Usuwanie artefaktów kompresji stratnej przy pomocy głębokich sieci neuronowych

Mateusz Szulc

13 kwietnia 2023

Celem dokumentu jest przedstawienie sprawozdania z wykonania projektu w ramach przedmiotu Projekt Indywidualny na semestrze 4 kierunku Informatyka Stosowana Wydziału Elektrycznego PW.

Projekt realizowany jest pod kierunkiem dr hab. inż. Marcina Iwanowskiego.

Spis treści

1 Opis problemu

Projekt ma na celu zaprezentowanie możliwości wykorzystania sieci neuronowych typu U-Net do usuwania artefaktów kompresji stratnej JPEG. Sprawdzone zostały różne architektury sieci, od 1 do 5 poziomów, oraz wpływ rozmiaru segmentu wejściowego na wartość funkcji straty. Rozwiązanie zostało przystosowane do przetwarzania obrazów o dowolnym rozmiarze, większym niż segment podawany na wejściu sieci.

2 Algorytm kompresji JPEG

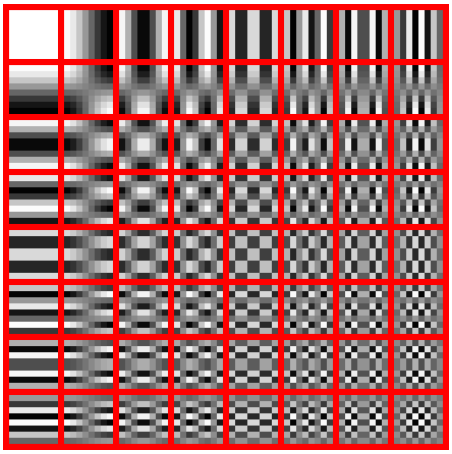
Przy niskich wartościach parametru jakości, na zapisanych obrazach wyraźny staje się podział zdjęcia na sekcje przez algorytm, jak i zaburzenia na krawędziach obiektów. Wynikają one z opisanego poniżej sposobu działania algorytmu. [3]

2.1 Operacje na przestrzeni barw

W pierwszej kolejności następuje konwersja przestrzeni barw RGB do przestrzeni YCbCr. Następnie chrominancja jest podpróbkowana, w wyniku czego każdy pixel obrazu wyjściowego stanowi średnią 4 pixeli obrazu wejściowego. Nastepuje utrata danych ale jest mało odczuwalna ze względu budowę ludzkiego oka.

2.2 Dyskretna transformata kosinusowa

Następuje podział obrazu na bloki 8x8 pixeli i odjęcie 128 od wartości każdego pixela. Każdy blok zostaje przekształcony w macierz, obrazującą użycie poszczególnych bloków dyskretnej transformaty kosinusowej.



Rysunek 1: Dyskretna Transformata Kosinusowa

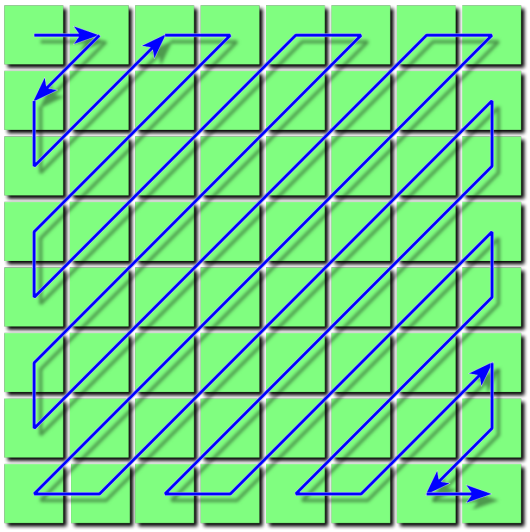
Sumując wartości pixeli z wybranych bloków, możemy uzyskać dowolną kombinację 8x8 pixeli.

2.3 Kwantyzacja

W kolejnym kroku następuje zmniejszenie ilości bitów potrzebnych do reprezentacji wartości pixeli. Wartości z macierzy są dzielone przez odpowiadające wartości z tablicy kwantyzacji, odwrotnie proporcjonalne do parametru jakości wybranego przy zapisie zdjęcia. Współczynniki w tablicy kwantyzacji rosną, zmierzając do dolnej, prawej części macierzy. Odpowiada to zwiększaniu się zagęszczenia informacji w tablicy bloków wykorzystanej w poprzednim kroku. Taka operacja wykorzystuje problem z rozróżnieniem informacji o dużym zagęszczeniu przez ludzkie oko. Otrzymane wartości są zaokrąglane do najbliższej liczby całkowitej, następuje utrata danych.

