

Politechnika Wrocławskas

Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:
Aplikacja webowa zwiększająca
rozdzielncość obrazów

AUTOR:
Eryk Wójcik

PROMOTOR:
dr hab. inż. Andrzej Rusiecki,
Katedra Informatyki Technicznej

0

TODO: dodać do bibliografii framework i biblioteki użyte w apce TODO: Comic.png jest z Set14

Spis treści

1	Wstęp	3
2	Porównanie algorytmów ESRGAN i DWSR	5
2.1	Jakość odtwarzania obrazów	5
2.2	Analiza wydajności	9
2.3	Ograniczenia i wyzwania	10
3	Podsumowanie i wnioski	11
3.1	Dyskusja wyników	11
3.2	Rekomendacje i kierunki dalszych badań	11
	Bibliografia	12

Rozdział 1

Wstęp

Cel pracy

Opis celu badań, czyli stworzenia aplikacji webowej służącej do zwiększania rozdzielczości obrazów z użyciem algorytmów ESRGAN i DWSR oraz analiza i porównanie tych algorytmów.

Zakres pracy

Przedstawienie koncepcji i zagadnień, które zostaną omówione w pracy, w tym wybrane metody i technologie.

Rozdział 2

Porównanie algorytmów ESRGAN i DWSR

Skoro mamy narzędzie pozwalające korzystać algorytmów **DWSR** [2] i **ESRGAN** [3] do zadania super-rozdzielczości, to warto porównać te metody. W tym rozdziale zostanie przeprowadzona analiza porównawcza algorytmów, oceniona zostanie jakość rekonstruowanych obrazów, szybkość działania oraz złożoność implementacji algorytmów.

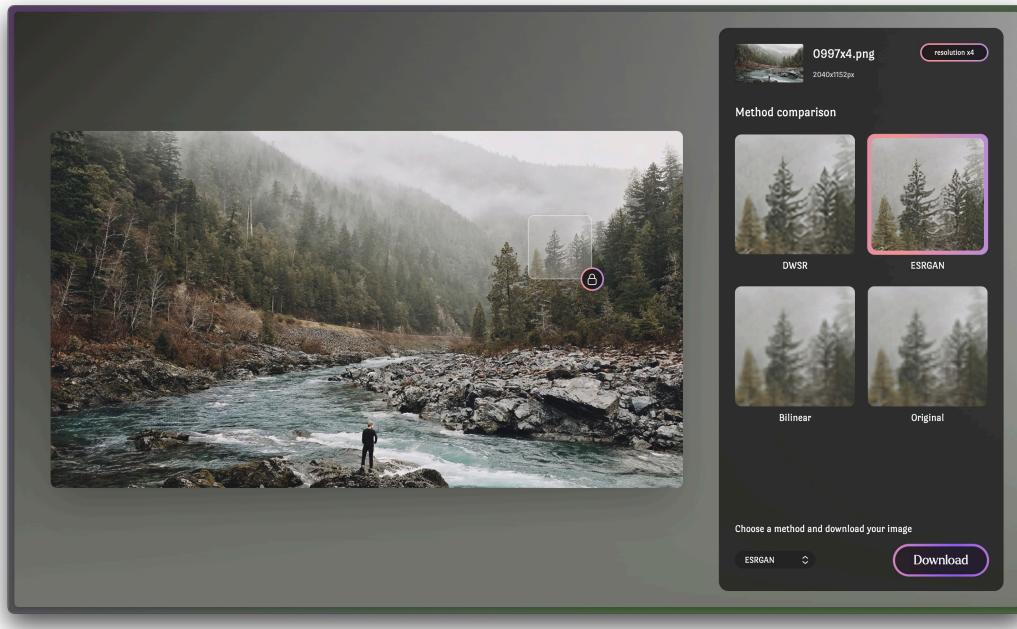
Do testów algorytmów wykorzystałem zestaw testowy z repozytorium **DWSR** [2]. Wybór padł na ten zestaw, gdyż obrazy te nie były wykorzystywane w treningu żadnej z implementowanych sieci. Obrazy w tym zestawie minimalnie różnią się od siebie rozdzielczością, co pozwala na dokładniejsze porównanie algorytmów zwłaszcza w kontekście analizy wydajności.

