Zeszyty Naukowe WWSI, No 29, Vol. 17, 2023, s. 105-124
DOI: 10.26348/znwwsi.29.105

Wykorzystanie wstępnie wytrenowanych konwolucyjnych sieci neuronowych w zagadnieniu transferu stylu

Weronika Gramacka*

Wojskowa Akademia Techniczna, Warszawa, Polska

Streszczenie

Artykuł dotyczy wykorzystania wstępnie wytrenowanych konwolucyjnych sieci neuronowych w zagadnieniu transferu stylu. Zbadane zostało, jak wybór warstw konwolucyjnych do reprezentacji obrazów stylu i zawartości wpływa na jakość odwzorowania stylu: kolorystykę generowanego obrazu, widoczną technikę jego wykonania oraz poziom uogólnienia szczegółów przedstawianej rzeczywistości. Zaproponowano architekturę konwolucyjnej sieci neuronowej dedykowaną rozważanemu zagadnieniu.

Słowa kluczowe – konwolucyjne sieci neuronowe, transfer stylu neuronowego, sieci wstępnie wytrenowane

* E-mail: gramackaweronika@gmail.com

Zgłoszono do druku 29 listopada 2023 r.

1. Wprowadzenie

Wzrok jest dla człowieka jednym z głównych zmysłów poznawania i analizowania otaczającego go świata. Rozpoznawanie przedmiotów, znajdowanie obiektów na obrazach, zarówno w świecie rzeczywistym, jak również będących wynikiem przekształceń artystycznych, nie sprawia człowiekowi trudności. Badania nad działaniem ludzkiego mózgu zainspirowały naukowców do prac nad stworzeniem komputerowego odwzorowania procesów zachodzących w mózgu.

W roku 1958 Frank Rosenblatt [1] przedstawił pierwszy model perceptronu (najprostszej, jednowarstwowej, sieci neuronowej), który zapoczątkował prace nad bardziej skomplikowanymi strukturami, z biegiem czasu coraz bardziej przypominającymi sieci neuronów w biologicznym systemie nerwowym. W roku 1974 Paul Werbos opracował algorytm wstecznej propagacji [2], który został „ponownie odkryty" w roku 1986, stając się podstawowym algorytmem uczenia wielowarstwowych sieci neuronowych [3,4].

W roku 2006 Reofrey Hinton i Ruslan Salakhutdinov [5] zaproponowali nową skuteczną metodę uczenia sieci wielowarstwowych – warstwa po warstwie, a następnie nadzorowane douczenie, co przyczyniło się do dalszego intensywnego rozwoju tych sieci.

Przetwarzanie obrazów w sposób podobny do ludzkiego mózgu stało się możliwe po roku 1998, kiedy to Yann LeCun z grupą badaczy [6] zaproponowali pierwszy model sieci splotowej (konwolucyjnej). Obecnie sieci konwolucyjne stały się podstawowym narzędziem wykorzystywanym w problemach analizy i syntezy obrazów. Analiza obrazów dotyczy procesu ekstrakcji informacji z obrazu. Jej wynikiem nie jest obraz, ale dane w postaci numerycznej lub symbolicznej. Przykładem zastosowań analizy obrazów może być przetwarzanie medyczne wspomagające diagnostykę, rozpoznawanie wzorców w systemach kontroli jakości czy rozpoznawanie twarzy w systemach monitoringu. Synteza obrazów oznacza generowanie nowych obrazów na podstawie dostarczonych danych wejściowych, w tym innych obrazów. Przykładem może tu być generowanie map pogodowych, tekstur dla gier komputerowych, trójwymiarowych obiektów dla symulacji naukowych.

Jednym z zagadnień syntezy obrazów jest transfer stylu, gdzie wynikiem jest obraz łączący cechy stylu jednego obrazu z zawartością innego. Zagadnienie to poja-

wiło się w roku 2015 w artykule [7], w którym zaproponowano metodę wykorzystującą konwolucyjną sieć neuronową do transferu stylu między dwoma obrazami, w ten sposób, że zdjęcie obiektu rzeczywistego, np. krajobrazu, zostaje przekształcone na obraz artystyczny przedstawiający ten obiekt „namalowany" w stylu konkretnego malarza, np. Van Gogha. Zagadnienia tego typu występują przy rekonstrukcji zdjęć, w produkcjach multimedialnych przy tworzeniu specjalnych efektów video, projektowaniu interfejsów użytkownika, a także podczas przetwarzania obrazów medycznych [8].

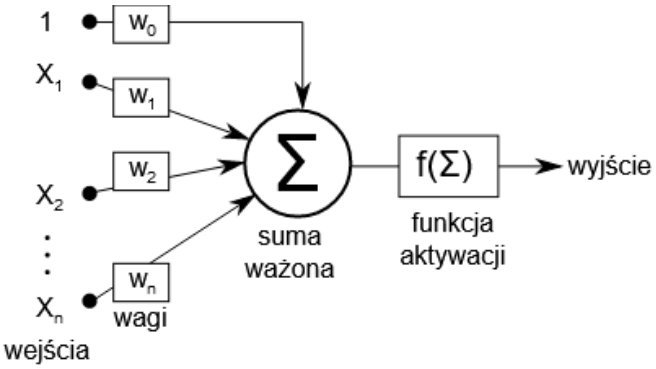
Celem niniejszego artykułu jest przedstawienie wykorzystania konwolucyjnych sieci neuronowych, w szczególności sieci wstępnie wytrenowanych, w zagadnieniach transferu stylu. Przeprowadzony został eksperyment obliczeniowy, którego wyniki pokazały, jak użycie różnych warstw z modelu sieci konwolucyjnych do reprezentacji obrazów stylu i zawartości wpływa na jakość odwzorowania stylu: kolorystykę generowanego obrazu, widoczną technikę jego wykonania oraz poziom uogólnienia szczegółów przedstawianej rzeczywistości. W eksperymencie wykorzystano dwie powszechnie stosowane sieci: VGG19 i EfficientNet-B0, a także sieć, której architekturę specjalnie zaprojektowano dla transferu stylu.

2. Sztuczne sieci neuronowe

Inspiracją do powstania sztucznych sieci neuronowych były biologiczne sieci neuronowe występujące w ludzkim mózgu, które składają się z komórek nerwowych – neuronów i połączeń między nimi – synaps. Pojedynczy neuron otrzymuje sygnały wejściowe od połączonych z nim innych neuronów. Sygnały te są odpowiednio sumowane w ciele komórki, po czym sygnał wyjściowy jest przekazywany z odpowiednią siłą do kolejnego neuronu. Sztuczna sieć neuronowa jest uproszczonym modelem matematycznym mózgu. Przetwarza ona dane z wykorzystaniem sztucznych neuronów, ułożonych w połączone między sobą warstwy.

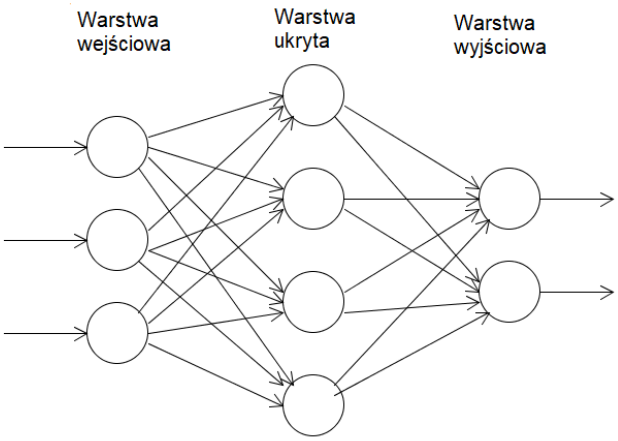
Pierwszy model neuronu, nazywany modelem McCullocha-Pittsa [9], powstał w roku 1943 (rysunek 1). Na wejście neuronu podawane są wartości wejściowe, będące danymi charakterystycznymi dla danego zagadnienia. Każda z wartości mnożona jest przez odpowiednią wagę będącą liczbą rzeczywistą, następnie trafia do wnętrza neuronu pełniącego rolę jednostki sumacyjnej. Suma wymnożonych przez wagi wartości wejściowych jest powiększana o wartość wyrazu wolnego, po czym trafia do

bloku funkcji aktywacji, która przekształca go na odpowiednią wartość wyjściową, najczęściej z przedziału $[0, 1]$ lub $[-1, 1]$.