2.4 Kodowanie

Otrzymana macierz jest spłaszczana według następującego wzorca:

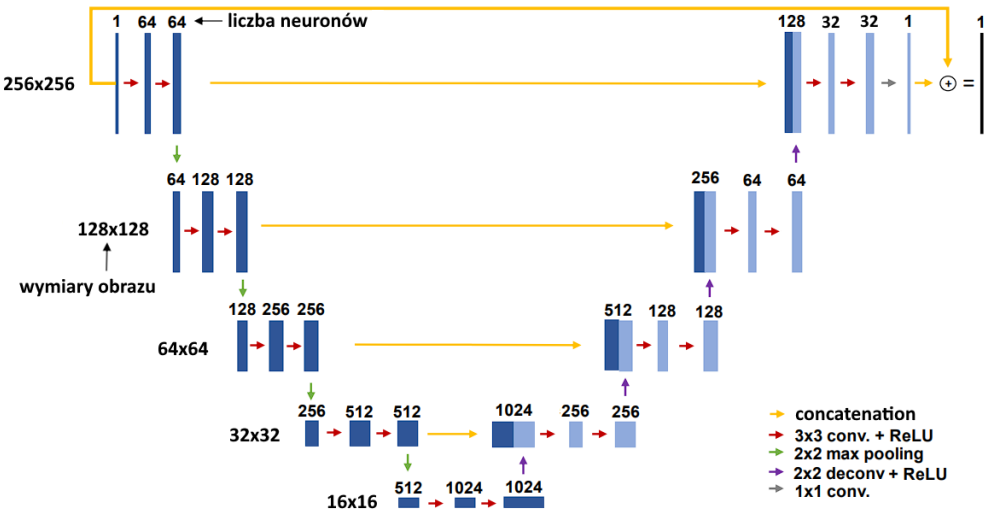


Rysunek 2: Kolejność spłaszczania segmentu

Co zwiększa prawdopodobieństwo wystąpienia długich sekwencji zer, możliwych do zapisania w zwięźlejszy sposób. Na koniec obraz zostaje poddany kompresji kodowaniem Huffmana.

3 Sieci typu U-Net

U-Net jest architekturą sieci konwolucyjnch, najczęściej stosowaną do segmentacji semantycznej obrazów [4]. Udokumentowane zostały również próby wykorzystania architektury do usuwania zakłóceń z obrazu [1] . Wyróżnić można 2 części sieci U-Net.



Rysunek 3: Zastosowana w projekcie architektura sieci U-Net o 5 poziomach [4]

3.1 Enkoder

Pierwszą częścią modelu jest warstwa wejściowa, w której każdemu pixelowi monochromatycznego segmentu wejściowego odpowiada 1 neuron. Następnie występują 2 etapy składające się z konwolucji o jądrze 3x3 oraz funkcji aktywacji ReLu, zwiększające liczbę neuronów do 64. Kolejne są powtarzalne segmenty składające się z redukcji rozdzielczości segmentu i konwolucji z funkcją aktywacji ReLu. Skutkuje to czterokrotnym zmniejszeniem rozmiaru segmentu oraz podwojeniem liczby neuronów w każdej warstwie.

Listing 1: Implementacja warstwy enkodera w bibliotece Keras

```
pooling = layers.MaxPooling2D(pool_size=2, strides=2)(input)
conv = layers.Conv2D(
    neurons_number, kernel_size=3, padding="same", activation="relu"
)(pooling)
layers.Conv2D(
    neurons_number, kernel_size=3, padding="same", activation="relu"
)(conv)
```

3.2 Dekoder

Każda z powtarzalnych warstw dekodera składa się z 3 elementów. Pierwsza jest transponowana konwolucja o jądrze 2x2. Następnie powstała sieć jest łączona z odpowiadającą jej warstwą enkodera. Takie połączenie przeciwdziała problemowi zanikającego gradientu - przywraca cechy wykryte w poprzednich warstwach sieci. Po niej występują 2 warstwy konwolucji z funkcją aktywacji ReLu, takie same jak w enkoderze.

