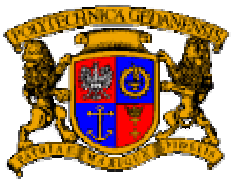


Politechnika Gdańska

WYDZIAŁ ELEKTRONIKI TELEKOMUNIKACJI I INFORMATYKI

Katedra Systemów Informacyjnych
Imię i nazwisko dyplomanta: Marek Puchalski
Nr albumu: 48686/WEE
Rodzaj studiów: Dzienne

Praca magisterska

Temat pracy:

Badanie jakości metody kompresji obrazów nieruchomych za pomocą sieci neuronowej ze wstępnym przybliżeniem transformatą Karhunena-Loeve.

Kierujący pracą:

Dr hab. inż. Roman Rykaczewski

Zakres pracy:

Stworzono program narzędziowy do analizy metody kompresji obrazów nieruchomych za pomocą sieci neuronowej oraz za pomocą transformaty KLT.

Przeprowadzono badania, w jaki sposób metoda KLT przybliża wielkość unormowanego błędu średniokwadratowego powstałego w wyniku działania sztucznej sieci neuronowej oraz badanie jakości metody kompresji obrazu za pomocą sztucznej sieci neuronowej.

Kierujący pracą | Kierownik Katedry

_____ _____

Spis treści

Wstęp

„Choćbyś głupca utarł wraz z krupami w moździerzu,
to i tak nie opuści go jego głupota”

Przyp.27,22

Tak Król Salomon piętnuje głupotę, a promuje szukanie mądrości, która daje człowiekowi władzę nad wszystkimi formami życia na ziemi. Czymże jest ta mądrość? Jedna z definicji opisuje ją jako „zasób wiedzy wykorzystywany przy pomocy inteligencji”. W związku z tym, że człowiek ma ograniczone możliwości przyswajania i przetwarzania informacji zaczął on budować maszyny zwane komputerami, które go w tej działalności wspomagają, a niekiedy nawet wyręczają. Zaczął on również budować sieci komputerowe w celu przesyłania tejże informacji w odległe miejsca na kuli ziemskiej.

Jednym z typów przesyłanych informacji jest obraz. Jest to jednocześnie jeden z najbardziej wymagających pod względem wielkości typów danych. W przypadku, gdy przesłanie jednego obrazu nie stanowi już dzisiaj większego problemu, to przesłanie zestawu wielu obrazów, dokładnych map, czy obrazów ruchomych powoduje znaczne obciążenie wykorzystywanych łączy. W związku z tym poszukuje się metod kompresji, które pozwoliłyby znacznie zredukować objętość przesyłanych danych bez utraty istotnej informacji. Jedna z takich metod wykorzystuje zasadę działania sztucznej sieci neuronowej, symulującej działanie ludzkiego mózgu, który na podstawie niewielu widocznych szczegółów jest w stanie odtworzyć pierwotny obraz.

Celem niniejszej pracy jest badanie jakości kompresji obrazów nieruchomych za pomocą sztucznej sieci neuronowej ze szczególnym uwzględnieniem stopnia kompresji i jakości odtworzonego obrazu. Dodatkowo przeprowadzone zostaną:

- analiza właściwości i zachowania się różnych konfiguracji sieci,
- wstępne przybliżenie parametrów sieci za pomocą transformaty KLT,
- przetwarzanie i obliczanie statystyk obrazu.

Praca składa się z rozdziałów obejmujących teoretyczny opis zagadnień, dokładne omówienie metody kompresji oraz analizę wyników z przeprowadzonych eksperymentów.

W niniejszej pracy postaram się dowieść prawdziwości założenia, że unormowany błąd średniokwadratowy obliczony na podstawie różnic pomiędzy oryginalnym obrazem, a wynikiem uzyskanym po jego kompresji i odtworzeniu najlepszą (pod względem minimalizacji tegoż błędu) z metod transformacyjnych, a mianowicie transformatą Karhunena – Loeve jest przybliżeniem analogicznej wartości po przetworzeniu przez wielowarstwową sieć neuronową.

W celu potwierdzenia powyższej tezy przeprowadzone zostaną badania dla obu metod kompresji. Pod uwagę zostaną wzięte zarówno różne rozmiary podobrazów, jak również różny stopień kompresji.

Narzędziem, przy którego pomocy będą generowane wyniki jest specjalnie to tego celu napisany program komputerowy o nazwie „Kompresja Neuronowa". Jest on zamieszczony wraz z kodem źródłowym na dołączonej do pracy płycie CD. Stworzony on został w środowisku programistycznym „Delphi 6" w wersji testowej, zamieszczonej w jednym z czasopism komputerowych.

Ze względu na to, że temat sieci neuronowych jest bardzo rozległy, w pracy ograniczono się do prostej sieci dwu i trzywarstwowej uczonej metodą propagacji wstecznej błędu. Pominięte natomiast zostały metody przyśpieszania uczenia, optymalnego doboru wag początkowych, czy też struktury samej sieci. Dodatkowo ograniczono się do analizy obrazów o 256 odcieniach szarości, ponieważ obrazy kolorowe to nic innego jak trzy składowe RGB o 256 możliwych wartościach, dla których można zastosować opisywaną metodę, lecz jej implementacja zaciemniłaby istotę opisywanych zagadnień.

W poszczególnych rozdziałach przedstawione zostaną kolejno:

- **czym jest kompresja** – ogólny opis istoty kompresji oraz przegląd wybranych metod stratnych i bezstratnych;
- **obraz** – jego reprezentacja i formaty w postaci cyfrowej.
- **transformata KLT** – zasada działania i jej implementacja w programie;
- **sieć neuronowa** – rys historyczny, podobieństwo do sieci neuronowych w ludzkim mózgu, zasada działania i jej realizacja w programie;
- **badania i podsumowanie** – eksperymenty i generowanie wyników oraz ich analiza i częściowe potwierdzenie postawionej tezy;

- **opis narzędzia** – dokumentacja programu „Kompresja Neuronowa”;
- **bibliografia.**

W niniejszym opracowaniu wykorzystano dostępne na naszym rynku książki, opracowania dr hab. inż. Romana Rykaczewskiego, pracę magisterską, którą złożyła pani mgr inż. Hanna Pempera na Wydziale ETI Politechniki Gdańskiej, jak również opracowania z internetu oraz z czasopism o tematyce komputerowej.

Nigdzie jednak nie znalazłem gotowego kodu źródłowego w języku Pascal/Delphi, który można by wykorzystać do badań nad zagadnieniem wykorzystania sztucznych sieci neuronowych do kompresji obrazów. Dlatego też mam nadzieję, że praca ta i skonstruowany przeze mnie program zajmujący ok. 5600 linii kodu udostępniony na załączonym dysku CD przyczynią się do szybszego rozwoju i pogłębienia wiedzy w zakresie budowy sztucznej inteligencji, która znakomicie ułatwi życie codzienne każdego z nas.