Na potrzebę wykonania testów wydajności algorytmów zostały wprowadzone zmiany w bazie danych w tabeli *Image*. Dodano pola *bilinear_time*, *dwsr_time* oraz *esrgan_time*, które przechowują czasy przetwarzania obrazu przez algorytmy. Czas przetwarzania został zmierzony również z etapami preprocessingu i postprocessingu obrazu, co ma znaczenie w przypadku algorytmu **DWSR**, który wymaga przeprowadzenia transformacji falkowej. Taki pomiar jest też bliższy rzeczywistości, gdyż w przypadku aplikacji webowej liczy się czas całego etapu przetwarzania obrazu, a nie samego powiększania.

2.1 Jakość odtwarzania obrazów

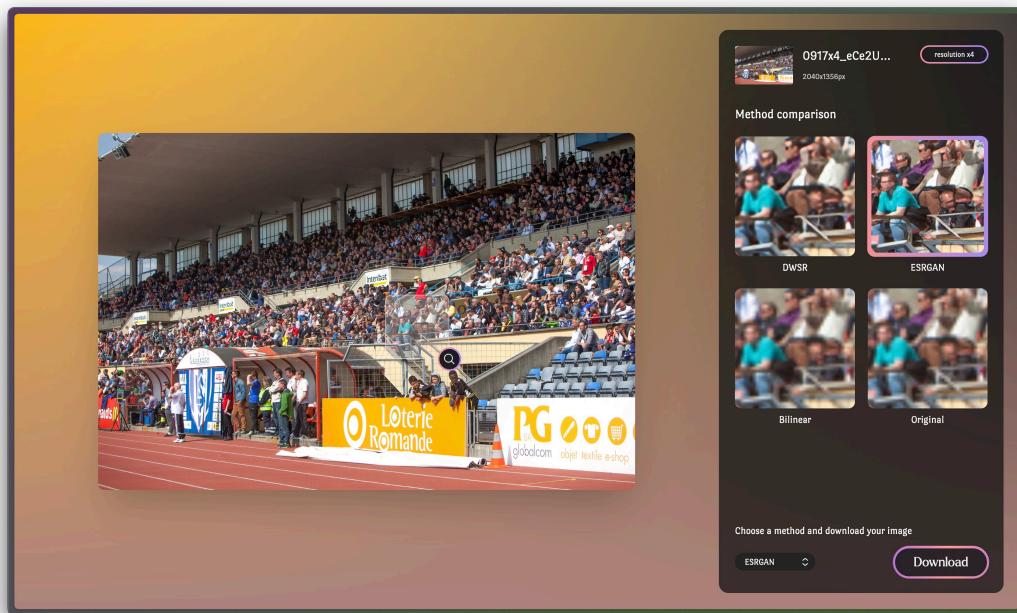
W tym rozdziale przedstawię wyniki testów jakości odtwarzania obrazów przez algorytmy **DWSR** i **ESRGAN**. Dokonałem selekcji obrazów - wybrałem te, które najlepiej obrazują różnice pomiędzy wynikami działania algorytmów.

Na obrazie 1. możemy sprawdzić jak algorytmy radzą sobie z rekonstrukcją tekstur drzew i liści. Są to detale wysokiej częstotliwości, które są trudne do odwzorowania, ale badane algorytmy potrafią sobie z nimi poradzić. Widać, że algorytm **DWSR** radzi sobie bardzo dobrze z zachowaniem kierunku tekstur, co zawdzięcza transformacji falkowej, lecz detale nie są odwzorowane idealnie. Algorytm **ESRGAN** dużo lepiej poradził sobie z odwzorowaniem detali i tekstur. Obrazy odwzorowane przez ten algorytm są bardzo ostre, miejscami aż nienaturalnie.



Rys 1. Obraz *0997x4.png* z zestawu testowego [2]

