

<b>UNIwersYTET ZIELONOGÓRSKI</b>	<b>WYKONALI:</b>	<b>GRUPA</b>		<b>OCENA:</b>
<b><i>Rozpoznawanie obrazów</i></b>				
<b>ĆWICZENIE NUMER: 7</b>	<b>TEMAT:</b> Kompresja obrazów	<b>DATA WYKONANIA:</b>	<b>DATA ODDANIA:</b>	<b>PODPIS:</b>

## 1. Cel ćwiczenia

Celem ćwiczenia jest zaznajomienie się z zagadnieniami kompresji stratnej obrazów, czyli takiego upakowania danych zawierających informacje aby przy jak najmniejszej stracie jakościowej uzyskać możliwie najmniejszy rozmiar objętościowy danych.

Ćwiczenie podzielone było na dwie części. Pierwsza część dotyczyła kompresji obrazów przy pomocy transformaty Fouriera, natomiast druga do redukcji rozmiaru używała prostej sieci neuronowej.

## 2. Kompresja przy pomocy transformaty Fouriera

Ta część ćwiczenia polegała na zaprojektowaniu automatu zdolnego kompresować obraz z wykorzystaniem transformaty Fouriera. Jak wiemy z jednych z poprzednich zajęć przy pomocy transformaty Fouriera możemy wydobyć interesujący nas zakres częstotliwości danych. Wiemy również że wysokie częstotliwości to części obrazu które niosą ze sobą istotna dla nas wiadomość, natomiast niskie częstotliwości są praktycznie mało znaczące. Wykorzystując tą cechę, możemy w łatwy sposób doprowadzić do zmniejszenia rozmiaru kompresowanego obiektu. Dokonać tego możemy, przez proste obcięcie wartości które nie niosą ze sobą istotnych informacji, dla tego celu ustalimy sobie próg poniżej którego wszystkie wartości zostaną utracone. Mówiąc „zostaną utracone” mamy na myśli zastąpienie tych wartości zerami, a jak wiadomo długie ciągi zer można w łatwy sposób upakować, dzięki czemu uzyskamy mniejszy rozmiar danych. Doszliśmy do wniosku, że najlepszą metodą na upakowanie wartości zerowych będzie skompresowanie transformaty Fouriera przez i po zastąpieniu wartości nieistotnych zerami przy pomocy kompresji JPEG. Na podstawie tak uzyskanych plików możemy oszacować o ile udało nam się zmniejszyć pamięć potrzebną do zapamiętania obrazu. Nie możemy jednak odtwarzać obrazu z skompresowanych obrazów transformaty Fouriera przy pomocy algorytmu JPEG, gdyż kompresja ta jest stratna i wpływała by na przekłamanie wyników. Taka kompresja ma nam tylko powiedzieć jak bardzo można upakować dane. Dlatego w programie przechowujemy oryginalną transformatę oraz tą z usuniętymi wartościami małoistotnymi, i to z nich odtwarzamy obraz. Jeśli chcielibyśmy zapisywać transformatę z usuniętymi wartościami małoistotnymi tak aby zajmowała mniej miejsca, musielibyśmy użyć kompresji bezstratnej. Jednym z rozwiązań mogło by być zapamiętywanie wartości istotnych wraz z współrzędnymi ich położenia na obrazie. Do naszego zadania w pełni wystarczy jednak metoda którą zastosowaliśmy gdyż mamy tylko oszacować stopień kompresji.

Algorytm zastosowany przez nas w programie Matlab wygląda następująco:

- Przy pomocy funkcji `fft2` wyznaczamy transformatę Fouriera dla obrazu oryginalnego.

```
G=fft2(obraz);
```

- Zapisujemy transformatę `G` do pliku: *przed.jpg*
- Ustalamy współczynnik poniżej którego wszystkie wartości będą uznawane za mało istotne i zostaną zastąpione zerami.

```
if log(abs(G(i,j)))<=wsp  
    G(i,j)=0;  
End
```

Wartość współczynnika zawiera się w przedziale od 0 do ok. 20. Górna wartość jest zależna od oryginalnego obrazka.