Rozdział 1
Kompresja danych

Opis istoty kompresji danych, metod i algorytmów oraz ich wykorzystania w przetwarzaniu i transmisji danych.

1.1 Istota kompresji danych

W dzisiejszym świecie najważniejszą wartością, którą operuje społeczeństwo jest informacja. Jej przesyłanie, składowanie i przeszukiwanie należy do najczęściej wykonywanych czynności w codziennym życiu. Dynamiczny rozwój sieci telekomunikacyjnych, a w tym telefonii komórkowej oraz sieci komputerowych umożliwia proste i szybki przesyłanie danych w najodleglejsze zakątki poznanego świata. Wielu z nas pamięta pierwsze modemy powstałe w erze telefonii analogowej, gdy na instalację telefonu trzeba było czekać latami. Były one w stanie przesyłać dane z szybkością 300 bitów/s i były szczytem techniki, a jednocześnie w zupełności wystarczyły do przesyłania danych tekstowych. Piękne pionierskie lata, które dawno minęły i w drodze ewolucji doprowadziły do czasów dzisiejszych, gdzie na porządku dziennym jest telefonia komórkowa dostępna „od zaraz" praktycznie dla każdego, internet czy też lokalne sieci komputerowe o przepustowości 100 Mbit/s. Wraz z rozwojem techniki wzrastały również stawiane jej wymagania i dla każdego użytkownika współczesnego komputera chlebem powszednim jest „okienkowy" system operacyjny, kolorowe obrazy, filmy czy wysokiej jakości dźwięk przetwarzany przez tenże PC. Bardzo często zdarza się tak, że te ogromne w swych rozmiarach ilości danych pragniemy komuś przesłać. Wtedy zaczynamy się zastanawiać, czy dane te zmieszczą się na dyskietce, płycie CD, czy pojemność skrzynki pocztowej wystarczy, czy koszt wysłania informacji przez modem nie zrujnuje domowego budżetu... W tym momencie z pomocą przychodzi działanie polegające na zmniejszaniu rozmiarów wysyłanych plików nazywane „kompresją danych".

Ponad pół wieku temu po raz pierwszy pojęcie „kompresji danych" pojawiło się w pracach Claude'a Shannona będące rezultatem badań prowadzonych w laboratoriach Bella w zakresie komunikacji, informacji oraz przechowywania danych. Zadaniem kompresji jest redukcja nadmiarowości danych, która najbardziej widoczna jest w przypadku „naturalnych danych", jakimi są obraz i dźwięk.

Dla tego typu danych najczęściej stosuje się metody kompresji „stratnej", tzn. takiej, w której dopuszcza się pewną utratę lub zniekształcenie danych. W przypadku dźwięku zapis w popularnym formacie MP3 powoduje zmniejszenie rozmiaru pliku około 10 krotne przy całkowicie akceptowalnej jakości sygnału i poziomie zniekształceń. Podobnie dzieje się w przypadku kompresji obrazów, dla których zastosowanie np. formatu JPEG powoduje redukcję wielkości nawet kilkusetkrotną! Znaczenie tego faktu najłatwiej zrozumieć na poniższym przykładzie:

Kartka formatu A4 (9"x12") zeskanowana z rozdzielczością 300dpi w pełnym kolorze tzn. z rozdzielczością 24 bit/piksel wymaga do jej zapisu 9" x 12" x 300 x 300 x 24bit = 233 280 000 bitów ≈ 29MB. Jej przesłanie w postaci obrazu w formacie BMP lub formacie JPG (wykorzystującym algorytm kompresji stratnej) za pomocą modemu pracującego z szybkością 56 kbit/s zajmuje czas podany w Tabeli 1.

Format zapisu	Rozmiar	Czas przesłania	Koszt przesłania
BMP	29MB	1h 9min	7zł
JPG	100kB	14s	2gr

Tabela1. Porównanie kosztów transmisji dla formatów BMP oraz JPG.

Algorytmy kompresji bezstratnej również potrafią dać imponujące wyniki. Tak np. użycie programu ARJ wykorzystującego połączone metody, słownikową i probabilistyczną, do pliku bazy danych DBF powoduje redukcję rozmiaru rzędu 100 razy. Jednak zwykle tego typu programy dają zysk ok. 30 - 50% i dlatego też do zapisu danych naturalnych stosuje się kompresję stratną.

Wielkością pozwalającą zmierzyć zawartość informacji w obrazie jest wprowadzona przez Claude'a Shannona „Entropia" i oznacza ona średnią ilość informacji niesionej przez źródło.

$$\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$$

gdzie:

p_i – prawdop. wystąpienia próbki o poziomie kwantyzacji równym „i",

$\log_2 \frac{1}{p_i}$ – ilość informacji niesionej przez i-ty poziom (symbol),

$\log_2$ – podstawa logarytmu = miara jednostek, w których chcemy wyrazić entropię,

N – liczba poziomów kwantyzacji (dla 8 bitów mamy 256 poziomów).

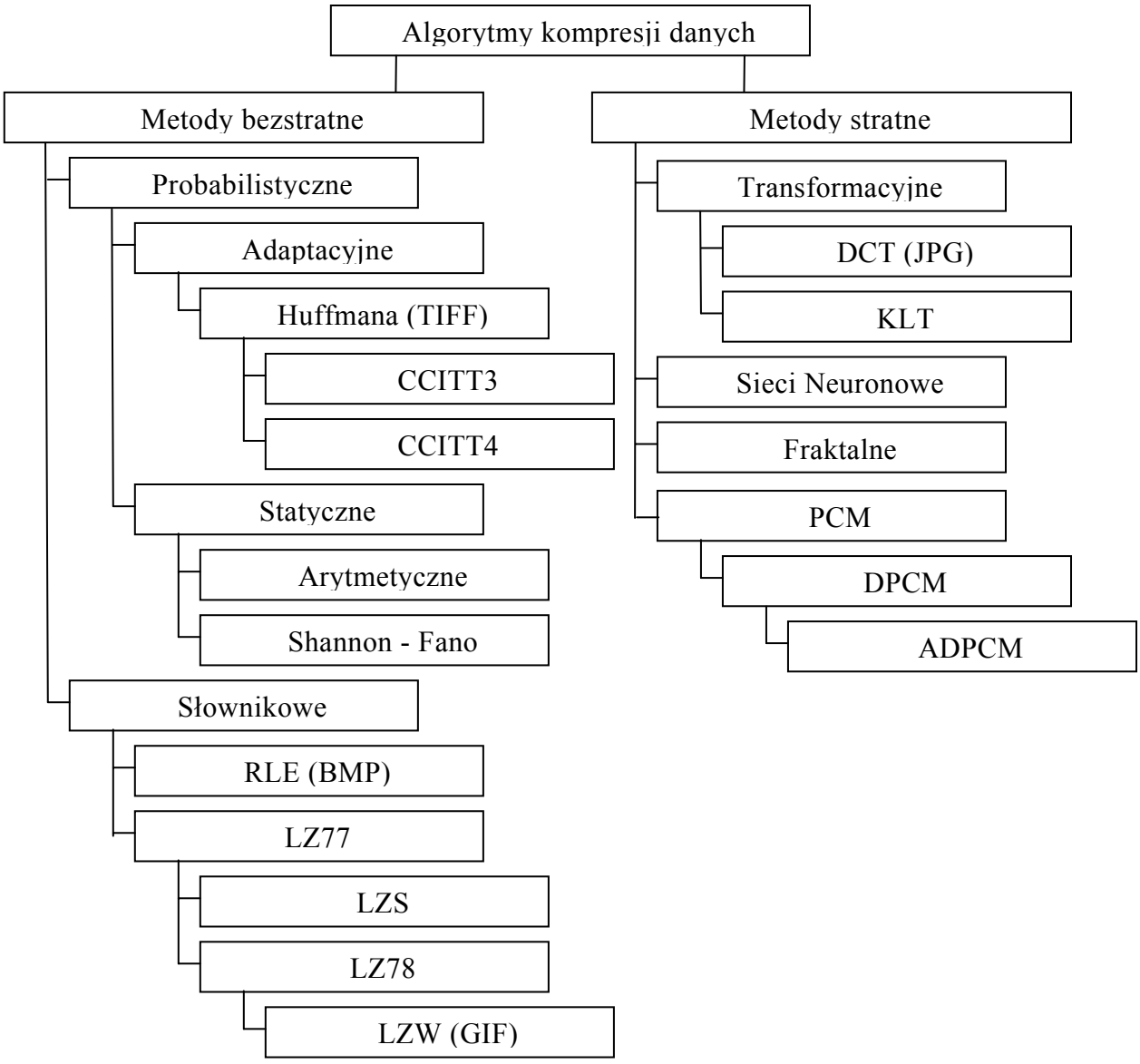
Natomiast średnia ilość informacji przypadająca na jeden symbol wyraża się zależnością:

$$h = \frac{1}{N}\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$$

Im większa jest entropia, tym więcej informacji zawiera dany obraz. Przykładowo, dla obrazu Lena użytego w testach entropia wynosi ok. 7.2 bit/piksel.

1.2 Przegląd metod kompresji

Istnieje wiele metod kompresji danych. W poniższym zestawieniu przedstawiono ogólny podział oraz opisano najpopularniejsze z nich.



Rys.1. Zestawienie metod kompresji danych na podstawie [12].

Algorytmy kompresji dzielimy na stratne i bezstratne. Do pierwszej grupy zaliczamy te z nich, które po odtworzeniu danych wykazują niewielkie zniekształcenia w stosunku do danych pierwotnych, lecz w zamian za to oferują zwykle bardzo duży stopień kompresji, który jest znacznie większy od metod oferowanych przez metody z grupy drugiej. Algorytmy stratne stosujemy zwykle do danych naturalnych takich jak dźwięk, czy obraz, natomiast bezstratne do danych, w których nie możemy sobie pozwolić na jakiekolwiek zniekształcenia.

Kompresja metodą ADPCM (ang. Adaptive Differential Pulse Code Modulation) polega na estymacji bieżącej próbki na podstawie próbki ją poprzedzającej. Wykorzystuje się tutaj fakt, że w sygnale dźwiękowym występuje silna zależność pomiędzy kolejnymi próbkami.

Metoda fraktalna opiera się na wykorzystaniu przekształceń IFS (Iterated Functions Systems) generujących fraktale.

Kompresja za pomocą przekształceń transformacyjnych, jak również za pomocą sztucznej sieci neuronowej, będzie opisana szczegółowo w dalszej części tej pracy.

Wśród algorytmów bezstratnych wyróżniamy algorytmy probabilistyczne i słownikowe. Algorytmy probabilistyczne wykorzystują statystyki prawdopodobieństwa występowania poszczególnych symboli. Przyporządkowanie odpowiednich ciągów binarnych symbolom następuje według zasady: krótszych tym, które występują częściej i dłuższych występującym rzadziej. Do tej grupy metod zaliczamy kodowanie Huffmana, Shanona-Fano oraz najlepszy, znany do tej pory, algorytm kompresji bezstratnej, którym jest kodowanie arytmetyczne [12].

Algorytmy słownikowe natomiast wywodzą się od prac J. Ziv i A. Lempela, którzy stworzyli w 1977 algorytm LZ77, a następnie w kolejnych latach jego modyfikacje LZSS i LZ78. Ten ostatni ze względu na problemy ze złożonością procesu dekompresji mającą znaczący wpływ na jej szybkość oraz problemy z przepełniającym się słownikiem pozostał jedynie opracowaniem teoretycznym. W roku 1984 Terry Welch pracujący w laboratoriach UNISYS wprowadził do algorytmu LZ78 kilka usprawnień i stworzył algorytm LZW wykorzystywany do dzisiaj w formacie zapisu grafiki GIF, jak również w protokołach kompresji danych V.42 bis.

Częstym rozwiązaniem dla algorytmów bezstratnych jest łączenie obu rodzajów kodowania z zachowaniem jednak odpowiedniej kolejności, czyli najpierw stosowany jest algorytm słownikowy, a następnie algorytm probabilistyczny

(w przeciwnym wypadku zostałaby zniszczona struktura danych i etap kodowania słownikowego nie przyniósłby już żadnych korzyści). Algorytmy wieloetapowe stosowane są w większości, jeśli nie we wszystkich popularnych kompresorach takich jak rar, zip czy arj.

Jak już wspomniano wcześniej algorytmy kompresji, w których godzimy się z częściową utratą danych lub ich zniekształceniem stosuje się do naturalnych danych takich jak obraz czy dźwięk. Np. format zapisu dźwięku MP3 przy zniekształceniach nierejestrowanych przez zmysł słuchu człowieka pozwala umieścić na jednej płycie CD 10-12 godzin muzyki, podczas gdy ta sama płyta mieści ok. jednej godziny dźwięku niepoddanego kompresji. Podobnie dzieje się podczas kompresji obrazu. Można tutaj oczywiście stosować metody bezstratne takie jak GIF (wykorzystujący algorytm LZW), TIFF (kodowanie Huffmana) czy BMP (algorytm RLE). Częściej jednak stosuje się algorytmy stratne wykorzystujące niedoskonałości ludzkiego wzroku. Najpopularniejszym formatem zapisu obrazu z tej grupy jest JPG, który potrafi zmniejszyć rozmiar pliku nawet kilkaset razy!

W niniejszej pracy omówione zostaną dwie metody kompresji stratnej, a mianowicie metoda transformacyjna wykorzystująca transformatę Karhunena-Loeve (KLT) oraz metoda wykorzystująca sztuczne sieci neuronowe.

Rozdział 2
Reprezentacja obrazu nieruchomego w postaci cyfrowej

Zamiana obrazu na postać cyfrową, formaty zapisu oraz ich wady i zalety.