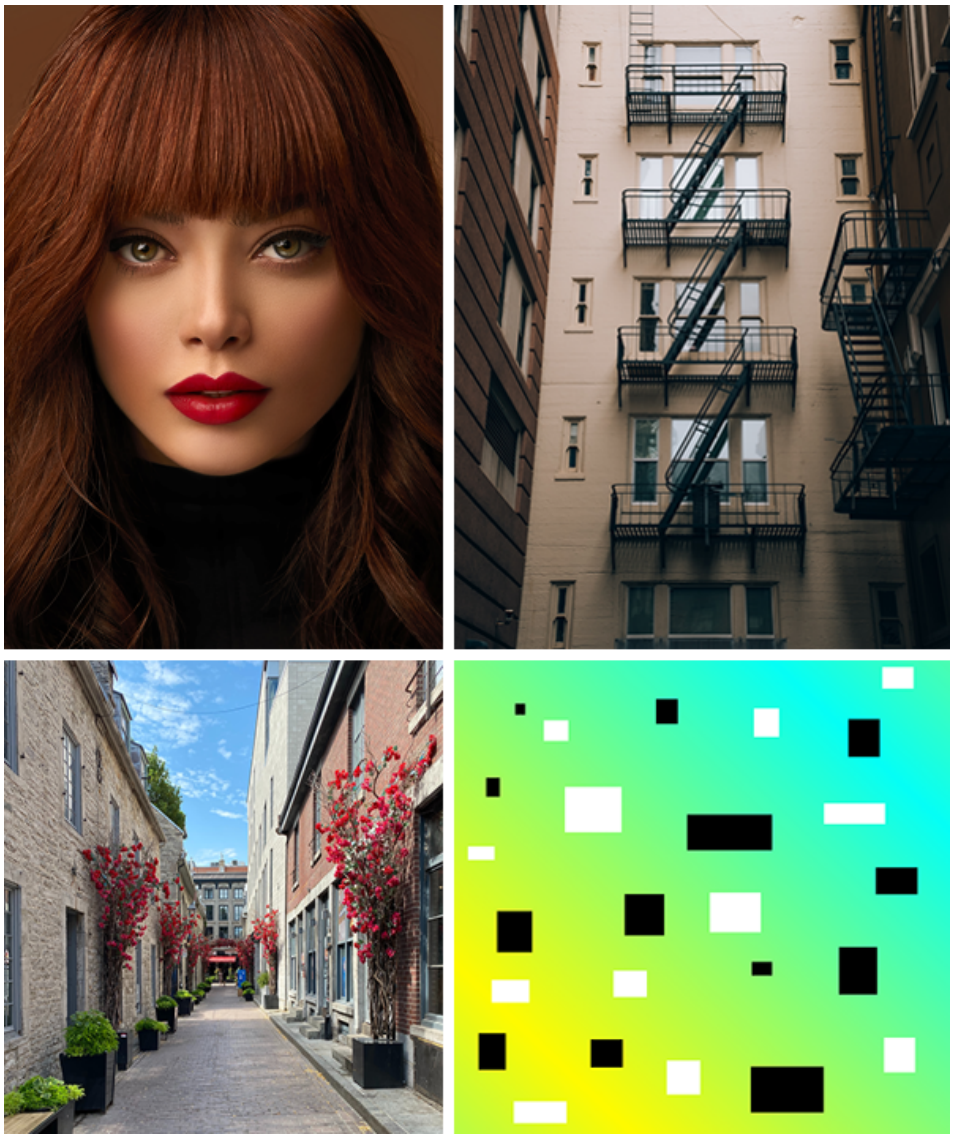
Listing 2: Implementacja warstwy dekodera w bibliotece Keras

```
tconv = layers.Conv2DTranspose(
    neurons_number, kernel_size=2, strides=(2, 2), activation="relu"
)(input)
concat = layers.Concatenate(axis=3)([tconv, connected_layer])
dconv = layers.Conv2D(
    neurons_number / 2, kernel_size=3, padding="same", activation="relu"
)(concat)
layers.Conv2D(
    neurons_number / 2, kernel_size=3, padding="same", activation="relu"
)(dconv)
```

Dekoder zakończony jest spłaszczeniem segmentu do postaci, w której każdy neuron odpowiada 1 pikselowi segmentu wyjściowego. Warstwa wyjściowa jest sumą segmentu wejściowego i spłaszczonej warstwy. Przy takim rozwiązaniu, zwanym połączeniami rezydualnymi, sieć tworzy maskę zmian potrzebnych w segmentu wejściowym, zamiast bezpośrednio przekształcać segment. [2]

4 Generowanie zbioru danych

Zbiór danych stanowi 16 kolorowych zdjęć wysokiej rozdzielczości, 1 monochromatyczny wykres służący do kalibracji sprzętu fotograficznego, oraz 3 wygenerowane obrazy składające się z gradientów kolorów i prostych kształtów. Zdjęcia zostają podzielone na warstwy RGB i pocięte na kwadratowe segmenty o zadanym rozmiarze. Wygenerowane zostają wszystkie możliwe kombinacje obróceń i odbić segmentów. Oryginał segemntu zostaje zapisany przy ustawieniu jakości równym 100 wykorzystywana jako maska w modelu. Dane uczące zostaja zapisane przy ustawieniu równym 25, przy którym artefakty kompresji zaczynają być łatwo zauważalne. W przypadku podziału na segmenty o rozmiarze 256x256 pixeli otrzymujemy 9500 próbek w zbiorze testowym, oraz 1000 w zbiorze walidacyjnym.