Rysunek 1. Model neuronu McCullocha-Pittsa [10]

Neurony są rozmieszczone w warstwach. Zwykle sieć neuronowa zawiera warstwę wejściową, co najmniej jedną warstwę ukrytą oraz warstwę wyjściową (rysunek 2).



Rysunek 2. Sieć neuronowa trzy warstwowa

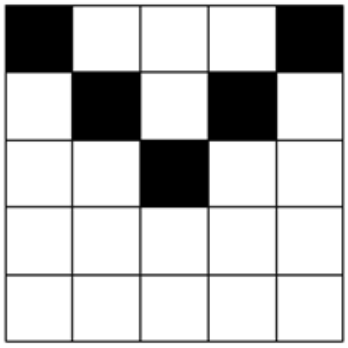
Warstwa wejściowa otrzymuje dane i przekazuje je do poszczególnych neuronów, a każda cecha wejściowa reprezentowana przez jest jeden neuron. Neurony warstwy ukrytej przetwarzają dane za pomocą wag i funkcji aktywacji. Każdy neuron tej warstwy otrzymuje dane od wszystkich neuronów poprzedniej warstwy i przekazuje

dane do wszystkich neuronów następnej warstwy. Wagi neuronów wyznaczane są w iteracyjnym procesie uczenia się sieci tak, aby dostosować model do rozwiązywanego zagadnienia. Warstwa wyjściowa podaje wynik końcowy.

Wykorzystanie wielu warstw ukrytych w sieci neuronowej zwiększa jej możliwości reprezentowania skomplikowanych zależności między danymi. Warstwy najniższe (znajdujące się blisko wejścia) na ogół rozpoznają pewne ogólne cechy danych wejściowych, natomiast warstwy wyższe określają wzajemne powiązania między tymi cechami. Proces uczenia się wielowarstwowej sieci neuronowej nazywany jest uczeniem głębokim.

3. Reprezentacja cyfrowa dwuwymiarowych danych graficznych

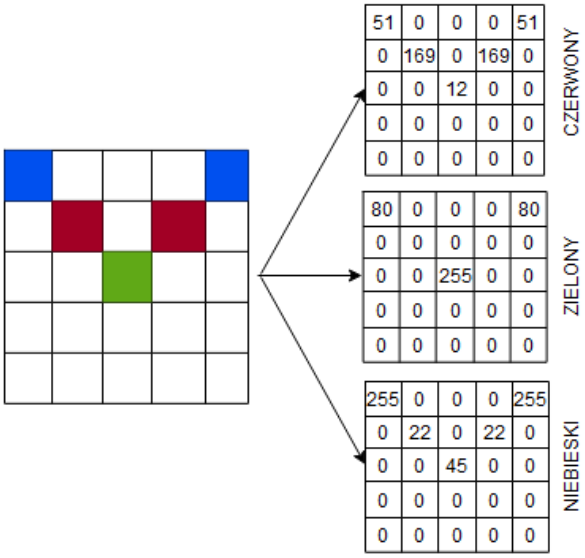
Reprezentacje graficzne elementów rzeczywistości nazywane są obrazami. W komputerze, obrazy reprezentowane są jako macierze dwuwymiarowe o określonej szerokości i wysokości, zbudowane z jednostek nazywanych pikselami, z których każdy przechowuje informację o swoim kolorze. Obraz czarno-biały jest reprezentowany przez jedną macierz, w której każda jednostka określa jasność piksela za pomocą liczby z przedziału [0, 255] (rysunek 3).



255	0	0	0	255
0	255	0	255	0
0	0	255	0	0
0	0	0	0	0
0	0	0	0	0

Rysunek 3. Przykład reprezentacji obrazu czarno-białego o wymiarach 5x5 pikseli

Obraz kolorowy jest reprezentowany przez trzy macierze, z których każda zawiera informację o jednym z trzech kolorów podstawowych: czerwonym, zielonym i niebieskim (rysunek 4).



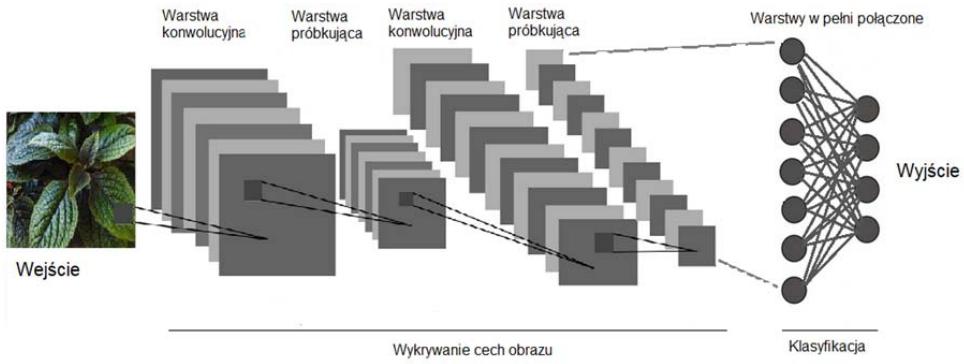
Rysunek 4. Przykład reprezentacji obrazu kolorowego o wymiarach 5x5 pikseli

Liczba pikseli, z których składa się obraz, określa jego rozdzielczość. Obrazy o małej rozdzielczości są mniej szczegółowe i mają wyraźnie widoczne poszczególne piksele. Obrazy o wysokiej rozdzielczości są bardziej zbliżone do rzeczywistości, jednak zajmują dużo więcej pamięci, przez co ich przetwarzanie jest trudniejsze i dużo dłuższe (rysunek 5).

Rysunek 5. Obraz w wysokiej (3000x2250 pikseli) i niskiej (100x75 pikseli) rozdzielczości

4. Przetwarzanie obrazu przy pomocy konwolucyjnych sieci neuronowych

Konwolucyjne sieci neuronowe (ang. *Convolutional Neural Networks*, CNN) lepiej niż inne rodzaje sieci neuronowych radzą sobie z zagadnieniami przetwarzania obrazów, mowy lub sygnałów dźwiękowych [11,12,13]. Można w nich wyróżnić trzy podstawowe typy warstw: warstwę splotową (konwolucyjną), warstwę próbkującą (poolingową) i warstwę w pełni połączoną. Typowa architektura konwolucyjnej sieci neuronowej przedstawiona jest na rysunku 6.



Rysunek 6. Typowa architektura konwolucyjnej sieci neuronowej

Pierwszą warstwą CNN jest warstwa konwolucyjna, za nią znajdują się następne warstwy konwolucyjne, przeplatane warstwami próbkującymi. Warstwą końcową jest warstwa w pełni połączona. Wcześniejsze warstwy konwolucyjne koncentrują się na prostych cechach, takich jak kolory i krawędzie. Warstwy dalsze mają za zadanie wydobywać coraz bardziej złożone cechy i struktury, a także je interpretować. Warstwy w pełni połączone, przetwarzając wyniki z warstw poprzednich, generują ostateczny wynik, na przykład w zagadnieniu klasyfikacji przypisanie obiektu przedstawionego na obrazie do konkretnej klasy. W zagadnieniach transferu stylu neuronowego warstwa w pełni połączona nie jest wykorzystywana.

