

Politechnika Warszawska



WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Sztuczna inteligencja w grafice komputerowej

Sprawozdanie z projektu:
Modyfikacje obrazów

Wojciech Makurat

Karol Zębala

05.11.2025

Prowadzący: dr Michał Chwesiuk

Spis treści

1. Odszumianie	2
1.1. Zaproponowana architektura	2
1.1.1. Architektura 1	2
1.1.2. Architektura 2	2
1.2. Wyniki	2
2. Inpating	3
2.1. Zaproponowana architektura	3
2.2. Wyniki	4

1. Odszumianie

Celem zadania jest przygotowanie architektury, która umożliwi odszumienie obrazu. Opracowane rozwiązanie zostanie porównane z metodą bazową - `denoise_bilateral` z `skimage`

1.1. Zaproponowana architektura

W ramach realizacji zadania opracowane zostały dwie architektury oparte o UNet. W pierwszej wyjściem sieci jest odszumiony obraz, a w drugiej szum, który należy odjąć od zaszumionego obrazu wejściowego aby otrzymać finalny obraz

W implementacji modelu wykorzystano następujące podstawowe komponenty:

DoubleConv Moduł składający się z dwóch kolejnych warstw konwolucyjnych o rozmiarze jądra 3×3 , po każdej z których zastosowano normalizację wsadową (*Batch Normalization*) oraz funkcję aktywacji ReLU.

Down Moduł realizujący operację redukcji wymiarów przestrzennych poprzez połączenie warstwy *MaxPooling* z modułem **DoubleConv**. Stosowany w etapach enkodera sieci.

Up Moduł odpowiedzialny za zwiększenie rozdzielczości danych wyjściowych (tzw. *upsampling*), połączony z modułem **DoubleConv**.

1.1.1. Architektura 1

Enkoder składa się z **DoubleConv** następnie z dwóch warstw **Down** oraz jednej **DoubleConv**. Dekoder składa się z 3 **Up** oraz **Finalnej konwolucji**. Wyjście z sieci jest ograniczone do [0,1]

1.1.2. Architektura 2

Endoder składa się z 3 warstw

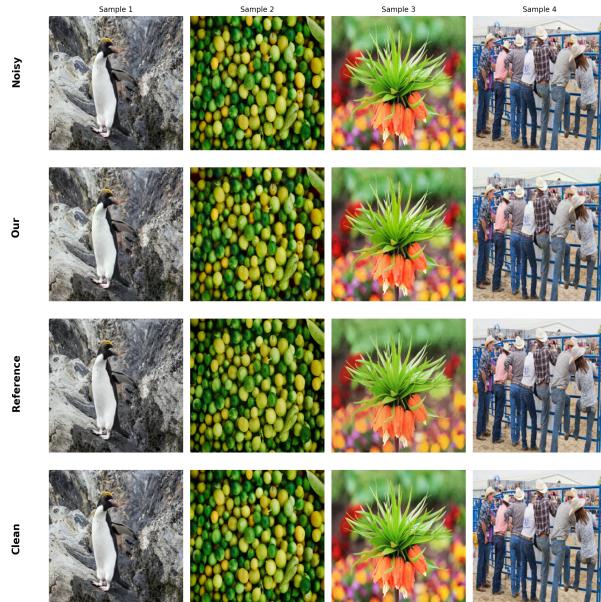
oraz

. Dekoder składa się z 4

oraz

. Wyjście z sieci jest ograniczone do [0,1]

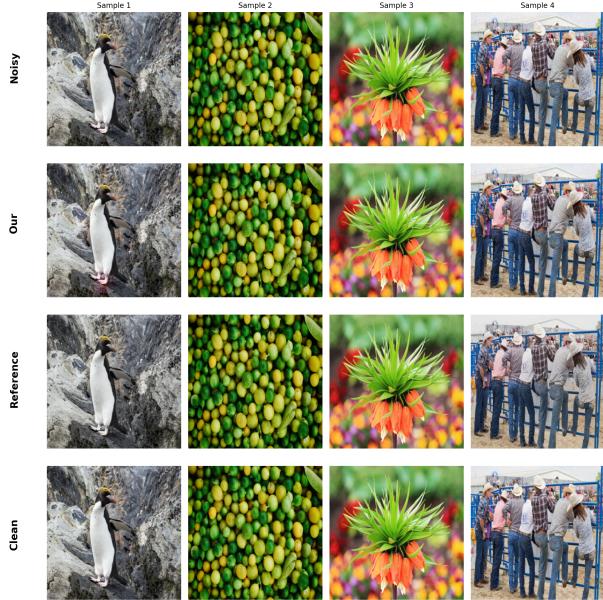
1.2. Wyniki



Rysunek 1: Porównanie wizualne dla pierwszej architektury

Tabela 1: Porównanie wyników różnych metod

Metoda	PSNR	SSIM	LPIPS
Denoise_bilateral	34.39	0.96	0.044
Architektura 1	30.41	0.91	0.070
Architektura 2	34.93	0.91	0.038