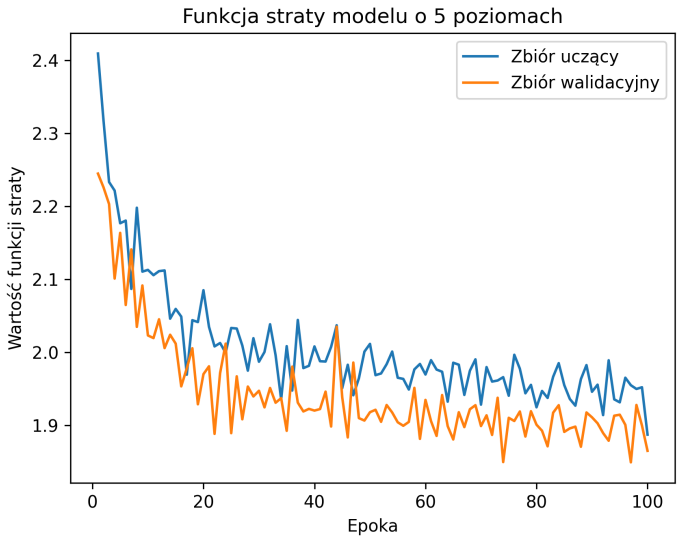


Rysunek 4: Przykładowe obrazy ze zbioru danych

5 Model i szkolenie

Podstawową architekturą w projekcie jest, najczęściej stosowana, sieć o 5 poziomach. Podczas uczenia utworzone zostają generatory danych testowych i walidacyjnych. Zawierają one skompresowane oraz oryginalne segmentu, te drugie wykorzystywane w modelu jako maska. Pogrupowane zostają w paczki po 3. Podczas jednego kroku uczenia, wykorzystane zostaje, losowo wybrane, 10% paczek ze zbioru uczącego. Proces uczenia trwa 100 epok. Dane uczące i walidacyjne wybierane są na podstawie stałego ziarna, co pozwala na porównanie wyników modeli. Jako funkcję straty wybrana została wartość bezwzględna błędu, optymalizator to Adam o współczynniku uczenia 0,0001.

Zauważyć można, że po 20 epoce, tempo zmniejszania wartości funkcji straty znacząco maleje, a po 60 epoce praktycznie zatrzymuje. Wartym uwagi jest niższa wartość straty w zbiorze walidacyjnym niż uczącym. Jako, że zbiory zostały utworzone z tych samych zdjęć, prawdopodobnie jest, że to losowy przypadek w



Rysunek 5: Funkcja straty

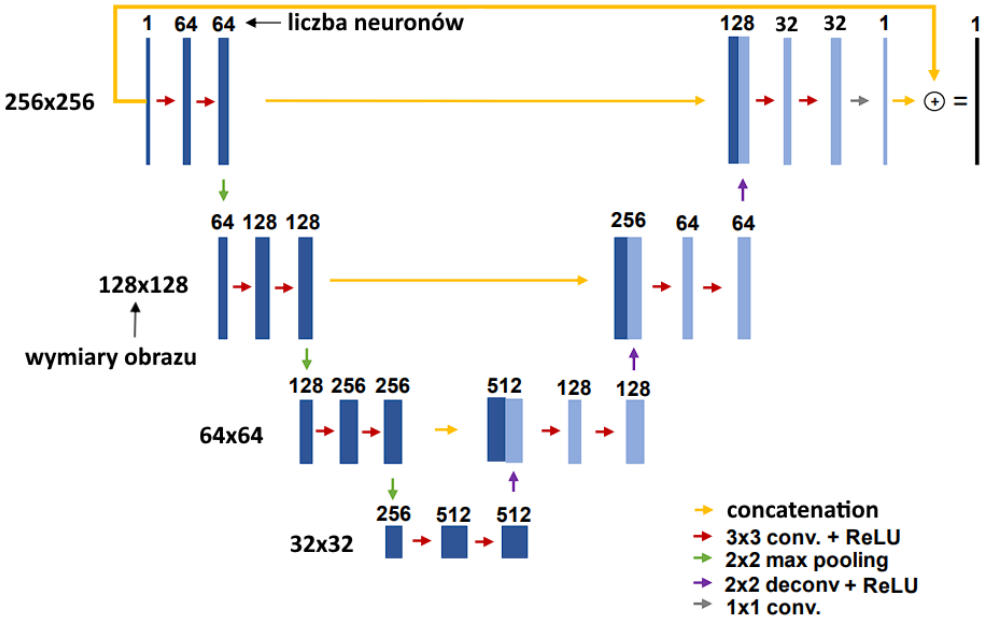
wybranym ziarnie generatora danych. Podczas uczenia dokładność była znikoma i stała. Nie jest ona jednak miarodajną metryką, gdyż oczekuje dokładnych wartości pixeli. Człowiek nie rozpoznaje małych różnic w wartościach RGB, w związku z czym ważna była jedynie wartość funkcji straty.