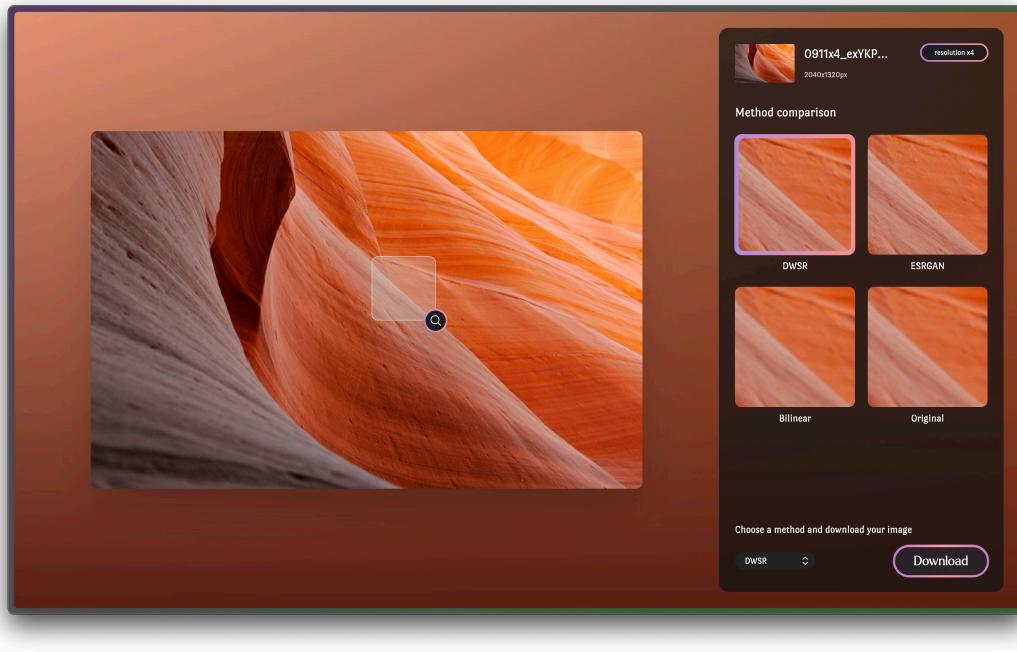
Na obrazie 2 możemy sprawdzić jak algorytmy radzą sobie z rekonstrukcją detali twarzy. Na tym przykładzie widać, że algorytm **ESRGAN** halucynuje detale twarzy i te wyglądają bardzo nienaturalnie, jednak brąz wygenerowany przez ten algorytm jest bardzo ostry. Algorytm **DWSR** radzi sobie dużo lepiej z zachowaniem naturalnego wyglądu twarzy, lecz nie odwzorowuje detali tak dobrze jak **ESRGAN**.



Rys 2. Obraz *0917x4.png* z zestawu testowego [2]

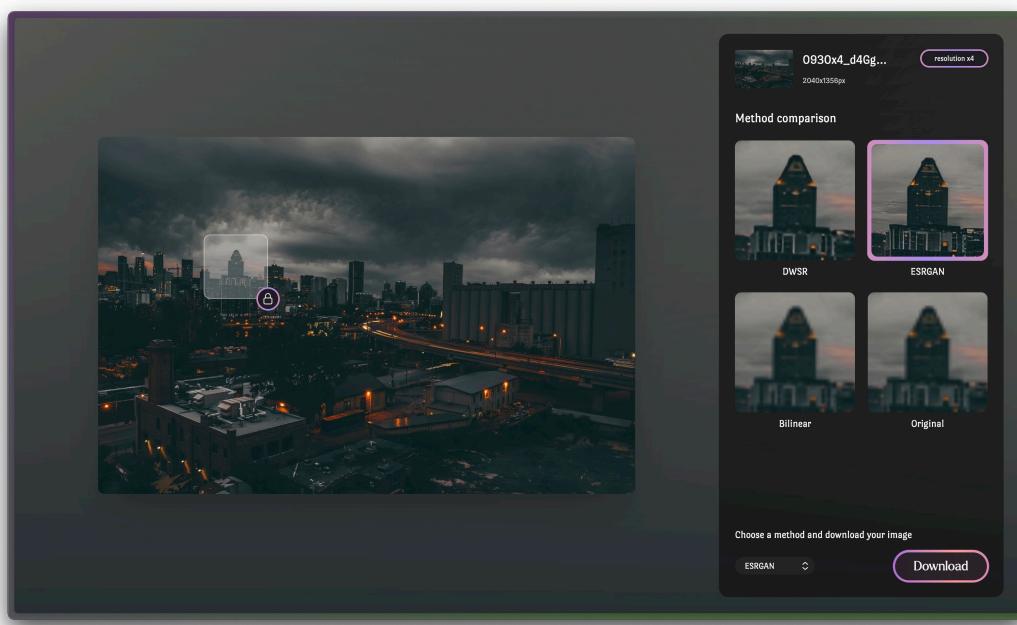
Na obrazie 3 możemy się przekonać jak algorytmy radzą sobie z powiększeniem obrazu, na którym zdecydowanie dominują niskie częstotliwości, ale występują też krzywe z ostrzejszymi krawędziami. Tutaj świetnie sprawdził się algorytm **DWSR**, który zachował