2.1 Kwantyzacja – zamiana na postać cyfrową

Jeden obraz jest wart tysiąca słów. Jest to prawda spotykana na każdym kroku w codziennym życiu. Jak bowiem wytłumaczyć komuś bez pokazywania, czy też rysowania, sposób wiązania sznurowadeł, lub oddać piękno i kształt oraz barwy błyskawicy? W myśl tej zasady wielokrotnie łatwiej jest zamienić obraz na postać cyfrową i przesłać np. pocztą elektroniczną.

Najprostszym sposobem zamiany obrazu naturalnego w postaci np. zdjęcia jest użycie urządzenia zwanego skanerem, które skanując z określoną rozdzielczością zapisuje obraz punkt po punkcie, jego barwę oraz jasność w postaci cyfrowej. Proces podziału obrazu na piksele[1] nazywamy dyskretyzacją. Natomiast zamiana analogowej wartości barwy i jasności na postać cyfrową nazywamy kwantyzacją. Taką postać cyfrową obrazu można już dowolnie przetwarzać programami komputerowymi, a w szczególności poddawać procesowi kompresji. Innym sposobem stworzenia obrazu cyfrowego jest użycie specjalistycznego oprogramowania graficznego.

2.2 Formaty zapisu – ich zalety i wady

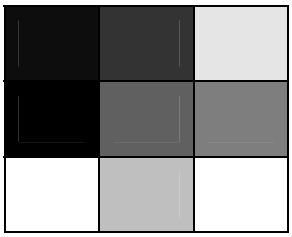
Obraz w postaci cyfrowej może przybierać dwie podstawowe reprezentacje, a mianowicie rastrową lub wektorową. Fakt ten ma kapitalne znaczenie w późniejszej obróbce tego obrazu, jego zapisie, wykorzystaniu, czy dziedzin, w których można go użyć.

Obraz rastrowy, najprościej rzecz ujmując, to tablica pikseli o indywidualnie przypisanych wartościach koloru lub jasności w przypadku obrazów w odcieniach szarości. Poniższy rysunek wyjaśnia na przykładzie obrazu 3x3 piksele zasadę jego budowy i reprezentacji na ekranie monitora.

[1] pixel – ang. picture element, co w języku polskim oznacza „punkt obrazu". W języku polskim stosuje się często w tym miejscu słowo „piksel".

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

13	51	230
0	96	128
255	191	255



Rys.2. Reprezentacja obrazu rastrowego (bitmapy).

W powyższym przykładzie każdy z pikseli obrazu jest opisany przez jego współrzędne w układzie kartezjańskim (rysunek po lewej) oraz jasność w zakresie wartości od 0 do 255 dla obrazu z odcieniami szarości (rysunek po środku). W wyniku takiego opisu otrzymujemy rysunek po prawej.

Najczęściej stosowanymi przestrzeniami barw dla mapy bitowej są:

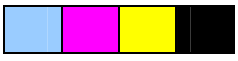
Nazwa formatu	**Liczba bitów**	**Opis formatu obrazu**
Monochromatyczny	1	Monochromatyczny zbudowany z pikseli, które przyjmują tylko dwie wartości – biały i czarny
Gray 8bit	8	256 odcieni szarości – podstawowy format niniejszej pracy.
Gray 10bit	10	1024 odcienie szarości, zastosowany w celu znacznego poprawienia jakości w stosunku do formatu „Gray 8bit”.
Kolor 8bit	8	Kolorowy ze standardową tablicą 256 barw.
Kolor 8bit - index	8	Kolorowy z tablicą 256 optymalnie wybranych barw z palety 65535 formatu „Kolor 16bit”.
Kolor 16bit	16	Kolorowy z paletą 65535 kolorów.
Kolor 24bit RGB 	24	Kolorowy z paletą 16,7mln barw. W takim formacie wyświetlany jest obraz na współczesnych monitorach CRT, a coraz częściej pojawia się także w wyświetlaczach LCD.
Kolor 32bit	32	Kolorowy z paletą 16,7 mln barw. Dodatkowe 8bitów jest przeznaczone na uzyskanie efektu przezroczystości.
CMYK 	32	System wykorzystywany w poligrafii, gdzie ze względów praktycznych reprezentacja barw została rozbita na cztery składowe: Cyan, Magenta, Yellow, Black.

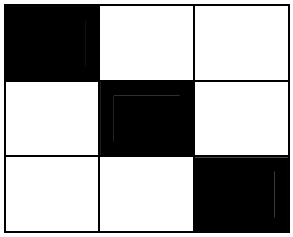
Tabela2. Zestawienie formatów map bitowych.

Oczywiście dostępnych i wykorzystywanych formatów jest znacznie więcej, lecz ze względu ograniczony rozmiar pracy, ich rzadkie występowanie oraz specjalistyczne zastosowania zostały one tutaj pominięte.

Drugim typem grafiki jest grafika wektorowa, która różni się od rastrowej tym, że wszelkiego rodzaju obiekty opisywane są wzorami, a nie są opisywane poszczególne piksele. Najlepiej widać to na poniższym przykładzie.