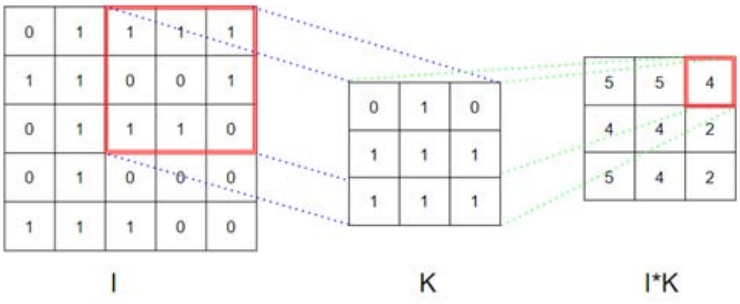
4.1. *Operacja konwolucji*

Załóżmy, że na wejściu CNN dany jest obraz reprezentowany przez macierz pikseli (lub 3 macierze odpowiadające kolorom podstawowym, jeżeli obraz jest kolorowy). Pierwsza warstwa konwolucyjna składa się neuronów ułożonych w siatkę, w której każdy neuron połączony jest tylko z pikselami pewnego obszaru przetwarzanego obrazu, nazywanego polem recepcyjnym tego neuronu. Analogicznie neurony w kolejnych warstwach konwolucyjnych połączone są tylko z neuronami poprzedniej warstwy znajdującymi się w ich polach recepcyjnych.

Załóżmy również, że mamy detektor cech (zwany filtrem lub jądrem) w postaci dwuwymiarowej macierzy wag (zwykle o rozmiarach 5x5 lub 3x3). Filtr jest przesuwany z pewnym krokiem przez obszar recepcyjny danych wejściowych wzdłuż obu wymiarów. Dla każdej lokalizacji filtra, obliczane są sumy iloczynów punktowych między elementami filtra a odpowiadającymi im elementami danych wejściowych, czyli wykonywana jest operacja konwolucji. Jeżeli oznaczymy przez I macierz danych wejściowych, przez K macierz filtra o rozmiarze MxM, to operacja konwolucji dla lokalizacji (m, n) zdefiniowana jest wzorem (1):

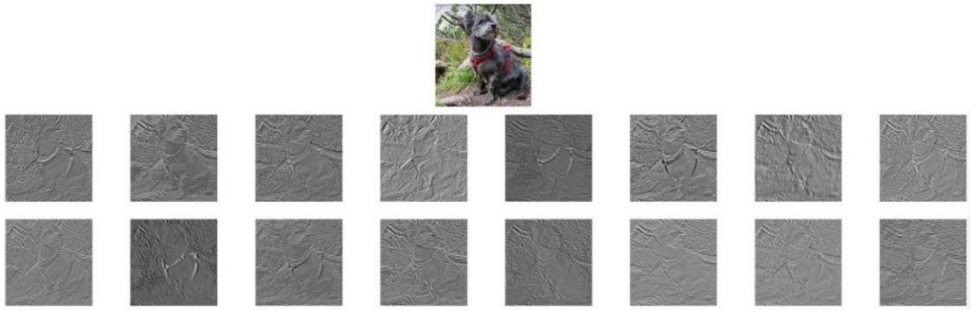
$$(I * K)(m,n) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} I(m+i, n+i) K(i,j) \tag{1}$$

Przykład operacji konwolucji z filtrem 3x3 pokazany jest na rysunku 7.



Rysunek 7. Przykład operacji konwolucji

Wagi filtra pozostają niezmienne podczas przesuwania się po obrazie. Wynikiem operacji konwolucji dla wszystkich lokalizacji filtra w danych wejściowych jest mapa cech, która reprezentuje przetworzone informacje o obrazie. Poszczególne warstwy konwolucyjne składają się z wielu map cech o identycznych rozmiarach. Przykładowe mapy cech pokazane są na rysunku 8.



Rysunek 8. Obraz wejściowy i przykładowe mapy cech

Wagi filtra są dostosowywane w procesie uczenia się sieci tak, aby minimalizować określoną funkcję straty, która mierzy, jak bardzo predykcje sieci różnią się od rzeczywistych wartości docelowych na zbiorze treningowym. Najczęściej stosowany jest w tym celu zmodyfikowany algorytm propagacji wstecznej albo algorytm stochastycznego spadku gradientu.

4.2. *Warstwa próbkująca*

Warstwa próbkująca ma za zadanie zredukować rozmiar przestrzenny danych i uogólnić informacje dotyczące danej cechy wydobyte przez filtry warstw konwolucyjnych. Neurony w warstwie redukującej nie posiadają wag, na wyjściu zwracają wartość bazującą bezpośrednio na wartościach wejściowych, np. maksymalną wartość spośród wartości wejściowych neuronu (*max-pooling*) lub średnią arytmetyczną wartości wejściowych (*average-pooling*).

Pomiędzy warstwami konwolucyjnymi a warstwami próbkującymi znajduje się warstwa wprowadzająca do modelu nieliniowość (warstwa aktywacji). Ma ona często postać transformacji ReLu (ang. *Rectified Linear Unit*) określonej wzorem:

$ReLu(x) = max\{0, x\}$. Bez wartstwy aktywacji głęboka sieć konwolucyjna przekształciłaby się w pojedynczą warstwę konwolucyjną o gorszym działaniu.

5. Transfer stylu

Transfer stylu to technika przetwarzania obrazów, polegająca na wykorzystaniu wytrenowanej wcześniej sieci neuronowej do przeniesienia charakterystycznych cech artystycznego stylu jednego obrazu na zawartość innego obrazu.

5.1. *Ogólny schemat procesu transferu stylu*

Aby przeprowadzić transfer stylu potrzebne są dwa obrazy wejściowe: obraz zawartości i obraz stylu oraz konwolucyjna sieć neuronowa, która będzie służyć do ekstrakcji odpowiednich cech z obydwu obrazów.

W sieci konwolucyjnej warstwy niższe (bliższe wejścia) odpowiadają najczęściej za wykrywanie bardziej ogólnych cech danego obrazu, takich jak krawędzie, barwy, proste różnice w tekturze, natomiast warstwy wyższe (bardziej oddalone od warstwy wejściowej) najczęściej odpowiadają za wykrywanie bardziej szczegółowych cech obrazu, takich jak głębia i detale. Dlatego też wyższe warstwy mogą zostać wykorzystane do reprezentacji treści obrazu zawartości. Aby uzyskać informację o stylu danego obrazu, wykorzystane zostaną niższe warstwy sieci.

Połączenie obrazu reprezentującego treść i obrazu reprezentującego styl spowoduje pewne utraty zarówno w reprezentacji treści jak i stylu. Aby uzyskać najlepszy obraz wyjściowy, konieczne jest zminimalizowanie utraty całkowitej, L_{total}, którą można wyrazić następującym wzorem:

$$L_{total} = \alpha L_{content} + \beta L_{style} \tag{2}$$

gdzie $L_{content}$ i L_{style} oznaczają utraty, odpowiednio, zawartości i stylu, α i β są parametrami określającymi wpływ obrazu zawartości i obrazu stylu na obraz generowany (wyższa wartość współczynnika α sprawia, że model bardziej „troszczy się" o zawartość, natomiast wyższa wartość β powoduje większe dopasowanie stylu).