6 Wybór optymalnej architektury

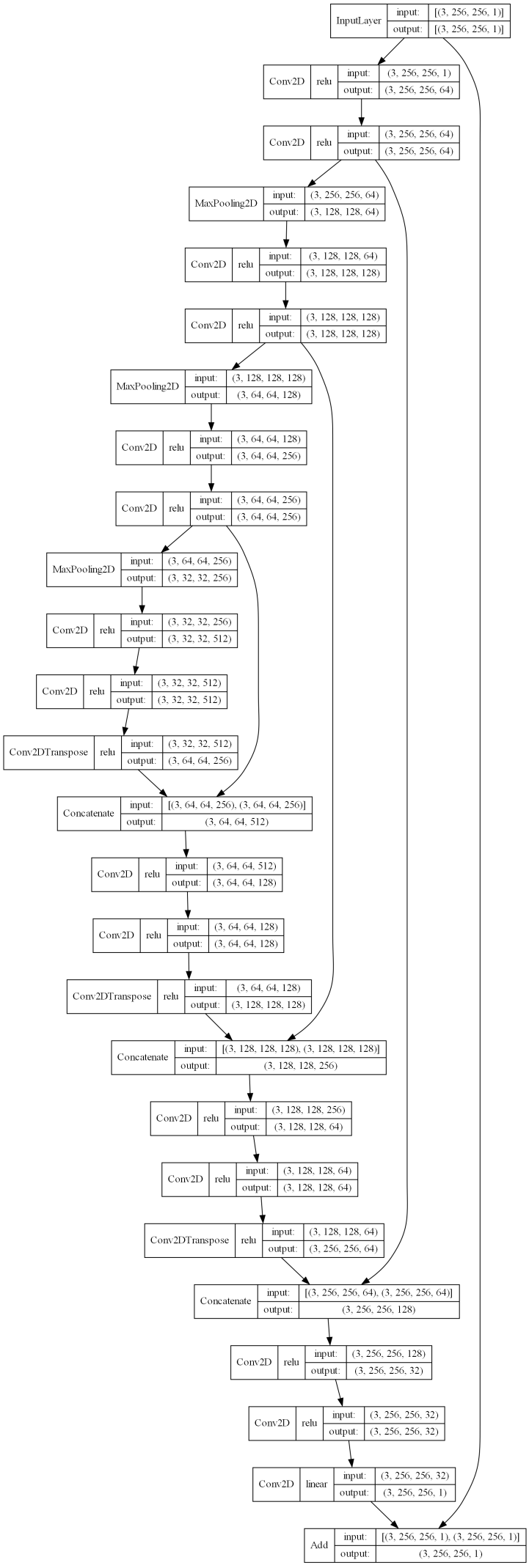
Ważnym przy wyborze architektury sieci jest jej złożoność, nadmierna negatywnie wpływa na czas uczenia pomimo znikomej poprawy wartości funkcji błędu.

6.1 Liczba poziomów sieci

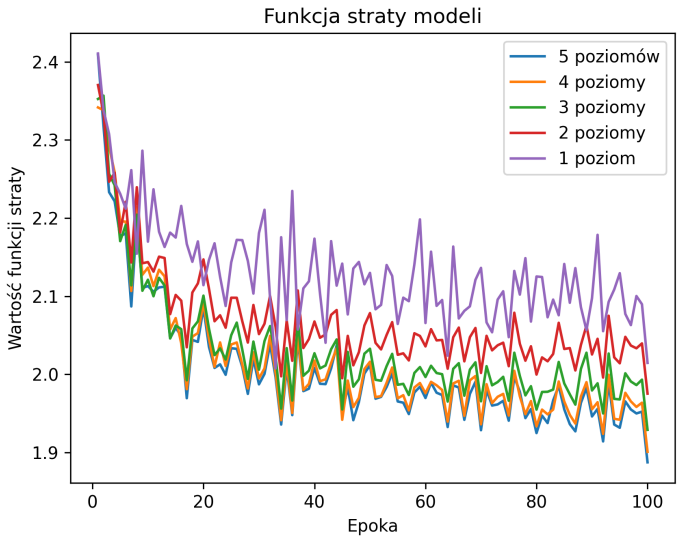
Przetestowane zostały sieci od 1 do 5 poziomów. Każda została wyszkolona w takich samych warunkach i przy stałym ziarnie generatora danych.



Rysunek 6: Architektura sieci U-Net o 4 poziomach



Rysunek 7: Implementacja sieci U-Net o 4 poziomach w Keras

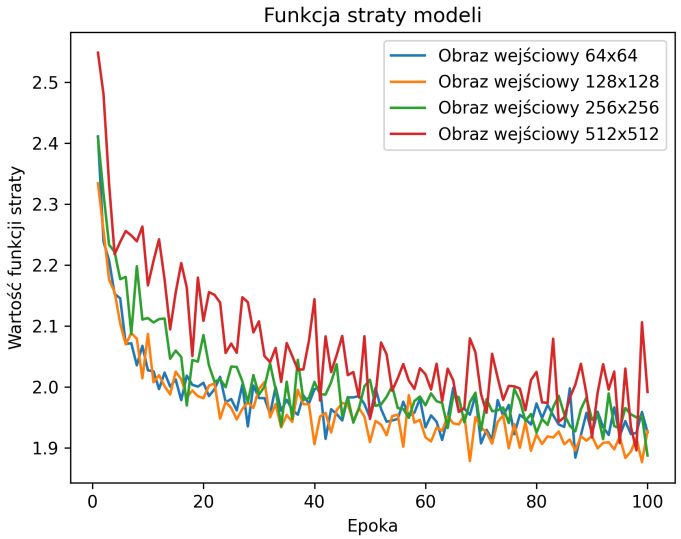


Rysunek 8: Funkcja straty przy różnej liczbie poziomów sieci

Zauważyć można zbliżone wyniki sieci o 3, 4 i 5 poziomach. Dla 1 i 2 poziomów wystąpiło znaczące pogorszenie wyników. Zmiana wartości funkcji straty spowolniła przy podobnej liczbie epok. Wynika z tego, że do problemu wystarczy zastosowania sieci o 3 lub 4 poziomach, co przyśpieszy proces uczenia.