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

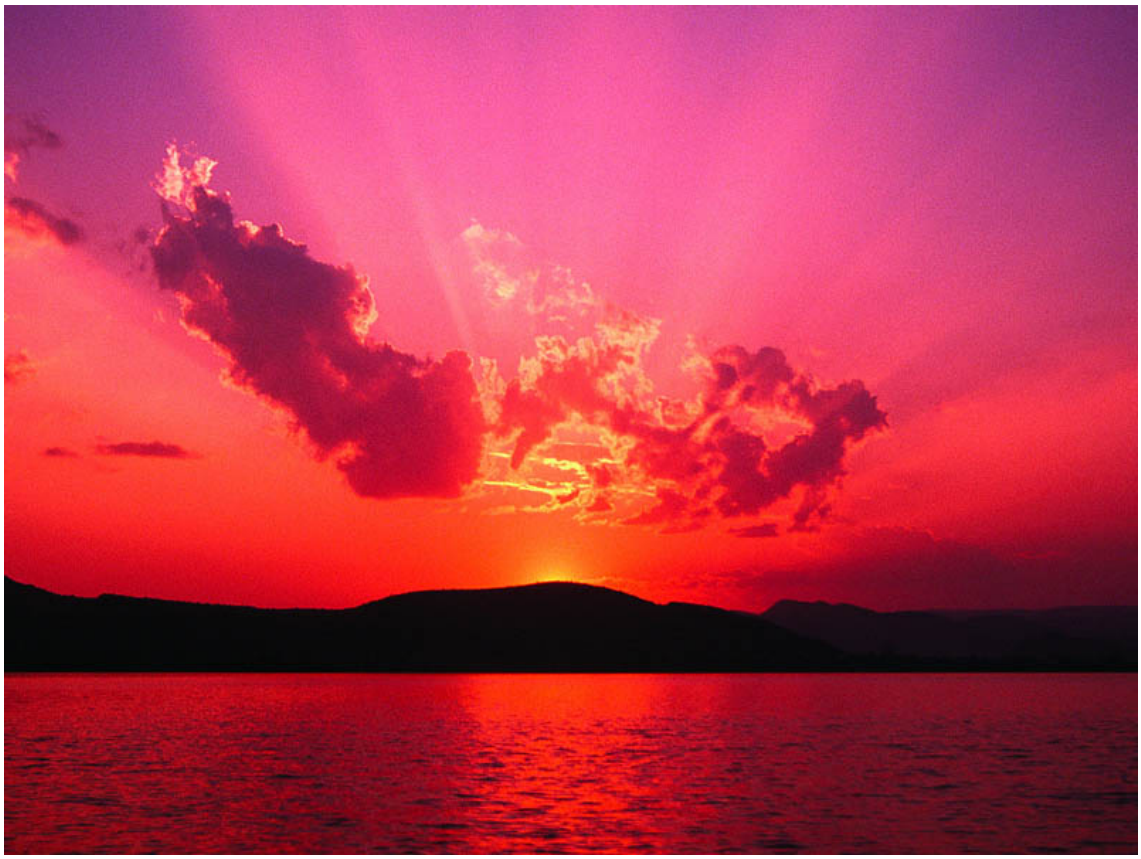
0	255	255
255	0	255
255	255	0



Rys.3. Różnice pomiędzy obrazem rastrowym, a wektorowym.

W obrazie rastrowym należy podać wszystkie piksele wraz z ich wartościami. Tutaj będzie to wyglądało następująco: 0,255,255,255,0,255,255,255,0. Należy oczywiście jeszcze podać rozmiary tego obrazu w pionie i poziomie oraz ew. położenie, paletę barw, jak również parę innych nieistotnych dla tego przykładu parametrów. W obrazie wektorowym widzimy, że jest to odcinek od punktu (1,1) do punktu (3,3) w kolorze czarnym i tak też go zapisujemy 255,1,1,3,3. Widać na tym prostym przykładzie, że jest to dużo bardziej ekonomiczny zapis dla grafiki, w której występują regularne figury geometryczne i elementy obrazu dające się zapisać w postaci matematycznej, takie jak schematy techniczne, czy wykresy. Jednak już dla kolorowej fotografii, na której widnieje piękny zachód słońca, ten rodzaj zapisu nie nadaje się w zupełności.

Rys.4.Obraz wektorowy „telefony”231x362 piksele zapisany w formacie WMF. Rozmiar pliku 21kB. Źródło – biblioteka klipart z pakietu MS Office.



Rys.5. Mapa bitowa 800x600 „Zachód słońca" zapisany w formacie JPG. Rozmiar pliku 70kB. Źródło – biblioteka klipart z pakietu MS Office.

Z każdym rodzajem grafiki związane jest format jej zapisu w pamięci masowej, np. na dysku twardym. Najpopularniejszy dla obrazów wektorowych jest format WMF, a dla obrazów bitowych formaty BMP, TIFF, GIF, JPG. Możliwych formatów zapisu na dysku jest oczywiście znacznie więcej i każdy z nich posiada swoje wady i zalety, a także może być stosowany w różnych okolicznościach i wymaganiach w stosunku do jakości samego obrazu jak również jego wielkości.

Rozdział 3
Kompresja obrazu transformatą KLT

Opis teoretyczny istoty działania transformaty Karhunena-Loeve (KLT) oraz jej implementacja w programie „Kompresja Neuronowa”.

3.1 Opis teoretyczny

Do poniższej analizy zakładam, że rozpatrujemy obraz o rozmiarach 256x256 pikseli z odcieniami szarości reprezentowanymi przez 8bitową wartość. Założenie to jest przyjęte ze względu na uproszczenie opisu transformaty i jednocześnie nie wpływa na jego istotę, czy jakość. Dodatkowo, w związku z tym, że standardowy obraz jest zbyt duży, aby rozpatrywać go jako całość, jest on dzielony na mniejsze podobrazy i dla każdego z nich przeprowadzana jest ta sama operacja kodowania/kompresji. Zwykle rozmiary podobrazu zawierają się w granicach od 2x2 do 16x16 pikseli.

Kompresja metodami transformacyjnymi polega na przekształceniu obrazu do innego układu współrzędnych i przetworzeniu statystycznie powiązanych elementów do postaci niezależnych współczynników, a następnie odrzuceniu najmniej istotnych (ze statystycznego punktu widzenia) z nich w ten sposób, aby przy późniejszym odtworzeniu pierwotnego obrazu zapewnić jak najmniejszy błąd.

Jeżeli N^2 elementów obrazu przedstawimy w postaci wektorowej $\{x(n)\}$ dla $n = 0,1,2,..,N^2-1$, to transformację do innego układu współrzędnych otrzymujemy w wyniku zależności

$$\Theta(k) = \sum_{n=0}^{N^2-1} x(n) \cdot a(k,n)$$

gdzie

$k = 0,1,2,..,N^2-1$

a(k,n) - jądro przekształcenia

$\Theta(k)$ - współczynniki transformacji

W celu odzyskania danych wejściowych stosuje się transformację odwrotną

$$x(n) = \sum_{n=0}^{N^2-1} \Theta(k) \cdot b(k,n)$$

gdzie

b(k,n) - jądro przekształcenia odwrotnego

Macierz transformacyjna $A=\{a_k\}$ składa się z wektorów bazowych $a_k=\{a(k,n)\}$, a macierz transformacji odwrotnej $B=\{b_k\}$, z wektorów bazowych $b_k=\{b(k,n)\}$, przy czym macierze A i B są macierzami ortonormalnymi i zachodzi pomiędzy nimi zależność[2]

$$B = A^{-1} = A^T$$

gdzie

B - macierz przekształcenia odwrotnego

A^{-1} - macierz odwrotna do macierzy A

A^T - transponowana macierz A

Najbardziej popularne metody transformacyjne to DCT[3], DFT[4], DWHT[5] czy KLT[6]. Spośród nich ta ostatnia jest najlepszą pod względem minimalizacji błędu średniokwadratowego. Jest to wynikiem samej istoty transformaty KLT, która polega na tym, że pod uwagę nie są brane niezależne od przetwarzanego sygnału funkcje transformacyjne, a macierz wektorów własnych estymaty macierzy kowariancji danych wejściowych.

Poniżej przedstawiony zostanie proces transformacji wraz z przykładem jej zastosowania.

W celu szczegółowego omówienia istoty metody kompresji za pomocą transformaty KLT w pierwszej kolejności zostanie zdefiniowane pojęcie wartości własnych i wektorów własnych macierzy, a dopiero po tym opisana zostanie metoda transformacji Karhunena – Loeve.

[2] Jest to tak zwana transformacja unitarna (macierz A jest macierzą unitarną).
[3] DCT – Discrete Cosine Transform (Dyskretna Transformata Cosinusów)
[4] DFT – Discrete Fourier Transform (Dyskretna Transformata Fouriera)
[5] DWHT – Discrete Walsh – Hadamard Transform (Dyskretna Transformata Walsha - Hadamarda)
[6] KLT – Karhunen–Loeve Transform (Transformata Karhunena – Loeve zwana transformatą Hotellinga).