Rysunek 2: Porównanie wizualne dla drugiej architektury

Architektura 2 osiągnęła lepsze wyniki niż bazowa funkcja oraz Architektura 1. Jedynie w przypadku SSIM wynik okazał się gorszy.

Architektura 1 okazała się gorsza od metody bazowej stąd decyzja o implementacji drugiej architektury.

Natomiast porównując wizualnie obrazy okazuje się, że odszumianie może powodować przebarwienia tak jak na obrazku 1

2. Inpating

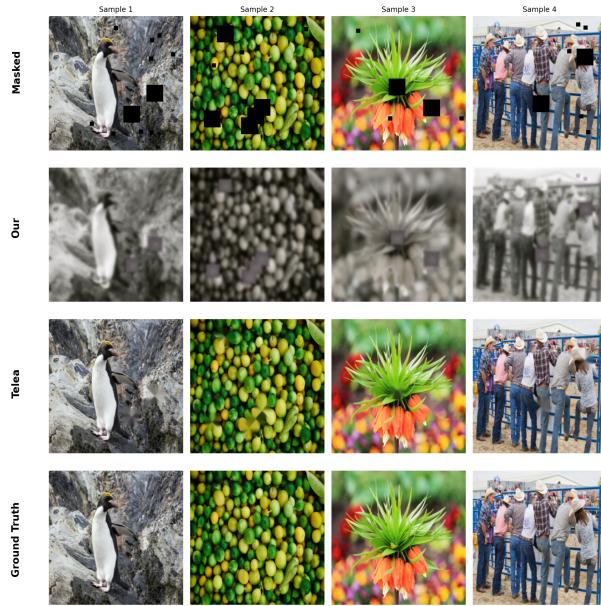
Celem zadania jest przygotowanie architektury, która umożliwi wykonanie inpatingu, czyli uzupełnienia brakujących fragmentów obrazu. Opracowane rozwiązanie zostanie porównane z metodą bazową - inpaint z OpenCv z parametrem INPAINT_TELEA

2.1. Zaproponowana architektura

W ramach realizacji zadania opracowana została architektura oparta o sieć U-Net z wykorzystaniem konwolucji częściowych (Partial Convolution). Sieć przyjmuje jako wejście obraz z maską określającą brakujące fragmenty, a jej celem jest rekonstrukcja pełnego obrazu poprzez uzupełnienie obszarów maskowanych.

W implementacji modelu wykorzystano następujące podstawowe komponenty: **PartialConv** – warstwa konwolucyjna przetwarzająca jedynie znane fragmenty obrazu (zgodnie z maską) oraz aktualizującą maskę dla kolejnych warstw. **Encoder** – część sieci odpowiedzialna za ekstrakcję cech, zbudowana z kolejnych warstw PartialConv o rosnącej liczbie filtrów i operacjach downsamplingu. **Decoder** – część odpowiedzialna za rekonstrukcję obrazu poprzez upsampling i łączenie z odpowiednimi poziomami enkodera (skip connections). Wykorzystuje warstwy PartialConv do stopniowego wypełniania brakujących obszarów. **Funkcja straty** – łączna miara błędu obejmująca składniki L1 (dla pikseli znanych i maskowanych), perceptual loss oparty na cechach VGG16, style loss wykorzystujący macierz Grama oraz total variation loss dla wygładzenia rekonstrukcji.

2.2. Wyniki



Rysunek 3: Porównanie wizualne dla i

Tabela 2: Porównanie wyników różnych metod

Metoda	PSNR	SSIM	LPIPS
OpenCV inpaintn	inf	0.97	0.03
Nasz	19.18	0.59	0.64

Nasz model poradził sobie z uzupełnieniem mniejszych fragmentów (3x3). Udało mu się również zmniejszyć większe luki (32x32). Niestety kolory na wyjściu sieci są bardzo wyblakłe. Jest to spowodowane najprawdopodobniej zbyt krótkim czasem uczenia. Aby nauczyć odpowiednio sieć musielibyśmy ją trenować kilka razy dłużej, co jest dodatkowo utrudnione przez brak dostępu do odpowiedniej infrastruktury