5.2. *Utrata zawartości*

Dla danej warstwy CNN, funkcję utraty zawartości możemy zdefiniować jako błąd średniokwadratowy między cechami obrazu zawartości i obrazu generowanego:

$$L_{content} = \frac{1}{2}\sum_{i,j,k}\left(C_{ijk}^{gen} - C_{ijk}^{content}\right)^2 \tag{3}$$

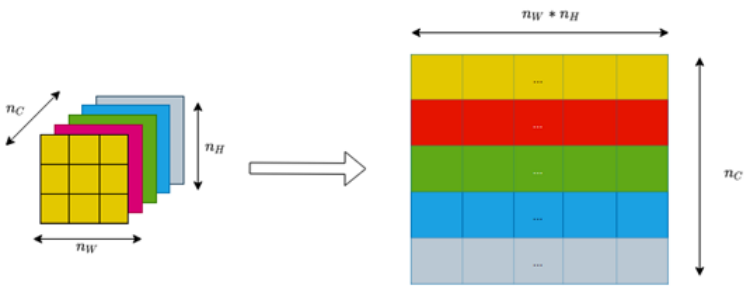
gdzie C_{ijk}^{gen} i $C_{ijk}^{content}$ oznaczają wartości na pozycji (i, j, k) dla, odpowiednio, obrazu wygenerowanego i referencyjnego obrazu zawartości; para (i, j) reprezentuje współrzędne przestrzenne, a k jest indeksem mapy cech. Minimalizacja funkcji utraty zawartości odbywa się z wykorzystaniem propagacji wstecznej.

5.3. *Utrata stylu*

Pojęcie stylu odnosi się do cech artystycznych obrazu – obejmuje jego teksturę, paletę kolorów oraz przestrzenne rozmieszczenie elementów wizualnych, które nadają obrazowi jego unikalny artystyczny charakter. Aby uchwycić cechy stylu z wybranej warstwy, wykorzystywana jest macierz Grama, która reprezentuje korelacje pomiędzy poszczególnymi cechami [7]. Macierz Grama jest wyznaczana z poniższego wzoru:

$$G = F \cdot F^T \tag{4}$$

gdzie F jest dwuwymiarową macierzą, w której pojedynczy wiersz reprezentuje spłaszczoną mapę cech (rysunek 9).



Rysunek 9. Przekształcenie map cech danej warstwy do postaci macierzowej

Wartość utraty stylu jest obliczana jako suma kwadratów różnic pomiędzy odpowiadającymi sobie elementami macierzy Grama obrazu generowanego i referencyjnego obrazu stylu według następującego wzoru:

$$L_{style} = \frac{1}{(2n_C n_W n_H)^2} \sum_{i,j} \left(G_{ij}^{gen} - G_{ij}^{style} \right)^2 \tag{5}$$

gdzie n_C jest liczbą map cech w danej warstwie, n_W i n_H oznaczają szerokość i wysokość mapy cech.

6. Transfer stylu przy wykorzystaniu wstępnie wytrenowanych konwolucyjnych sieci neuronowych VGG19 i EfficientNet-B0

Wstępnie wytrenowane sieci konwolucyjne to modele, które zostały przeszkolone na dużych zbiorach danych, zazwyczaj na zadaniach klasyfikacji obrazów. Modele te są dostępne do użycia w innych zadaniach, często jako podstawowe sieci do ekstrakcji cech.

6.1. *Sieć VGG19*

Konwolucyjna sieć neuronowa VGG19 powstała w roku 2014 na Uniwersytecie Oksfordzkim [14]. Trenowana była na ponad 1 milionie kolorowych obrazów o wymiarach 224x224 pikseli z bazy danych ImageNet. Sieć jest w stanie rozpoznawać 1000 różnych kategorii.

Sieć VGG19 składa się z 26 warstw, w tym 16 warstw konwolucyjnych połączonych w 5 bloków. W każdej warstwie konwolucyjnej wykorzystane zostało jądro konwolucji o bardzo małym wymiarze 3x3, co pozwoliło na uzyskanie głębokiej architektury oraz na zmniejszenie liczby parametrów modelu, co znacząco wpłynęło na ograniczenie problemu przetrenowania sieci.

Architektura sieci VGG19 została pokazana w tabeli 1.

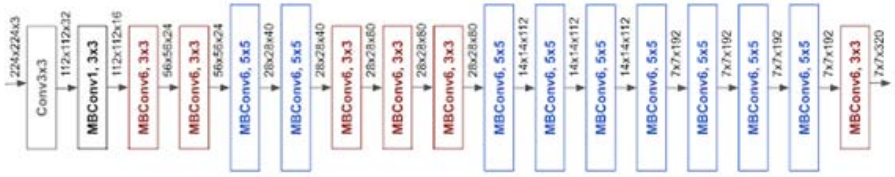
Tabela 1. Architektura sieci VGG19

Nazwa warstwy	**Typ warstwy**	**Rozmiar wyjściowy (w pikselach)**
Input	InputLayer	224x224x3
block1_conv1	Conv2D	224x224x64
block1_conv2	Conv2D	224x224x64
block1_pool	MaxPooling2D	112x112x64
block2_conv1	Conv2D	112x112x128
block2_conv2	Conv2D	112x112x128
block2_pool	MaxPooling2D	56x56x128
block3_conv1	Conv2D	56x56x256
block3_conv2	Conv2D	56x56x256
block3_conv3	Conv2D	56x56x256
block3_conv4	Conv2D	56x56x256
block3_pool	MaxPooling2D	28x28x256
block4_conv1	Conv2D	28x28x512
block4_conv2	Conv2D	28x28x512
block4_conv3	Conv2D	28x28x512
block4_conv4	Conv2D	28x28x512
block4_pool	MaxPooling2D	28x28x512
block5_conv1	Conv2D	14x14x512
block5_conv2	Conv2D	14x14x512
block5_conv3	Conv2D	14x14x512
block5_conv4	Conv2D	14x14x512
block5_pool	MaxPooling2D	7x7x512
Flatten	Flatten	25088
fc1	Dense	4096
fc2	Dense	4096
prediction	Dense	1000

6.2. *Sieć EfficientNet-B0*

Sieć EfficientNet-B0 jest jednym z modeli w rodzinie sieci EfficientNet, które zostały zaprezentowane przez Google w roku 2019 [15]. Sieć była trenowana na obrazach z bazy danych ImageNet o wymiarach 224x224 pikseli.

EfficientNet-B0 składa się łącznie z 236 warstw. Architektura tej sieci (rysunek 10) jest oparta na blokach MBConv (ang. *Inverted Residual Blocks*). Bloki te najpierw zmniejszają wymiarowość wejścia (warstwa zwężenia liniowego o jądrze 1x1), a następnie redukują liczbę parametrów modelu (dwie następujące po sobie warstwy: konwolucji głębokościowej o większym jądrze, np. 3x3 i konwolucji punktowej o jądrze 1x1). Sieć ta przetwarza dane bardzo szybko i jest w stanie wyodrębnić z podanego obrazu dużo szczegółów, np. strukturę wykorzystanych w nim materiałów.



Rysunek 10. Fragment sieci EfficientNet-B0 [16]

6.3. *Transfer stylu z wykorzystaniem modelu VGG19*

Aby dokonać transferu stylu za pomocą sieci VGG19 niezbędne jest odpowiednie przygotowanie danych wejściowych. Obrazy wejściowe powinny zostać zmniejszone, aby przyspieszyć wykonywanie obliczeń, oraz poddane wstępnej obróbce specyficznej dla sieci VGG19. Jako maksymalną szerokość obrazu przyjęto 1024 piksele, natomiast jego wysokość jest obliczana automatycznie (aby uniknąć deformacji).

Funkcja realizująca transfer stylu (napisana w języku Python z wykorzystaniem biblioteki Tensorflow [17]) jest przedstawiona na rysunku 11. Wykonuje ona 4 podstawowe kroki:

1. Załadowanie wstępnie wytrenowanego modelu.
2. Wygenerowanie map cech dla obrazów stylu i zawartości.
3. Przypisanie wartości początkowej (obrazu zawartości) obrazowi generowanemu.