- Wartości własne i wektory własne macierzy (na podstawie [21]).

Załóżmy, że A jest niezerową, symetryczną i rzeczywistą macierzą, a X wektorem. Przyjmując takie oznaczenia, równanie

$$Y = A \cdot X$$

przedstawia wektor Y będący obrazem wektora X przy przekształceniu o macierzy A. Następnie szukamy takiego wektora X, dla którego obraz Y jest z nim kolinearny, a współczynniki kolinearności λ wyznaczamy z równania macierzowego

$$A \cdot X = \lambda \cdot X$$

gdzie $\lambda \in R$, a ponieważ $\lambda \cdot X = \lambda \cdot I \cdot X$, to powyższe równanie można zapisać w postaci

$$A \cdot X = \lambda \cdot I \cdot X \quad \text{lub} \quad (A - \lambda \cdot I) \cdot X = 0$$

Niezerowe rozwiązanie powyższego równania istnieje wtedy i tylko wtedy gdy macierz $A - \lambda \cdot I$ jest osobliwa, tzn. gdy $\det(A - \lambda \cdot I) = 0$. Liczby λ spełniające to równanie nazywane są wartościami własnymi, a wektory X spełniające równanie $(A - \lambda \cdot I) \cdot X = 0$ wektorami własnymi macierzy A.

- Transformata KLT (na podstawie [21]).

Pierwszą czynnością, jaką należy zrobić, to podział obrazu na standardowe podobrazy o wymiarach PxP pikseli i ich przekształcenie w wektory P^2-elementowe, np.o rozmiarach 2x2 piksele zamienia się je na wektory 1x4, zgodnie ze wzorem

$$\begin{bmatrix} (1,1) & (2,1) \\ (1,2) & (2,2) \end{bmatrix} \Rightarrow \begin{bmatrix} (1,1) \\ (2,1) \\ (1,2) \\ (2,2) \end{bmatrix}$$

Tak otrzymane wektory możemy traktować jako realizacje pewnego wektora losowego $\underline{\vec{X}}$. Macierz kowariancji wektora losowego $\underline{\vec{X}}$ jest zdefiniowana wzorem

$$C_x = \mathrm{cov}(\underline{\vec{X}}) = E\{[\underline{\vec{X}} - E(\underline{\vec{X}})][\underline{\vec{X}} - E(\underline{\vec{X}})]^T\}$$

gdzie $E(\cdot)$ to operator uśrednienia statystycznego.

Aby obliczyć estymatę macierzy C_x należy najpierw obliczyć estymatę wartości średniej wektora losowego, którą oznaczymy $\overline{X}$ ze wzoru:

$$\overline{X} = \begin{bmatrix} \frac{1}{K}\sum_{i=1}^{K} X^{(i)}{}_{1,1} \\ \frac{1}{K}\sum_{i=1}^{K} X^{(i)}{}_{2,1} \\ \frac{1}{K}\sum_{i=1}^{K} X^{(i)}{}_{1,2} \\ \frac{1}{K}\sum_{i=1}^{K} X^{(i)}{}_{2,2} \end{bmatrix}$$

gdzie $X^{(i)}{}_{k,l}$ to element i-tego podobrazu, a K jest liczbą realizacji wektora losowego $\underline{\vec{X}}$.

Następnie dla każdego podobrazu obliczamy macierz

$$\begin{bmatrix} (X_{1,1}-\overline{X}_{1,1})\times(X_{1,1}-\overline{X}_{1,1}) & (X_{1,1}-\overline{X}_{1,1})\times(X_{2,1}-\overline{X}_{2,1}) & (X_{1,1}-\overline{X}_{1,1})\times(X_{1,2}-\overline{X}_{1,2}) & (X_{1,1}-\overline{X}_{1,1})\times(X_{2,2}-\overline{X}_{2,2}) \\ (X_{2,1}-\overline{X}_{2,1})\times(X_{1,1}-\overline{X}_{1,1}) & (X_{2,1}-\overline{X}_{2,1})\times(X_{2,1}-\overline{X}_{2,1}) & (X_{2,1}-\overline{X}_{2,1})\times(X_{1,2}-\overline{X}_{1,2}) & (X_{2,1}-\overline{X}_{2,1})\times(X_{2,2}-\overline{X}_{2,2}) \\ (X_{1,2}-\overline{X}_{1,2})\times(X_{1,1}-\overline{X}_{1,1}) & (X_{1,2}-\overline{X}_{1,2})\times(X_{2,1}-\overline{X}_{2,1}) & (X_{1,2}-\overline{X}_{1,2})\times(X_{1,2}-\overline{X}_{1,2}) & (X_{1,2}-\overline{X}_{1,2})\times(X_{2,2}-\overline{X}_{2,2}) \\ (X_{2,2}-\overline{X}_{2,2})\times(X_{1,1}-\overline{X}_{1,1}) & (X_{2,2}-\overline{X}_{2,2})\times(X_{2,1}-\overline{X}_{2,1}) & (X_{2,2}-\overline{X}_{2,2})\times(X_{1,2}-\overline{X}_{1,2}) & (X_{2,2}-\overline{X}_{2,2})\times(X_{2,2}-\overline{X}_{2,2}) \end{bmatrix}$$

będącą estymatą macierzy kowariancji tego konkretnego podobrazu.

Estymatę macierzy kowariancji podobrazów z całego obrazu obliczamy ze wzoru:

$$\overline{\mathrm{cov}(X)} = \begin{bmatrix} \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{1,1}-\overline{X}_{1,1})\times(X^{(i)}{}_{1,1}-\overline{X}_{1,1}) & \cdots & \cdots & \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{1,1}-\overline{X}_{1,1})\times(X^{(i)}{}_{2,2}-\overline{X}_{2,2}) \\ \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{2,1}-\overline{X}_{2,1})\times(X^{(i)}{}_{1,1}-\overline{X}_{1,1}) & \cdots & \cdots & \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{2,1}-\overline{X}_{2,1})\times(X^{(i)}{}_{2,2}-\overline{X}_{2,2}) \\ \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{1,2}-\overline{X}_{1,2})\times(X^{(i)}{}_{1,1}-\overline{X}_{1,1}) & \cdots & \cdots & \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{1,2}-\overline{X}_{1,2})\times(X^{(i)}{}_{2,2}-\overline{X}_{2,2}) \\ \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{2,2}-\overline{X}_{2,2})\times(X^{(i)}{}_{1,1}-\overline{X}_{1,1}) & \cdots & \cdots & \frac{1}{K}\sum_{i=1}^{K}(X^{(i)}{}_{2,2}-\overline{X}_{2,2})\times(X^{(i)}{}_{2,2}-\overline{X}_{2,2}) \end{bmatrix}$$

Dla estymaty macierzy kowariancji oblicza się macierz wartości własnych i odpowiadających im wektorów własnych, które dla transformaty KLT są wektorami bazowymi przekształcenia. Wartości własne powinny być

uporządkowane pod względem wartości. Do ich obliczenia stosuje się procedury numeryczne, np. tak jak w programie „Kompresja Neuronowa”.