naturalny wygląd obrazu, dodatkowo zachowując ostrość krawędzi. Algorytm **ESRGAN** również poradził sobie bardzo dobrze. Na tego typu obrazach algorytm **Bilinear** (interpolacja dwuliniowa), również radzi sobie przyzwoicie, ale niestety krawędzie w obrazie wynikowym są bardzo rozmyte.



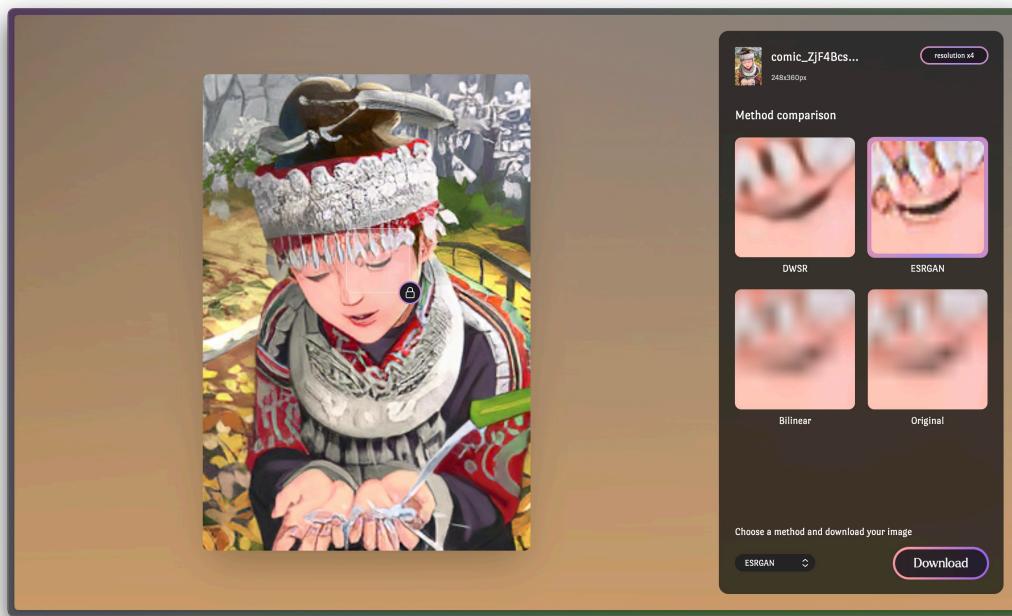
Rys 3. Obraz *0911x4.png* z zestawu testowego [2]

Obraz testowy 4 jest bardzo dobrym przykładem scenerii z dużą ilością detali i tekstur. Algorytm **ESRGAN** bezbłędnie poradził sobie z rekonstrukcją detali takich jak budynki, czy ulice. Algorytm **DWSR** również poradził sobie dobrze, jednak obraz nie jest nawet bliski ostrości jaką osiągnął algorytm **ESRGAN**.



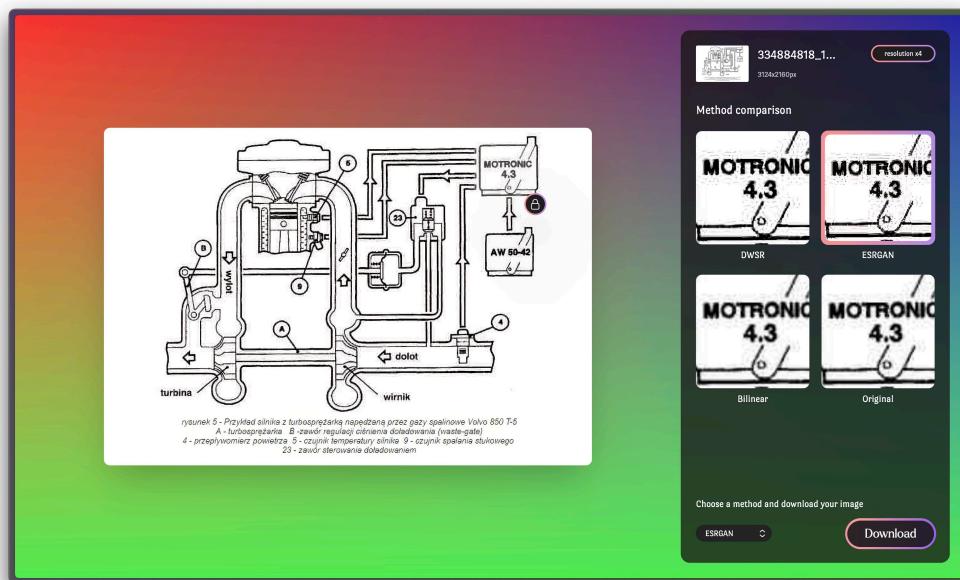
Rys 4. Obraz *0930x4.png* z zestawu testowego [2]