6.2 Rozmiar segmentu wejściowego

Przetestowany został również wpływ rozmiaru segmentu wejściowego na wartość funkcji straty przy 4 poziomach sieci, przedstawiony na Rys. 9. Zauważyć



Rysunek 9: Funcja straty przy różnych rozmiarach segmentu wejściowego

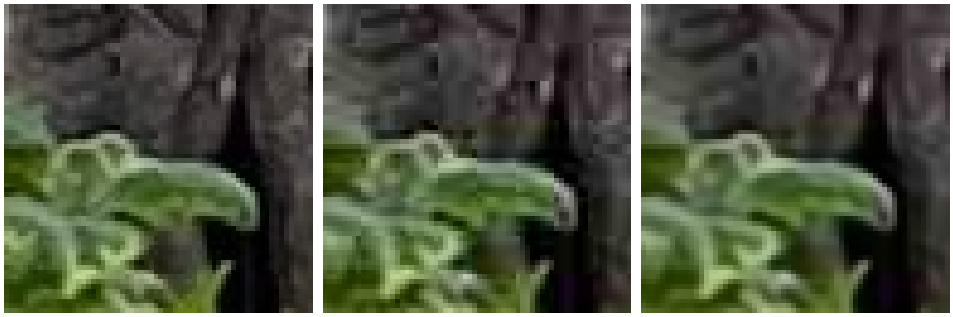
można podobne wyniki dla segmentów o rozmiarze 64x64 oraz 128x128 pixeli. Dla segmentu o rozmiarach 128x128 pixeli zauważyć można nieznacznie niższy błąd dla końcowych epok. Jest to jednak skutkiem przetrenowania, gdyż wartość

funkcji straty w zbiorze walidacyjnym była stała. W przypadku segmentów o rozmiarze 512x512 pixeli nastąpiło znaczące pogorszenie wyników. Rozważyć można wybór dowolnego z rozmiarów 64x64, 128x128 oraz 256x256 pixeli. Należy jednak pamiętać, że mniejszy rozmiar segmentu wejściowego, choć przyśpiesza szkolenie, proporcjonalnie wpływa na czas rekonstrukcję obrazu.

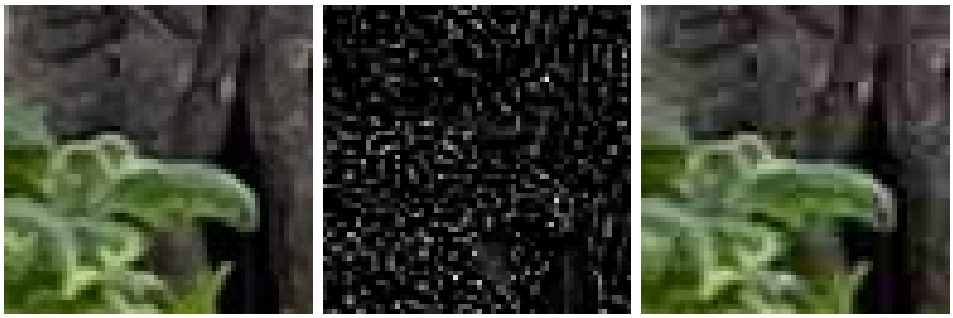
7 Wyniki działania modelu

Na kolejnych stronach przedstawione zostały wyniki działania optymalnej sieci o 4 warstwach, rozmiarze segmentu 256x256 oraz 8 pixelach przesunięcia przy rekonstrukcji.

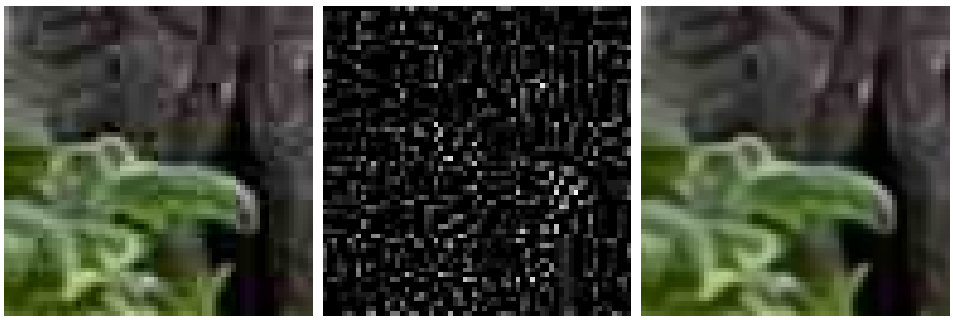
7.1 Roślinność na tle kamiennej ściany



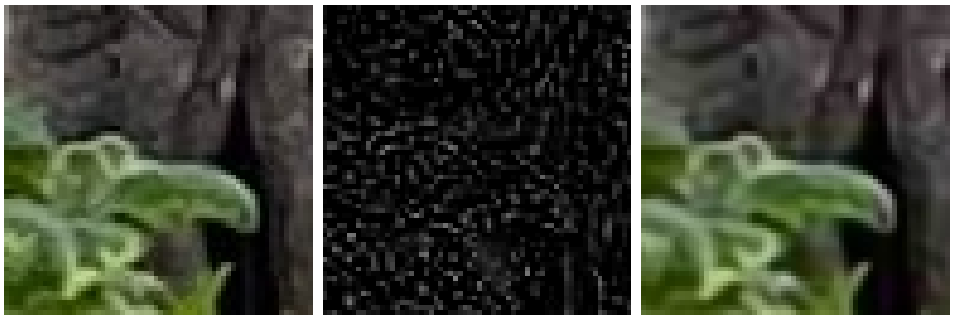
Rysunek 10: Segment oryginalny, skompresowany, oraz zrekonstruowany