4. Iteracyjne przeprowadzanie minimalizacji funkcji utraty całkowitej w celu aktualizacji wag modelu.

```
def neural_style_transfer(content_path, style_path, num_iterations, content_weight, style_weight):
   model = get_model()
   style_features, content_features = get_feature_representations(model, content_path, style_path)
   initial_image = load_and_process_img(content_path)
   initial_image = tf.Variable(initial_image)
   opt = tf.optimizers.Adam(learning_rate=5)
   smallest_loss, best_image = float('inf'), None
   for i in range(num_iterations):
     grads, all_loss = compute_grads(model, (style_weight, content_weight),
                                          init_image, gram_style_features, content_features)
     loss, style_score, content_score = all_loss
     opt.apply_gradients([(grads, initial_image)])
     if loss < smallest_loss:
       smallest _loss = loss
       best_image = deprocess_img(initial_image.numpy())
   return best_image, smallest_loss
```

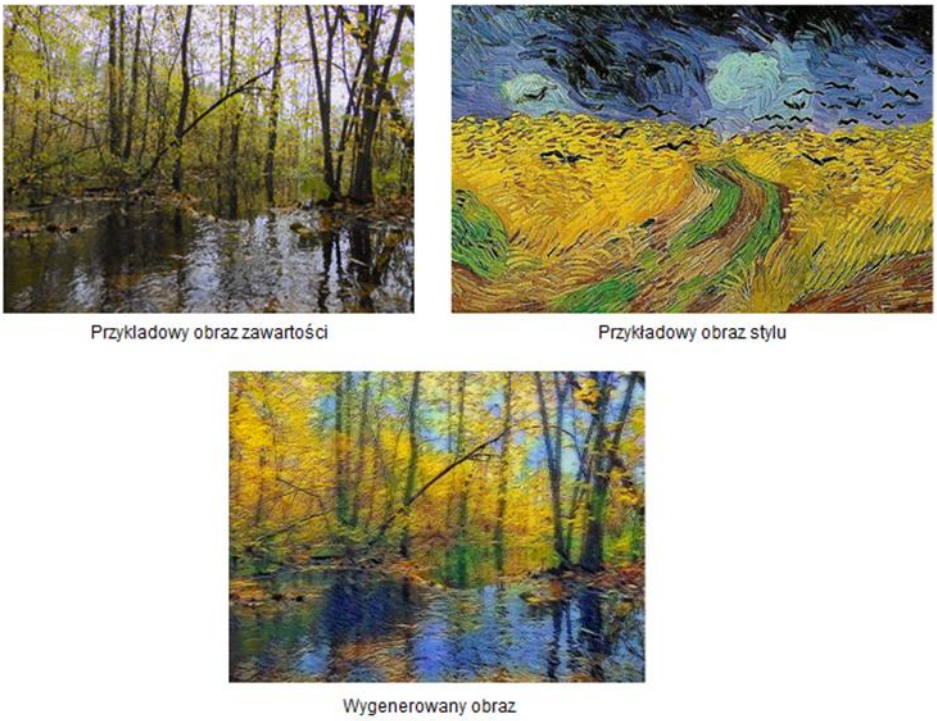
Rysunek 11. Istotne elementy funkcji przeprowadzającej transfer stylu

Kluczowym elementem w transferze stylu neuronowego jest wybranie odpowiednich warstw jako warstw stylu i warstw treści. W pierwszym teście jako warstwa treści użyta została najwyższa warstwa modelu, block5_conv4. Jako warstwy stylu wybrane zostały dwie niższe warstwy: block1_conv1 i block2_conv1. Wartości współczynników utraty stylu i zawartości ustawiono – odpowiednio – na 0,02 i 0,03.

Otrzymane wyniki zostały przedstawione na rysunku 12. Jako obraz treści wykorzystane zostało zdjęcie przedstawiające krajobraz z dużą ilością szczegółów takich jak gałęzie drzew i liście. Jako obraz stylu wybrany został obraz Vincenta Van Gogha *Pole pszenicy z krukami*, który również cechuje się dużą liczbą szczegółów, ale uproszczonych przez artystę. Przykładowo, jesteśmy w stanie rozróżnić, że na obrazie znajdują się kłosy, jednak są one uproszczone za pomocą odpowiedniej kombinacji linii w różnych odcieniach żółci i sepii.

Całkowita utrata dla wygenerowanego obrazu wynosi 1,98e + 02, z czego na utratę stylu przypada 7,01e + 01 a 1,28e + 02 na utratę treści. Zawartość wygenerowanego obrazu jest bardzo zbliżona do obrazu treści – można bezproblemowo określić, że obydwa obrazy przedstawiają ten sam krajobraz. Styl artysty został stosunkowo dobrze odwzorowany. Kolorystycznie obraz wygenerowany jest bardzo zbliżony do obrazu stylu, chociaż jego kolory są trochę bardziej intensywne. Dobrze odwzorowany został sposób przedstawiania i prowadzenia linii. Są one wyraźne, falowane z jaśniejszymi krawędziami. Jednak wygenerowany obraz zbyt dokładnie przedstawia szczegóły krajobrazu.

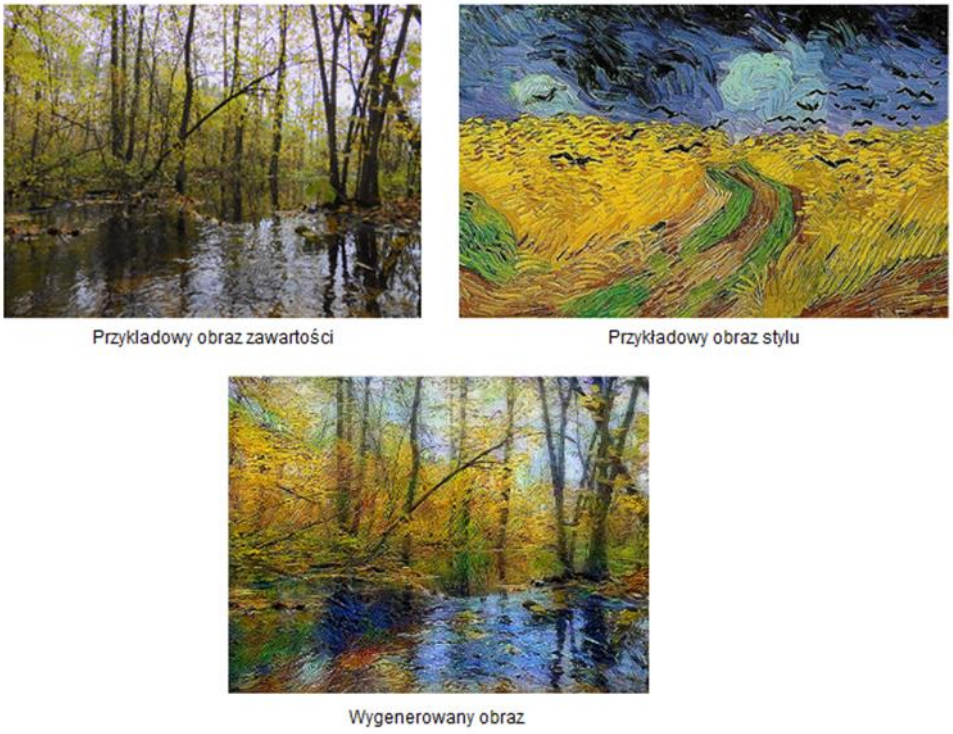
Obraz stylu jest wyraźnie uproszczony w porównaniu z rzeczywistością.



Rysunek 12. Transfer stylu za pomocą sieci VGG19; warstwa zawartości: block5_conv4, warstwy stylu: block1_conv1 i block2_conv1. Źródło obrazu stylu: [18].