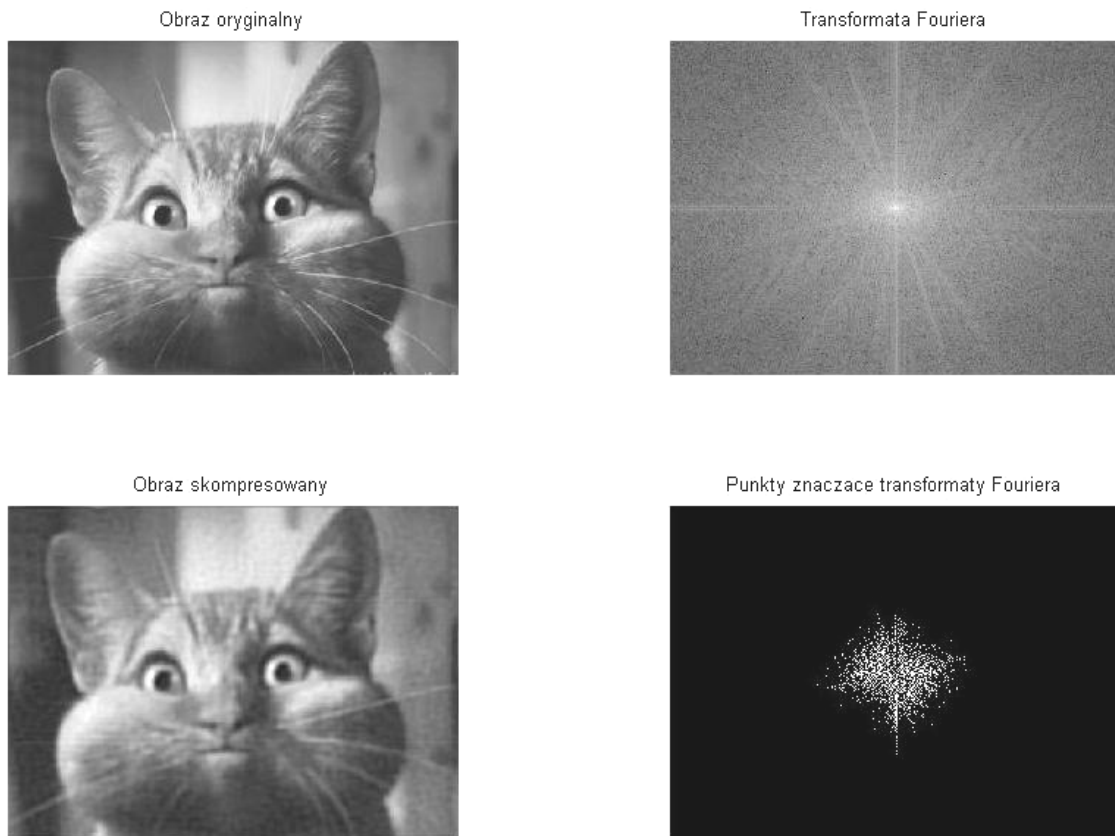
- Po zastąpieniu zerami wartości nieistotnych, zapisujemy transformatę do pliku: *po.jpg*
- Współczynnik kompresji wyznaczamy na podstawie tego ile razy większy jest plik *przed.jpg* od pliku *po.jpg*.

```
przed = dir('przed.jpg');  
po = dir('po.jpg');  
st_komp = przed.bytes / po.bytes;
```

Błąd kompresji wyznaczamy jako błąd średniokwadratowy obrazu oryginalnego i obrazu odtworzonego z transformaty Fouriera po zastąpieniu wartości nieistotnych zerami.

```
przed=imread('a_przed.jpg');  
po=imread('a_po.jpg');  
e=(przed-po);  
e=double(e);  
blad=mse(e);
```