Kompresję danych wejściowych uzyskuje się poprzez pozostawienie określonej liczby największych, najistotniejszych pod względem statystycznym wartości własnych i skojarzonych z nimi wektorów własnych. Wynikowa macierz wektorów własnych A będzie macierzą transformacyjną. Po wymnożeniu wejściowego wektora danych X przez macierz transformacyjną A otrzymujemy wektor wynikowy Y, który zawiera taką samą liczbę elementów, co wektor pozostawionych wartości własnych.

$$\mathrm{A} \times \mathrm{X} = \mathrm{Y}$$

Dekompresja – odtworzenie danych polega na wymnożeniu wektora otrzymanego w wyniku kompresji przez transponowaną macierz transformacyjną.

$$\mathrm{A}^T \times \mathrm{Y} = \mathrm{X}$$

Dla łatwiejszego zrozumienia przedstawionej metody przedstawiono poniżej przykład jej zastosowania dla trzech podobrazów o rozmiarach 2x2 piksele.

Przykład 1. Przykład zastosowania transformaty KLT.

Wejściowe podobrazy:

$$X1 = \begin{bmatrix} 5 & 17 \\ 36 & 25 \end{bmatrix} \quad X2 = \begin{bmatrix} 9 & 13 \\ 22 & 30 \end{bmatrix} \quad X3 = \begin{bmatrix} 7 & 15 \\ 29 & 29 \end{bmatrix}$$

Podobrazy zamienione na postać wektorową:

$$X1 = \begin{bmatrix} 5 \\ 36 \\ 17 \\ 25 \end{bmatrix} \quad X2 = \begin{bmatrix} 9 \\ 22 \\ 13 \\ 30 \end{bmatrix} \quad X3 = \begin{bmatrix} 7 \\ 29 \\ 15 \\ 29 \end{bmatrix}$$

Estymata wartości średniej podobrazu:

$$\overline{X} = \begin{bmatrix} \frac{1}{3}(5+9+7) \\ \frac{1}{3}(36+22+29) \\ \frac{1}{3}(17+13+15) \\ \frac{1}{3}(25+30+29) \end{bmatrix} \Rightarrow \overline{X} = \begin{bmatrix} 7 \\ 29 \\ 15 \\ 28 \end{bmatrix}$$

Estymaty macierzy kowariancji dla każdego podobrazu:

$$\operatorname{cov} X1 = X1 \times X1^T = \begin{bmatrix} (5-7)\times(5-7) & (36-29)\times(5-7) & (17-15)\times(5-7) & (25-28)\times(5-7) \\ (5-7)\times(36-29) & (36-29)\times(36-29) & (17-15)\times(36-29) & (25-28)\times(36-29) \\ (5-7)\times(17-15) & (36-29)\times(17-15) & (17-15)\times(17-15) & (25-28)\times(17-15) \\ (5-7)\times(25-28) & (36-29)\times(25-28) & (17-15)\times(25-28) & (25-28)\times(25-28) \end{bmatrix}$$

$$\operatorname{cov} X1 = \begin{bmatrix} 4 & -14 & -4 & 6 \\ -14 & 49 & 14 & -21 \\ -4 & 14 & 4 & -6 \\ 6 & -21 & -6 & 9 \end{bmatrix}$$

$$\operatorname{cov} X2 = X2 \times X2^T = \begin{bmatrix} (9-7)\times(9-7) & (22-29)\times(9-7) & (13-15)\times(9-7) & (30-28)\times(9-7) \\ (9-7)\times(22-29) & (22-29)\times(22-29) & (13-15)\times(22-29) & (30-28)\times(22-29) \\ (9-7)\times(13-15) & (22-29)\times(13-15) & (13-15)\times(13-15) & (30-28)\times(13-15) \\ (9-7)\times(30-28) & (22-29)\times(30-28) & (13-15)\times(30-28) & (30-28)\times(30-28) \end{bmatrix}$$

$$\operatorname{cov} X2 = \begin{bmatrix} 4 & -14 & -4 & 4 \\ -14 & 49 & 14 & -14 \\ -4 & 14 & 4 & -4 \\ 4 & -14 & -4 & 4 \end{bmatrix}$$

$$\operatorname{cov} X3 = X3 \times X3^T = \begin{bmatrix} (7-7)\times(7-7) & (29-29)\times(7-7) & (15-15)\times(7-7) & (29-28)\times(7-7) \\ (7-7)\times(29-29) & (29-29)\times(36-29) & (15-15)\times(29-29) & (29-28)\times(29-29) \\ (7-7)\times(15-15) & (29-29)\times(15-15) & (15-15)\times(15-15) & (29-28)\times(15-15) \\ (7-7)\times(29-28) & (29-29)\times(29-28) & (15-15)\times(29-28) & (29-28)\times(29-28) \end{bmatrix}$$

$$\operatorname{cov} X3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Estymata macierzy kowariancji:

$$\overline{\operatorname{cov} X} = \begin{bmatrix} \frac{1}{3}(4+4+0) & \frac{1}{3}(-14-14+0) & \frac{1}{3}(-4-4+0) & \frac{1}{3}(6+4+0) \\ \frac{1}{3}(-14-14+0) & \frac{1}{3}(49+49+0) & \frac{1}{3}(14+14+0) & \frac{1}{3}(-21-14+0) \\ \frac{1}{3}(-4-4+0) & \frac{1}{3}(14+14+0) & \frac{1}{3}(4+4+0) & \frac{1}{3}(-6-4+0) \\ \frac{1}{3}(6+4+0) & \frac{1}{3}(-21-14+0) & \frac{1}{3}(-6-4+0) & \frac{1}{3}(9+4+1) \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{8}{3} & \frac{-28}{3} & \frac{-8}{3} & \frac{10}{3} \\ \frac{-28}{3} & \frac{98}{3} & \frac{28}{3} & \frac{-35}{3} \\ \frac{-8}{3} & \frac{28}{3} & \frac{8}{3} & \frac{-10}{3} \\ \frac{10}{3} & \frac{-35}{3} & \frac{-10}{3} & \frac{14}{3} \end{bmatrix}$$

Wektor wartości własnych:

$$\lambda = \begin{bmatrix} 42.217 \\ 0.45 \\ 0 \\ 0 \end{bmatrix}$$

Macierz wektorów własnych:

$$\mathrm{A} = \begin{bmatrix} -0.251 & 0.879 & 0.251 & -0.318 \\ -0.084 & 0.295 & 0.084 & 0.948 \\ 0.754 & 0.028 & 0.656 & 0 \\ -0.311 & -0.342 & 0.887 & 0 \end{bmatrix}$$

W celu kompresji odrzucamy najmniej istotne wartości własne, a tym samym określamy stopień kompresji. W tym przykładzie odrzucamy najmniejsze wartości równe „0”. Dla pozostałych obliczamy skojarzone z nimi wektory własne. Tak powstała macierz jest naszą poszukiwaną macierzą transformacyjną A.