Obraz testowy 5 jest przykładem działania algorytmów, gdy mamy do czynienia z obrazem o bardzo niskiej rozdzielczości. Algorytm **ESRGAN** poradził sobie bardzo dobrze ze stworzeniem detali. Dokładnie zarysowane są wszystkie linie, ale obraz miejscami odbiega od oryginału, widać to między innymi na oku postaci. Algorytm **DWSR** poradził sobie dobrze, odtworzenie detali jest bliskie oryginałowi ale obraz jest bardzo rozmyty.



Rys 5. Obraz *comic.png* z zestawu testowego Set14 [4]

Ostatnim przykładem będzie rysunek techniczny z obrazu 6. Algorytm **ESRGAN** jest bardzo wrażliwy na kompresję jpg, w wyniku czego obraz wygląda nieprzyjemnie a czcionki są zbyt ostre. Algorytm **DWSR** poradził sobie dużo lepiej z rekonstrukcją obrazu, czcionki są wyraźne i czytelne, a obraz nie jest mocno zniekształcony.

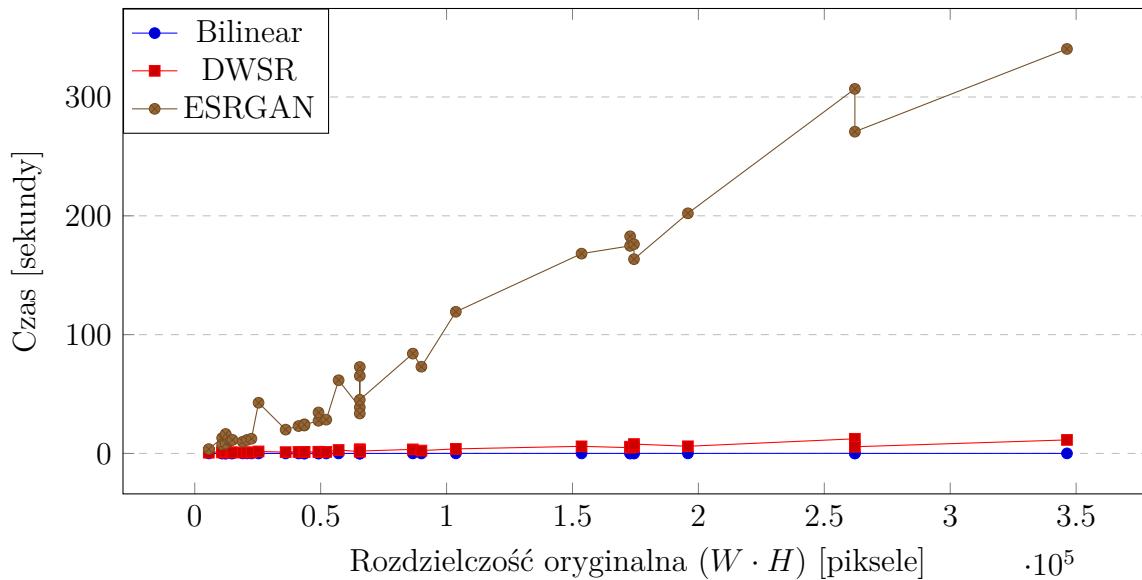


Rys 6. Schemat techniczny silnika z turbosprężarką ze strony [1]

2.2 Analiza wydajności

Na podstawie rozdziałów traktujących o strukturach algorytmów **DWSR** i **ESRGAN** można zauważać, że algorytm **ESRGAN** jest dużo bardziej skomplikowany. W związku z tym można przypuszczać, że algorytm **DWSR** będzie działał szybciej. W tym rozdziale sprawdzę jak wygląda wydajność obu algorytmów.