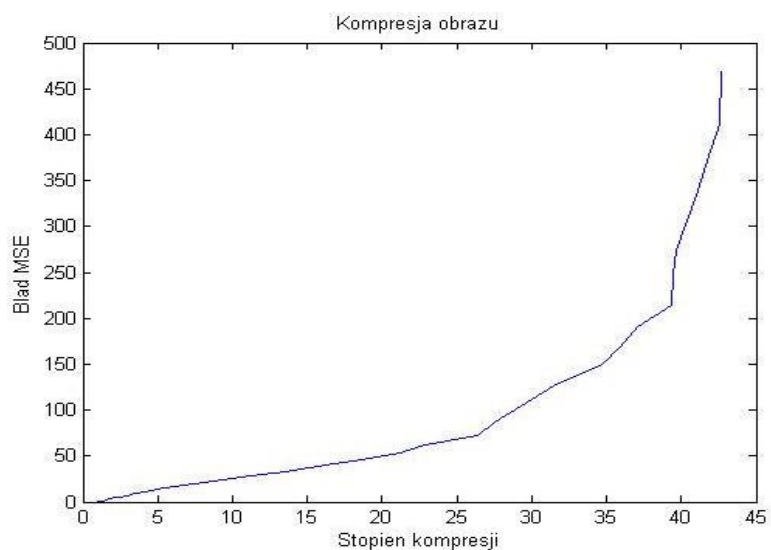
Przykład działania algorytmu kompresji zamieszczamy poniżej.



Rysunek 1. Przykład kompresji.

W tym przypadku współczynnik odcinania wartości w transformacie został ustalony na poziom 3,71. Rozmiar pliku oryginalnej transformaty wynosi 76036 B a po odcięciu wartości nieistotnych 7202 B.

Następnym krokiem było przeprowadzenie kompresji dla różnych wartości współczynnika odcięcia wartości nieistotnych. Aby zautomatyzować nasz program zadanie wykonaliśmy w pętli a współczynnik przyjmował kolejno wartości od 0 do 19.8.



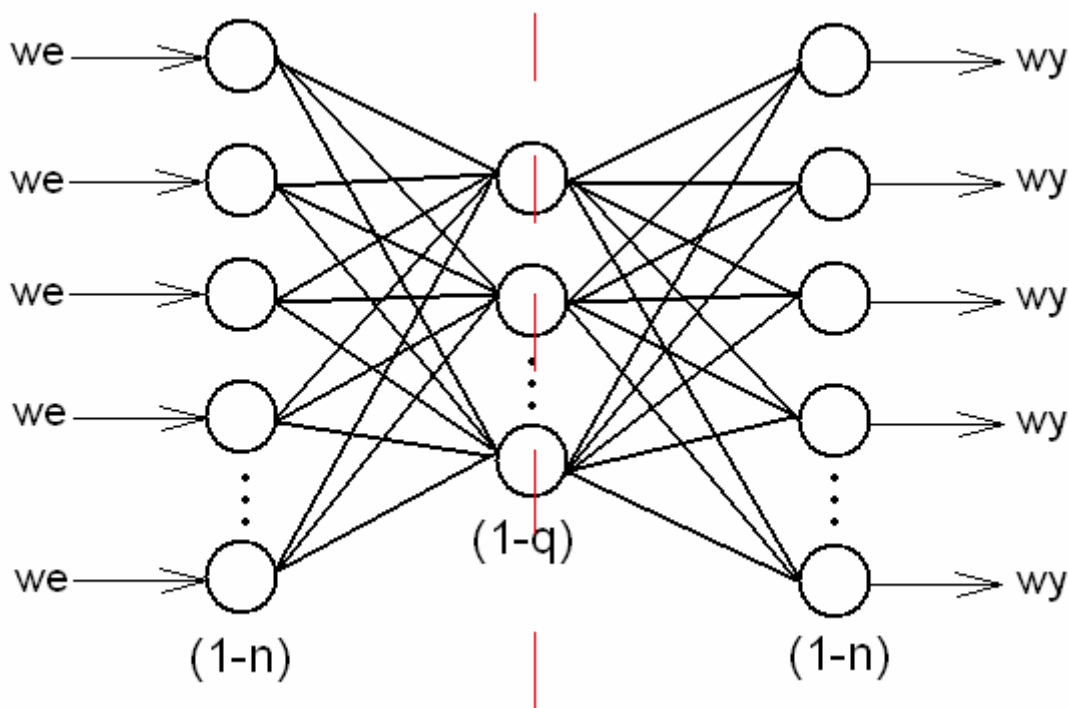
Rysunek 2. Wykres kompresji obrazu.