Aby uzyskać lepsze odwzorowanie stylu, jako warstwę zawartości wybrano niższą niż poprzednio warstwę block5_conv2, i więcej (cztery) warstwy stylu: block1_conv1, block2_conv1, block3_conv1 i block4_conv1. Wynik transferu stylu przestawiono na rysunku 13. Całkowita utrata dla wygenerowanego obrazu wynosi 3,93e + 05, z czego 1,06e + 05 przypada na utratę stylu, a 2,87e + 05 na utratę treści. Utrata treści jest ponad 2 razy większa niż utrata stylu. Nie utrudnia to jednak rozpoznania, że obraz generowany przedstawia ten sam krajobraz, co obraz treści. W porównaniu z poprzednio wygenerowanym obrazem, kolorystyka jest bardziej przygaszona, co lepiej odpowiada kolorystyce obrazu stylu. Szczegóły zawartości są połączone w płaszczyzny kolorystyczne, bardzo dobrze odwzorowane są linie charakterystyczne dla obrazów Van Gogha: dość szerokie, lekko falowane, charakterystycznie kładzione obok siebie, częściowo na siebie zachodzące. Obraz

wygenerowany odwzorowuje również technikę, w jakiej został namalowany obraz stylu – w niektórych miejscach można zauważyć charakterystyczne dla malarstwa olejnego pociągnięcia pędzla (rysunek 14).



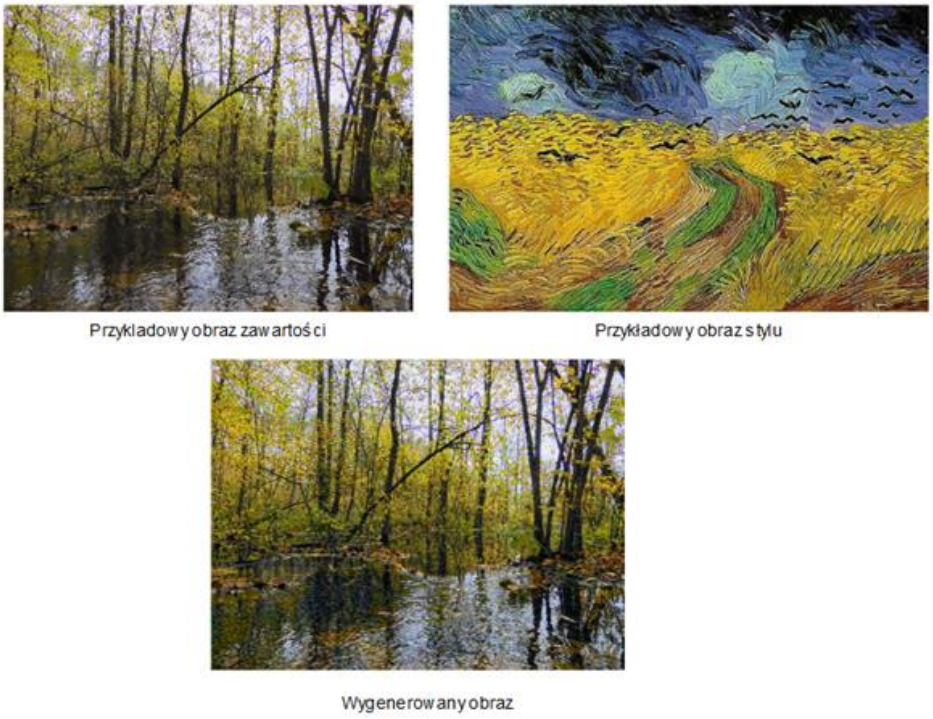
Rysunek 13. Transfer stylu za pomocą sieci VGG19; warstwa zawartości: block5_conv2, warstwy stylu: block1_conv1, block2_conv1, block3_conv1 i block4_conv1

Rysunek 14. Przykład widocznej imitacji wykorzystania farb olejnych na wygenerowanym obrazie

6.4. *Transfer stylu z wykorzystaniem modelu EfficientNet-B0*

Procedura generowania obrazu wzorowanego na styl artysty za pomocą sieci EfficientNet-B0 jest podobna do procedury wykorzystanej w poprzednim podrozdziale.

W pierwszym badaniu wybrano jako warstwę zawartości wysoką warstwę top_conv, a jako warstwy stylu cztery warstwy początkowe: block1a_se_expand, block2a_expand_conv, block2a_project_conv i block3a_expand_conv. Wartości utraty zawartości i stylu były małe i wynosiły, odpowiednio, 0,87 i 0,42. Wykorzystanie bardzo wysokiej warstwy jako warstwy zawartości spowodowało wygenerowanie obrazu o bardzo dużej szczegółowości (rysunek 15), podobnego do obrazu zawartości, a nie do obrazu stylu.



Rysunek 15. Transfer stylu za pomocą sieci EfficientNetB0; warstwa zawartości: top_conv, warstwy stylu: block1a_se_expand, block2a_expand_conv, block2a_project_conv, block3a_expand_conv

W celu poprawy działania modelu jako warstwę zawartości wybrano niższą warstwę block3a_se_expand. Warstwy stylu pozostały bez zmian. Styl otrzymanego obrazu jest teraz podobny do obrazu stylu (rysunek 16), wciąż jednak widoczna jest duża liczba szczegółów, które najprawdopodobniej zostałyby pominięte przez artystę.

Szczegóły te są najczęściej połączone we wspólne płaszczyzny kolorystyczne, co odpowiada oczekiwaniom. Dobrze odwzorowana została kolorystyka, chociaż pewne fragmenty zostały nadmiernie dopasowane do obrazu stylu (np. jaskrawo brązowe plamy na wodzie w rzeczywistości są przygaszone). Na ciemniejszych fragmentach obrazu można zobaczyć podłoże z płótna malarskiego wykorzystane w oryginalnym obrazie stylu (rysunek 17). Uzyskanie takiego efektu możliwe było dzięki zastosowaniu stosunkowo wysokiej warstwy sieci jako jednej z warstw stylu (block3a_expand_conv).

Rysunek 16. Transfer stylu za pomocą sieci EfficientNetB0; warstwa zawartości: block3a_se_expand, warstwy stylu: block1a_se_expand, block2a_expand_conv, block2a_project_conv, block3a_expand_conv

Rysunek 17. Przykład częściowego wygenerowania podłoża z obrazu stylu

7. Transfer stylu z wykorzystaniem specjalnie zaprojektowanej sieci konwolucyjnej

W celu uzyskania poprawy wyników dla transferu stylu, zaprojektowana i zaimplementowana została sieć o niewielkiej głębokości, zawierająca 11 warstw konwolucyjnych. Architektura sieci została przedstawiona w tabeli 2.

W celu znalezienia najlepszych wag początkowych dla modelu, został on przyuczony do problemu klasyfikacji dla danych ze zbioru „Animal Image Dataset" udostępnionego w serwisie Kaggle [19]. Zbiór ten został wybrany z powodu niewielkich rozmiarów (łącznie 3000 plików), które są jednak wystarczające do przyuczenia modelu do rozpoznawania odpowiednich cech danego obrazu. Zbiór składa się z trzech kategorii danych – obrazów przedstawiających psy, koty i pandy. Każda z kategorii zawiera po 1000 obrazów, które przekształcono do rozmiaru 224x224 pikseli. Do transferu stylu wykorzystywane były warstwy konwolucyjne. Sieć gęsto połączona występująca w modelu służyła do ustawienia wag początkowych, które odbyło się przez trenowanie modelu dla problemu klasyfikacji. Dokładność otrzymanego klasyfikatora dla zbioru testowego wyniosła 73% – nie jest to wysoka wartość dla klasyfikacji, ale dla rozważanego problemu transferu stylu nie ma to znaczenia.