Do testów wydajności wykorzystałem obrazy z zestawu testowego [2] oraz [4]. Testy zostały przeprowadzone na komputerze z procesorem **Intel Core i7-9750H** z systemem **MacOS**, więc niestety nie mogłem wykorzystać GPU do przyspieszenia obliczeń, ponieważ biblioteka **PyTorch** nie wspiera kart graficznych **AMD**, w której wyposażony jest sprzęt.



Rys 7. Wykres czasu przetwarzania obrazów przez algorytmy w zależności od rozdzielczości

Jak widać na wynikach przedstawionych na wykresie 7, algorytm **Bilinear** działa najszybciej, ale nie jest to zaskoczeniem, ponieważ jest to najprostsza metoda. Algorytm **DWSR** działa zdecydowanie szybciej niż **ESRGAN** i moim zdaniem do w wielu przypadkach jest wystarczający. Widzimy tutaj kolejną zaletę korzystania z współczynników falkowych, których estymacja jest zdecydowanie szybsza niż rekonstrukcja całego obrazu przez sieć neuronową.

Warto zaznaczyć, że badania wydajnościowe na wykresie 7 zawierają czas działania nie tylko samego powiększania obrazów, ale również czas *preprocessingu* i *postprocessingu* obrazów, co ma znaczenie w przypadku algorytmu **DWSR**, który wymaga przeprowadzenia transformacji falkowej i odwróconej transformacji falkowej.

Algorytm **ESRGAN** natomiast był dużo łatwiejszy w implementacji, jedynym wymaganiem było zainstalowanie wspomnianych w rozdziale ?? bibliotek. Algorytm **DWSR** wymagał nie tylko implementacji biblioteki **PyWavelets** do przeprowadzenia transformacji falkowej, ale również przepisania programów do pre i postprocessingu z języka **MATLAB** do **Pythona**.

2.3 Ograniczenia i wyzwania

Obydwa algorytmy mają swoje wady i zalety. Algorytm **DWSR** jest dużo szybszy, ale nie jest w stanie odwzorować detali tak dobrze jak **ESRGAN**. **ESRGAN** jest dużo wolniejszy, zdecydowanie lepiej odwzorowuje szczegóły, ale często robi to w sposób nienaturalny, czego efektem mogą być obrazy wyglądające mniej przyjemnie niż **DWSR**.

Zauważałem również problem z aliasingiem i nienaturalną ostrością na obrazach wygenerowanych przez **ESRGAN**. Problemy nie występują nagminnie, ale gdy występują sprawiają, że obraz wygląda sztucznie.

Algorytmy równie dobrze radzą sobie z zaszumionymi obrazami, ale w przypadku obrazów mocno znieksztalconych przez kompresję, algorytm **ESRGAN** radzi sobie dużo gorzej, gdyż podbija te fragmenty. Algorytm **DWSR** radzi sobie dużo lepiej, gdyż nie podbija tak mocno znieksztalconych fragmentów, ale osłabia ich widoczność.

Rozdział 3

Podsumowanie i wnioski

3.1 Dyskusja wyników

Krytyczna analiza uzyskanych wyników w kontekście celów pracy oraz istniejących badań i literatury w dziedzinie.

3.2 Rekomendacje i kierunki dalszych badań

Sugestie dotyczące potencjalnych ulepszeń i obszarów, które wymagają dalszych badań, w oparciu o obserwacje i wyniki badań.

Literatura

- [1] D. Bukowski. Doładowanie silników spalinowych. https://www.zssplus.pl/transport/referaty/referat_07.htm, 2004/2005.
- [2] T. Guo, H. S. Mousavi, T. H. Vu, V. Monga. Deep wavelet prediction for image super-resolution. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.
- [3] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C. C. Loy. Esrgan: Enhanced super-resolution generative adversarial networks. *The European Conference on Computer Vision Workshops (ECCVW)*, September 2018.
- [4] R. Zeyde, M. Elad, M. Protter. On single image scale-up using sparse-representations. *International conference on curves and surfaces*, strony 711–730. Springer, 2010.