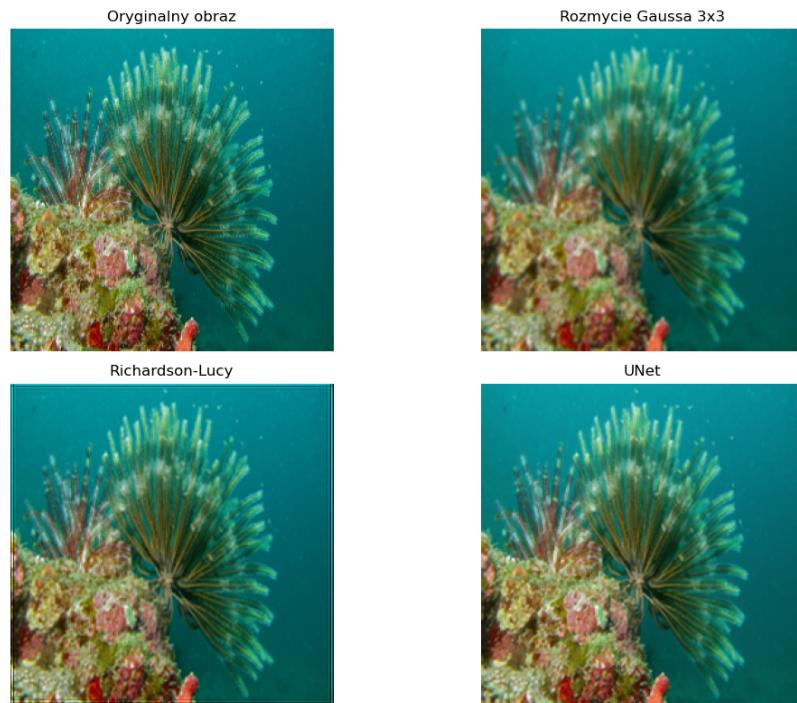
2.4 Wyniki



Rysunek 18: Palmy, rozmycie Gaussa 3×3



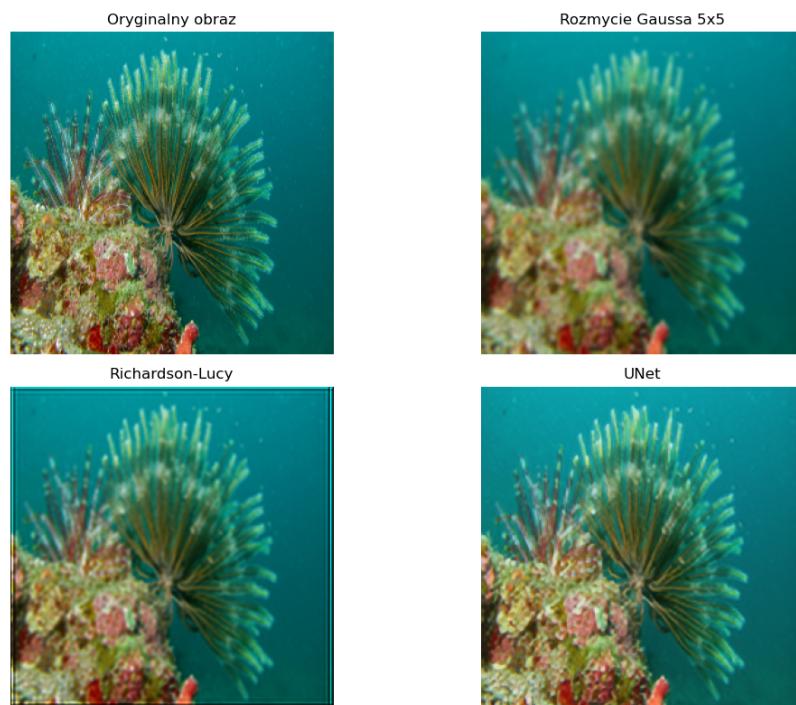
Rysunek 19: Tygrys, rozmycie Gaussa 3×3



Rysunek 20: Rafa kolarowa, rozmycie Gaussa 3×3



Rysunek 21: Palmy, rozmycie Gaussa 5×5

Rysunek 22: Tygrys, rozmycie Gaussa 5×5 Rysunek 23: Rafa kolarowa, rozmycie Gaussa 5×5

Rozmiar jądra	SNE	PSNR	SSIM	LPIPS
3×3	669,8537	24,22	0,7789	0,1203
5×5	789,1681	23,96	0,7477	0,1422

Tabela 2: Porównanie metryk dla sieci UNet

2.5 Wnioski

Dla mniejszego jądra rozmycia Gaussa wytrenowany model radzi sobie z obrazami podobnie do algorytmu Richardson-Lucy. Z kolei dla większego jądra różnice są bardziej widoczne - obrazy deblurowane z wykorzystaniem modelu wyglądają lepiej niż z wykorzystaniem algorytmu. Jest to zwłaszcza widoczne dla szczegółów o wysokiej częstotliwości. Wadą algorytmu Richardson-Lucy jest wymóg dostosowania funkcji do dekonwolucji - co może być trudniejsze, gdy nie wiemy, w jaki sposób rozmycie zostało dodane do obrazu. Z kolei ograniczeniem modelu jest stały rozmiar obrazów.