Tabela 2. Architektura zaprojektowanej konwolucyjnej sieci neuronowej

Nazwa warstwy	Typ warstwy	Rozmiar wyjściowy	Rozmiar filtra	Liczba filtrów
Input	Input	224x224x3	-	-
conv_1	Conv2D	218x218x54	7x7	56
conv_2	Conv2D	214x214x54	5x5	56
max_pool_1	MaxPooling2D	107x107x54	-	-
conv_3	Conv2D	103x103x108	5x5	112
conv_4	Conv2D	99x99x108	5x5	112
max_pool_2	MaxPooling2D	49x49x108	-	-
conv_5	Conv2D	47x47x216	3x3	224
conv_6	Conv2D	45x45x216	3x3	224
conv_7	Conv2D	43x43x216	3x3	224
conv_8	Conv2D	41x41x216	3x3	224
max_pool_3	MaxPooling2D	20x20x216	-	-
conv_9	Conv2D	18x18x432	3x3	448
conv_10	Conv2D	16x16x432	3x3	448
conv_11	Conv2D	14x14x432	3x3	448
max_pool_4	MaxPooling2D	7x7x432	-	-
Flatten	Flatten	21168	-	-
dense_1	Dense	2697	-	-
dropout_1	Dropout	2697	-	-
dense_2	Dense	1348	-	-
dense_3	Dense	3	-	-

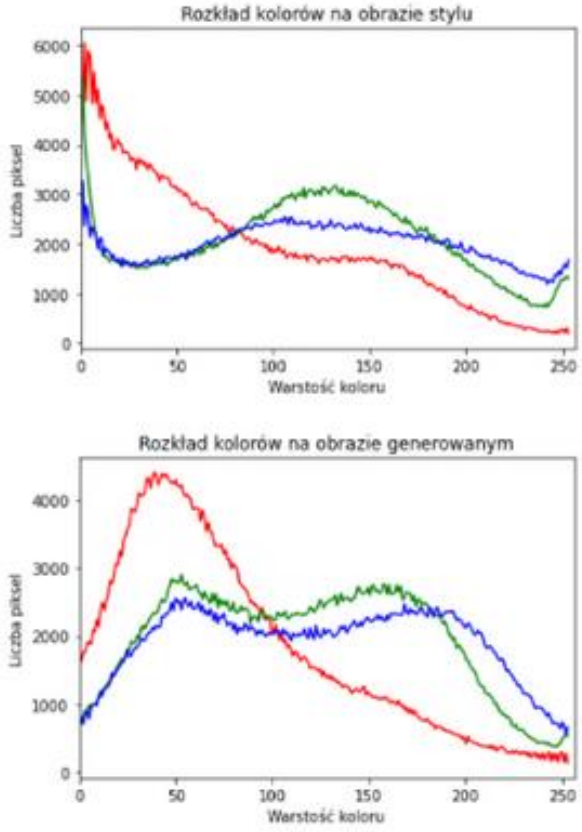
Rozmiary jądra konwolucji dla każdej warstwy zostały wybrane w taki sposób, aby ekstrahowały cechy przydane przy transferze stylu. Pierwsza warstwa konwolucyjna wykorzystuje stosunkowo duży filtr o rozmiarach 7x7 pikseli. Pozwala to na zmniejszenie szczegółowości w trakcie ekstrakcji cech z obrazu. W większości technik malarstwa lub rysunku duża część szczegółów świata rzeczywistego jest pomijana. Zdjęcia, zwłaszcza w dużej rozdzielczości, są bardzo dokładne. Artysta nie byłby w stanie odzwierciedlić każdego szczegółu, dlatego stosowane są skróty, takie jak

płaszczyzny barwne lub określone użycie światłocienia. Kolejne 3 warstwy konwolucyjne również korzystają z dużego filtra o wymiarach 5x5, co pozwala na dalsze wyodrębnienie ogólniejszych cech obrazu. Pozostałe warstwy korzystają z małego filtra 3x3, ponieważ nie można utracić wszystkich cech szczegółowych w trakcie przetwarzania obrazu zawartości. Liczba wykorzystanych w każdej warstwie konwolucyjnej filtrów dobrana została za pomocą funkcji gp_minimize() z biblioteki scikit-optimize [20].

Początkowo jako warstwę zawartości wybrano warstwę conv_9. Jako warstwy stylu wybrane zostały trzy początkowe warstwy: conv_1, conv_3 i conv_4. Utrata stylu w każdej warstwie brana była pod uwagę z jednakową wagą. Otrzymana wartość utraty całkowitej jest bardzo niska (wynosi 3.9e-01) i rozkłada się po równo między utratę zawartości i utratę stylu. Dzięki zastosowaniu większych filtrów niż w modelu VGG19 i EfficientNet-B0 udało się bardzo dobrze uzyskać efekt uogólnienia szczegółów obrazu (rysunek 18).

Rysunek 18. Transfer stylu za pomocą dedykowanej sieci; warstwa zawartości: conv_9, warstwy stylu: conv_1, conv_3 i conv_4

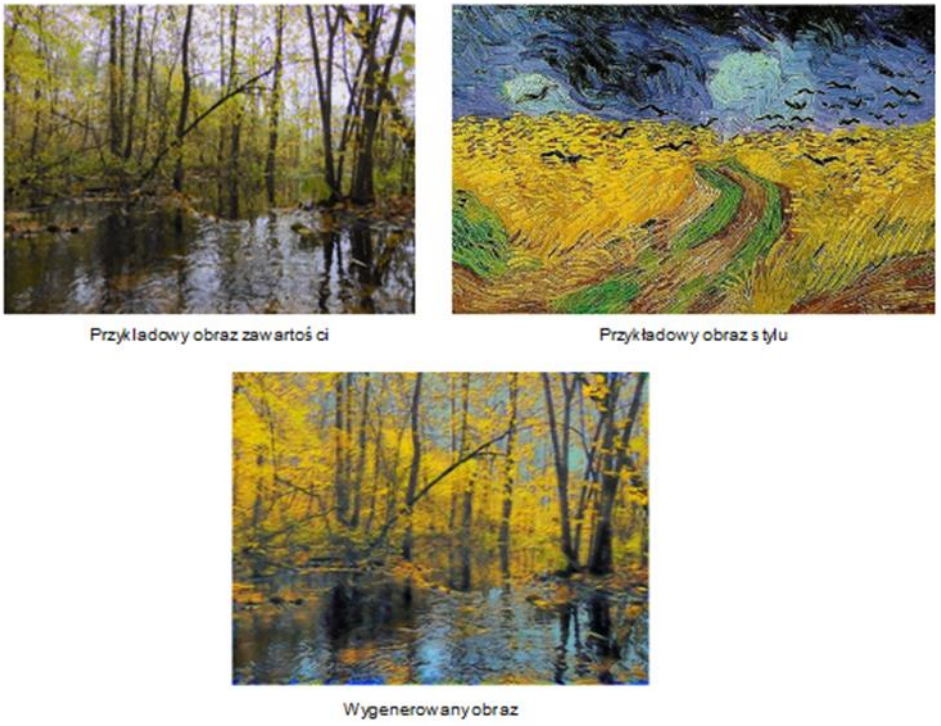
Uogólnienia te widoczne są zwłaszcza w bardziej odległych fragmentach obrazu. Tło i skupienia liści zbudowane są z płaszczyzn, które zwierają pewnego rodzaju sugestie istnienia większej liczby szczegółów w rzeczywistości (w postaci uproszczonych linii). Kolorystyka obrazu stylu została w bardzo dobry sposób przeniesiona na wygenerowany obraz, zwłaszcza odcienie żółtego oraz niebieskiego. Porównując rozkłady kolorów dla obrazu wygenerowanego i obrazu stylu, widzimy ich duże podobieństwo (rysunek 19).



Rysunek 19. Rozkład barw na obrazie stylu i obrazie generowanym

Bardzo dobrze widoczne są charakterystyczne dla obrazów Van Gogha falowane linie, o wyraźnym pociągnięciu farby olejnej – z jasnymi krawędziami (rysunek 20).