Z przedstawionego wykresu widać że wraz ze wzrostem stopnia kompresji rośnie także błąd, jednak do pewnego momentu błąd ten utrzymuje się na niewielkim, akceptowalnym pułapie. Podczas gdy kompresja osiąga duże wartości błąd ten zaczyna rosnać lawinowo. Na rysunku objawia się to mocnym pogorszeniem jakości grafiki, aż do tego stopnia że obraz skompresowany zaczyna przypominać kilka rozmazanych plam.

Jak widać kompresja przy pomocy transformaty Fouriera, należy do grona kompresji stratnych, która w pewnym przedziale stopnia kompresji daje całkiem zadowalające wyniki, ale tylko wtedy gdy nie zależy nam na ostrości obrazka, bo tą na pewno utracimy decydując się na taką metodę.

### 3. Kompresja przy pomocy sieci neuronowej

Ta część ćwiczenia polegała na zbudowaniu kompresora obrazów graficznych z wykorzystaniem sztucznej sieci neuronowej. Możemy tego dokonać stosując specyficzną strukturę budowy sieci, a mianowicie kiedy zastosujemy budowę podobną do tej przedstawionej na poniższy rysunku.



Rysunek 3. Budowa kompresora graficznego przy użyciu sieci neuronowej.

Jak widzimy na rys.3 sieć neuronowa która ma kompresować obrazy, musi posiadać tyle samo wyjść co wejść ( $n$ ), natomiast warstwa środkowa posiada mniejszą liczbę neuronów ( $q$ ). Dzięki temu sieć możemy podzielić na dwie części (czerwona przerywana linia). Pierwsza część będzie służyła do kompresji obrazów a druga do dekompresji. Dzięki zastosowaniu mniejszej ilości neuronów w środkowej warstwie, będziemy mogli zapamiętać dane na mniejszej przestrzeni niż oryginał. Z tak zapamiętanych danych, możemy je później odtworzyć używając do tego celu drugą część sieci (po prawej stronie czerwonej linii).

Musimy jeszcze przeanalizować jedną sprawę. Jak dobrać liczbę wejść? Wejść nie może być tyle ile jest pikseli w obrazie, taka sieć uczyła by się zbyt długo a jej zastosowanie byłoby wątpliwe. Dlatego zastosujemy inne rozwiązanie. Podzielmy obraz wejściowy na małe bloki które to, będziemy po kolei podawać na wejście naszej sieci. Takie

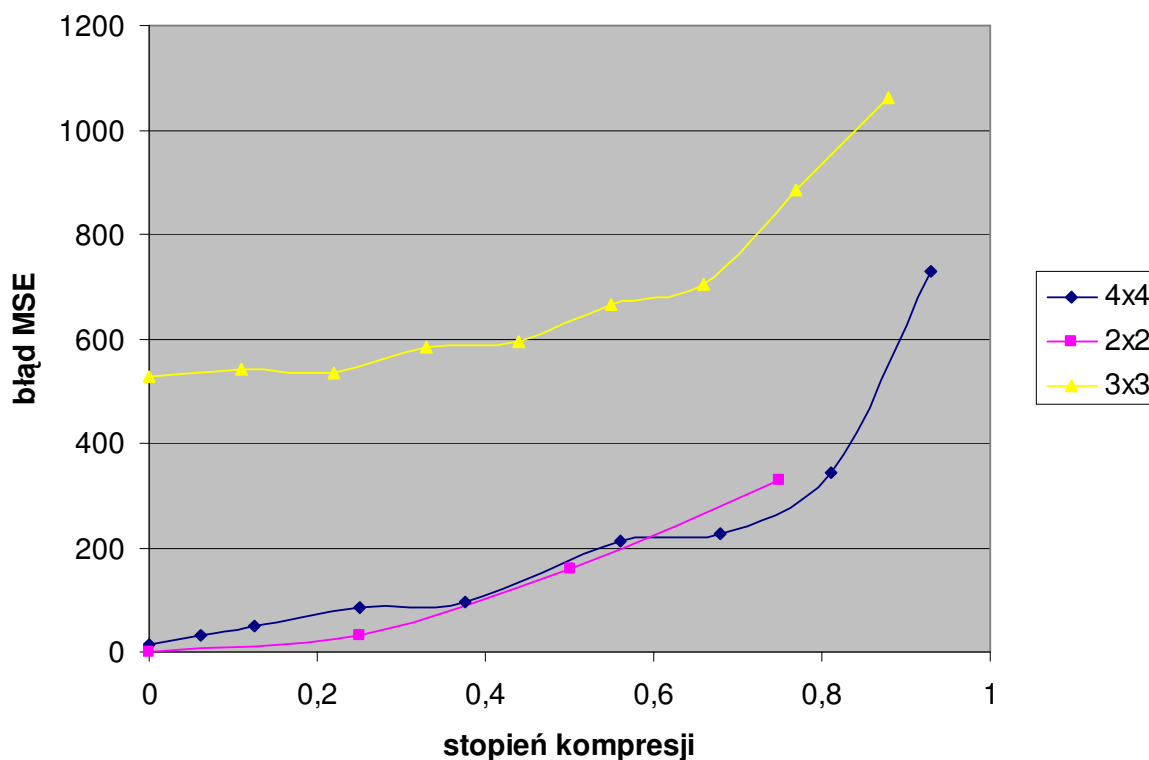
rozwiązanie zagwarantuje nam to iż sieć będzie się uczyła względnie szybko, oraz nie będzie posiadała gigantycznych rozmiarów. Jest jeszcze jedna zalet takiego rozwiązania, mianowicie nie musimy przebudowywać sieci neuronowej jeśli zmieniamy rozmiar obrazka. W przypadku kiedy sieć miała by tyle wejść co obraz pikseli, to podczas podania obrazka o innej rozdzielczości zabrakło by nam lub mieli byśmy zbyt dużą liczbę wejść sieci.