Rysunek 11: Różnice między segmentem oryginalnym i skompresowanym



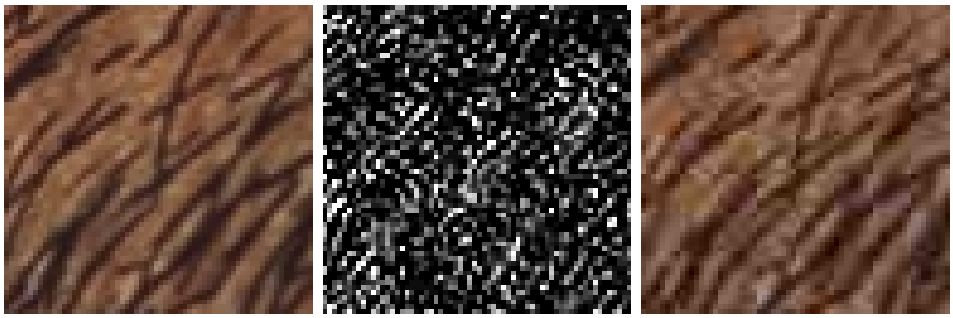
Rysunek 12: Różnice między segmentem skompresowanym i zrekonstruowanym



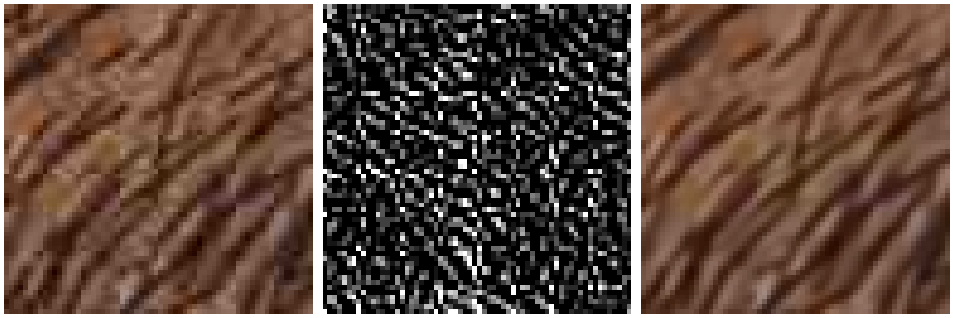
Rysunek 13: Różnice między segmentem oryginalnym i zrekonstruowanym

7.2 Zarost na twarzy

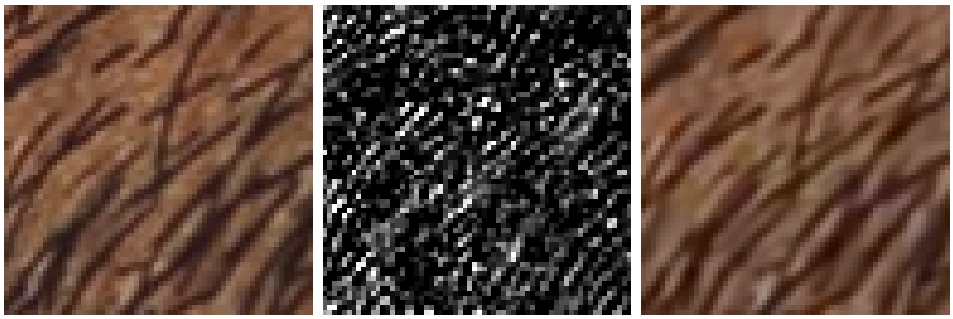
Rysunek 14: Segment oryginalny, skompresowany, oraz zrekonstruowany



Rysunek 15: Różnice między segmentem oryginalnym i skompresowanym



Rysunek 16: Różnice między segmentem skompresowanym i zrekonstruowanym



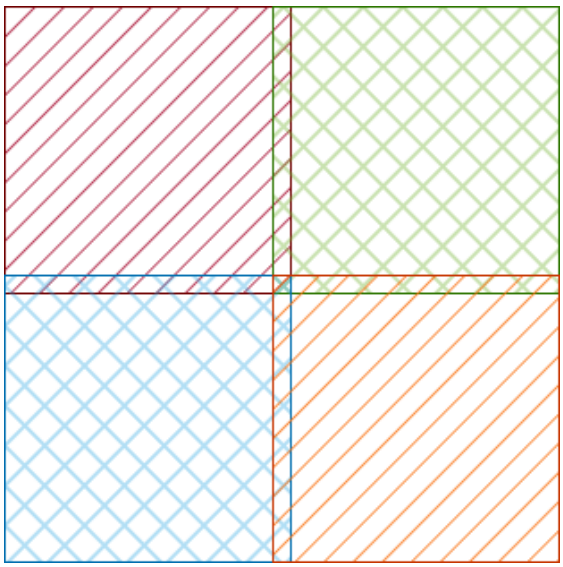
Rysunek 17: Różnice między segmentem oryginalnym i zrekonstruowanym