Rysunek 20. Długie, falowane linie na wygenerowanym obrazie



Rysunek 21. Transfer stylu za pomocą dedykowanej sieci; warstwa zawartości: conv_8, warstwy stylu: conv_1, conv_2, conv_3, conv_4

Działanie modelu, mimo że generuje on dobre wyniki, można jeszcze poprawić. Kolory w niektórych miejscach obrazu generowanego są zbyt żywe, szczegółowość obrazu mogłaby zostać dodatkowo zmniejszona, zwłaszcza reprezentacja wody. W celu

poprawienia działania transferu stylu jako warstwę zawartości wybrano warstwę conv_8, a jako warstwy stylu: conv_1, conv_2, conv_3, conv_4 z wagami wynoszącymi odpowiednio, 0,1; 0,3; 0,5; 0,1. Wygenerowany obraz (rysunek 21) zawiera trochę więcej uogólnień w porównaniu do poprzedniego obrazu (rysunek 18). Dodatkowo pojawiły się przygaszenia barw, które nadają obrazowi bardziej ponury charakter, podobny do obrazu stylu. Żółcienie są prawie idealnie odzwierciedlone, ciemne brązy występujące na obrazie zawartości zamienione zostały na bardzo przyciemnione odcienie sienny, które widoczne są w obrazie stylu (droga przez zboże).

Przeprowadzone zostało również jeszcze jedno badanie, w którym jako obraz stylu został wykorzystany obraz namalowany inną techniką niż olejna, a mianowicie za pomocą suchych pasteli (rysunek 22).

Rysunek 22. Transfer stylu za pomocą dedykowanej sieci; technika obraz stylu: suchy pastel. Źródło obrazu zawartości: [21].

W efekcie otrzymano obraz, na którym widoczne są cechy szczególne dla malarstwa pastelami, takie jak wyraźna struktura pigmentu na kartce oraz płynność przejść pomiędzy barwami. W malarskie olejnym prezentowanym w poprzednich przykładach wyraźnie widoczne były krawędzie pomiędzy poszczególnymi barwami, które bardzo rzadko występują w przypadku użycia pasteli.

Ponieważ jako obraz zawartości wybrana została „Mona Lisa" Leonarda da Vinci, problem zbyt dużej liczby szczegółów nie wystąpił. Zarówno obraz zawartości jak i stylu są zbliżone szczegółowością. Kolorystyka, zwłaszcza w obrębie twarzy, została bardzo dobrze przeniesiona, wygenerowany obraz zachował sposób przedstawiania cieni z obrazu zawartości, ale widoczne są na nim bardzo silne rozjaśnienia charakterystyczne dla obrazu stylu, które dają wrażenie lepszej różnicy przestrzennej, zwłaszcza w obrębie twarzy.

8. Podsumowanie

Transfer stylu artystycznego jest zadaniem bardzo subiektywnym. Ludzie odbierają obrazy w różny sposób. Dla niektórych większe znaczenie w rozpoznawaniu stylu ma zachowanie kolorystyki danego obrazu, inni patrzą na styl w szerszym znaczeniu stylu występującego we wszystkich obrazach danego artysty lub danej epoki. Z powodu takiej różnorodności pojmowania pojęcia stylu, jego transfer jest bardzo trudnym zadaniem.

W niniejszym artykule transfer stylu został przeprowadzony z wykorzystaniem wstępnie wytrenowanych konwolucyjnych sieci neuronowych: powszechnie stosowanych VGG19 i EffcientNet-B0 oraz sieci specjalnie zaprojektowanej do transferu stylu o stosunkowo dużych rozmiarach filtrów w najniższych warstwach modelu. Pokazane zostało, jak przez odpowiedni dobór warstw dla obrazów stylu i zawartości, można osiągnąć oczekiwaną stylizację pod względem kolorystyki, techniki wykonania obrazu i poziomu uogólnienia szczegółów rzeczywistości. Zastosowanie sieci o architekturze dedykowanej do transferu stylu przyczyniło się do dalszej poprawy odwzorowania stylu.

Chociaż otrzymane efekty są całkiem zadowalające konieczne są dalsze badania w celu opracowania ogólnych technik i metod pozwalających osiągać dokładniejsze odzwierciedlenia stylu z zadanego obrazu. Potencjał do dalszego rozwoju widoczny jest w możliwości zastosowania modeli generatywnych sztucznych sieci neuronowych.

Literatura

[1] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* No 65(6), ss. 386-408, 1958.

[2] P. Werbos, *Beyond regression: new tools for prediction and analysis in the behavioral sciences*, Ph.D. thesis. Cambridge: Harvard University, 1974.

[3] R. Dechter, *Learning while searching in constraint-satisfaction problems*. Los Angeles: University of California, Computer Science Department, Cognitive Systems Laboratory, 1986.

[4] D.E. Rumelhart, G.E. Hinton, R.J. Williams. *Learning Representations by Back-Propagating Errors* [w:] (Eds.) D.E. Rumelhart, J.L. McClelland. *Parallel distributed processing: explorations in the microstructure of cognition*, Vol. 1, ss. 318-362, Cambridge: MIT Press, 1986.

[5] G.E. Hinton, R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks, *Science* No 313 (5786), ss. 504-507, 2006.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE* No 86/11, ss. 2278-2324, 1998.

[7] L.A. Gatys, A.S. Ecker, M. Bethge. *A neural algorithm of artistic style*. [Online]. https://doi.org/10.48550/arXiv.1508.06576, 2015.

[8] Q. Cai, M. Ma, C. Wang, H. Li. Image neural style transfer: A review. *Computers and Electrical Engineering* Vol. 108, s. 108723, 2023.

[9] W. McCulloch, W. Pitts. A Logical Calculus of the ideas Imminent in Nervous Activity. *Bulletin of Mathematical Biophysics* Vol. 52, ss. 99-115, 1990.

[10] *Neuron McCulloha-Pittsa*. [Online]. https://pl.wikipedia.org/wiki/Neuron_McCullocha-Pittsa.

[11] *What are convolutional neural networks*. [Online]. https://www.ibm.com/topics/convolutional-neural-networks.

[12] A. Ajit, K. Acharya, A. Samanta. A review of convolutional neural networks, *Int. Conference on Emerging trends in Information Technology and Engineering*, 2020.

[13] C. Albon, R. Górczynski. *Uczenie maszynowe w Pythonie: receptury*. Gliwice: Helion, 2019.

[14] VGG Very Deep Convolutional Networks (VGGNet) – What you need to know. [Online]. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.
[15] M. Tan, Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. [Online]. https://arxiv.org/abs/1905.11946.
[16] *EfficientNet: Improving Accuracy and Efficiency through AutoMl and Model Scaling*. [Online]. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.
[17] *Create production-grade machine learning models with TensorFlow*. [Online]. https://www.tensorflow.org/.
[18] *Pole pszenicy z krukami*. [Online]. https://pl.wikipedia.org/wiki/Pole_pszenicy_z_krukami.
[19] *Animal Image Dataset (DOG, CAT and PANDA)*. [Online]. https://www.kaggle.com/ashishsaxena2209/animal-image-datasetdog-cat-and-panda.
[20] *scikit-optimize*. [Online]. https://scikit-optimize.github.io/stable/.
[21] *Mona Lisa*. [Online]. https://pl.wikipedia.org/wiki/Mona_Lisa.

The use of pre-trained convolutional neural networks for style transfer

Abstract

The article concerns the use of pre-trained convolutional neural networks for the problem of style transfer. It examines how the selection of convolutional layers for representing the style and content images influences the quality of style replication, including the color palette of the generated image, the visible technique of its execution, and the level of generalization of details in the presented reality. The article also proposes an architecture for the convolutional neural network dedicated to the discussed problem.

Keywords: convolutional neural networks, neural style transfer, pre-trained neural networks