Na wejście naszego kompresora podaliśmy obrazek którego rozdzielczość wynosiła 160 x 116 pikseli.

W poniższej tabeli zamieszczamy wyniki jakie uzyskaliśmy.

2 x 2		3 x 3		4 x 4	
stopień kompresji	błąd MSE	stopień kompresji	błąd MSE	stopień kompresji	błąd MSE
0	0,034	0	528,58	0	15,65
0,25	33,55	0,11	540,84	0,062	33,21
0,5	160,62	0,22	533,43	0,125	49,32
0,75	328,85	0,33	584,4	0,25	84,51
		0,44	594,27	0,375	96,28
		0,55	663,99	0,56	211,41
		0,66	705,35	0,68	226,9
		0,77	884,11	0,81	343,1
		0,88	1060,5	0,93	728,23

Poniżej znajduje się wykres stopnia kompresji od błędu MSE.



Istotnym problemem który wpływał na czas uczenia sieci była rozdzielczość obrazka, im większa tym sieć uczyła się wolniej. Nie stwierdziliśmy jednak większych różnic w czasie uczenia przy zmianie rozmiaru sieci na większy, a niekiedy nawet sieć uczyła się szybciej kiedy była większa (posiadała więcej neuronów).

Podobnie jak w poprzednim zadaniu, ta metoda kompresji obrazu jest również stratna, co oznacza iż nie da się odzyskać oryginalnej postaci obrazka z jego wersji skompresowanej. Jak widać z naszych doświadczeń, większy rozmiar bloków, wcale nie oznacza mniejszego błędu kompresji. Zauważmy że bloki 4x4 były lepsze od 2x2 dopiero po przekroczeniu stopnia kompresji ok. 0,6. Całkowicie nie wyjaśniane są wyniki działania programu dla bloków 3x3, błąd kompresji osiąga wtedy zawrotnie duże wartości. Udało nam się jednak ustalić że wyniki te zależą od tego jaki obrazek podamy na sieć, dlatego zastosowanie metody kompresji przy użyciu jednokierunkowych sieci neuronowych wydaje nam się całkowicie nieuzasadnione. Lepszym wyborem jeśli chodzi o sieci neuronowe było by zastosowanie sieci Kohonena. Te sieci doskonale nadają się do kompresji obrazów.

Porównując obie metody kompresji, tj. z użyciem transformaty Fouriera, oraz jednokierunkowej sieci neuronowej, zdecydowalibyśmy się na wybranie metody pierwszej z uwagi na mniejsze błędy kompresowanych obrazów.