7.3 Spostrzeżenia

Zauważalna jest utrata detali względem segmentu oryginalnego, a nawet względem segmentu skompresowanego. Sieć ma cechy filtra uśredniającego. W porównaniach widoczna jest siatka kwadratów o boku 8 pixeli, wynikajaca ze sposobu działania algorytmu JPEG. Różnica między segmentem oryginalnym i zrekonstruowanym jest mniejsza niż między segmentem oryginalnym i skompresowanym oraz skompresowanym i zrekonstruowanym.

8 Rekonstrukcja pełnego obrazu

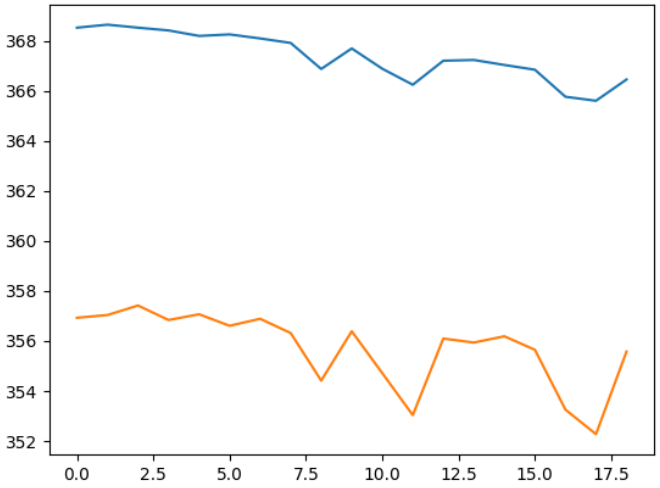
Sieć została przystosowana do przetwarzania monochromatycznych segmentów o rozmiarze 256x256 pixeli. Tworzy to potrzebę przystosowania rozwiązania do przetwarzania obrazów o dowolnym rozmiarze. Obraz zostaje podzielony na kanały RGB, a następnie, kolejne segmenty każdej z warstw są przetwarzane przez model. Wartości otrzymanej macierzy są zaokrąglane do liczb całkowitych oraz następuje 3yzacja wartości spoza zakresu 0-255. W przypadku segmentów na krawędziach obrazu, brakujący fragment zostaje uzupełniony lustrzanym odbiciem obrazu a po rekontrukcji odcięty. Z podejściem wiąże się problem, widocznych granic między przetworzonymi segmentami. Rozwiązaniem jest częściowe pokrycie segmentów przy przetwarzaniu oraz zastosowanie średniej ważonej na ich granicy. Wagi rzędów i kolumn z segmentu rosną w głąb segmentu.



Rysunek 18: Nachodzace na siebie segmenty

Pomimo zastosowania średniej ważonej, na granicy segmentów często pozostaje widoczna linnia. Problem ten można rozwiązać poprzez wybranie odpowiedniej wartości przesunięcia segmentów. Wpływ przenunięcia na obraz liczony był wartością średniego błędu bezwzględnego policzonego dla całego obrazu. Zauważyć można spadek wartości dla przesunięcia o 8 i 16 pixeli Rys. 19. Jest to związane z podziałem obrazu na obszary 8x8 przez algorytm JPEG. Jeżeli krawędzie powstałe w wyniku rekonstrukcji obrazu pokryją się z krawędziami powstającymi przy

kompresji, wynikowy błąd będzie mniejszy niż w przypadku, gdyby na obrazie tworzyły się dodatkowe granice. Dodatkowo widoczny jest spadek dla 11 pixeli, dla którego nie znam wyjaśnienia. Wartość 8 pixeli została wybrana jako optymalna, przy większych wartościach pojawia się ryzyko rozmazania obrazu przez średnią ważoną.



Rysunek 19: Wartość funkcji błędu dla różnych wartości przesunięcia

9 Wnioski

Można zastosować sieci typu U-Net do usuwania artefaktów kompresji stratnej. Takie rozwiązanie jest jednak bardzo wolne. Przetworzenie obrazu o wymiarach 3024x4032 pixeli zajmuje około 80 sekund z wykorzystaniem karty graficznej Radeon RX 5700 XT. Optymalne wydaje się zastosowanie sieci o 4 poziomach, przy rozmiarze segmentu wejściowego 256x256 pixeli. Przy rekonstrukcji obrazu należy zastosować przesunięcie 8 pixeli, aby granice między segmentami pokryły się z granicami powstałymi w wyniku działania algorytmu JPEG.

Literatura

[1] Stephan Antholzer, Markus Haltmeier, and Johannes Schwab. Deep learning for photoacoustic tomography from sparse data, 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Christopher G. Jennings. How jpeg works, 2017.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.