Macierz transformacyjna:

$$\mathrm{A} = \begin{bmatrix} -0.251 & 0.879 & 0.251 & -0.318 \\ -0.084 & 0.295 & 0.084 & 0.948 \end{bmatrix}$$

Przetworzone podobrazy pomniejszone o estymatę wartości średniej podobrazu:

$$\mathrm{Y1} = \begin{bmatrix} -0.251 & 0.879 & 0.251 & -0.318 \\ -0.084 & 0.295 & 0.084 & 0.948 \end{bmatrix} \times \begin{bmatrix} -2 \\ 7 \\ 2 \\ -3 \end{bmatrix} = \begin{bmatrix} 8.111 \\ -0.443 \end{bmatrix}$$

$$\mathrm{Y2} = \begin{bmatrix} -0.251 & 0.879 & 0.251 & -0.318 \\ -0.084 & 0.295 & 0.084 & 0.948 \end{bmatrix} \times \begin{bmatrix} 2 \\ -7 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} -7.793 \\ -0.505 \end{bmatrix}$$

$$Y3 = \begin{bmatrix} -0.251 & 0.879 & 0.251 & -0.318 \\ -0.084 & 0.295 & 0.084 & 0.948 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.318 \\ 0.948 \end{bmatrix}$$

Jak widać po transformacji dane zajmują tylko 50% tego co przed transformacją. W celu odtworzenia wartości wejściowych wektorów dokonujemy transformacji odwrotnej.

$$X1 = \begin{bmatrix} -0.251 & -0.084 \\ 0.879 & 0.295 \\ 0.251 & 0.084 \\ -0.318 & 0.948 \end{bmatrix} \times \begin{bmatrix} 8.111 \\ -0.443 \end{bmatrix} = \begin{bmatrix} -2 \\ 7 \\ 2 \\ -3 \end{bmatrix}$$

$$X2 = \begin{bmatrix} -0.251 & -0.084 \\ 0.879 & 0.295 \\ 0.251 & 0.084 \\ -0.318 & 0.948 \end{bmatrix} \times \begin{bmatrix} -7.793 \\ -0.505 \end{bmatrix} = \begin{bmatrix} 2 \\ -7 \\ -2 \\ 2 \end{bmatrix}$$

$$X1 = \begin{bmatrix} -0.251 & -0.084 \\ 0.879 & 0.295 \\ 0.251 & 0.084 \\ -0.318 & 0.948 \end{bmatrix} \times \begin{bmatrix} -0.318 \\ 0.948 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Sumaryczny błąd średniokwadratowy jest równy sumie wartości własnych odrzuconych w procesie transformacji i wynosi:

$$E = \sum_{i=1}^{M} \sum_{j=1}^{N} (X_j^{(i)} - Y_j^{(i)})^2$$

gdzie:

M – liczba podobrazów

N – liczba elementów obrazu w podobrazie

X – wektor wartości pikseli podobrazu wejściowego

Y – wektor odtworzony po transformacji

i – numer podobrazu

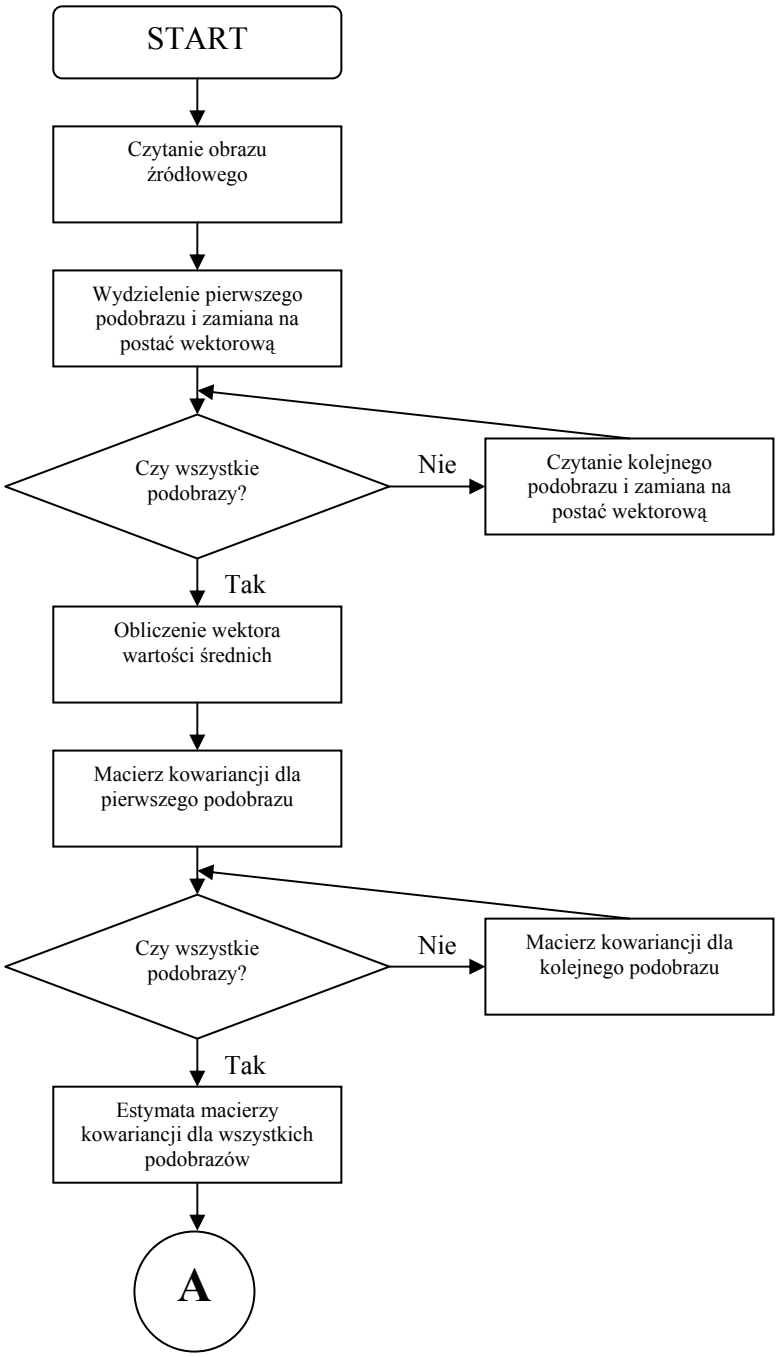
j – numer elementu w podobrazie

$$E = \sum_{i=1}^{3} \sum_{j=1}^{4} (X_j^{(i)} - Y_j^{(i)})^2 = 0$$

Koniec Przykładu 1. Przykład zastosowania transformaty KLT.

3.2 Implementacja w programie

W programie „Kompresja Neuronowa" zastosowano następujący algorytm obliczania transformaty KLT.



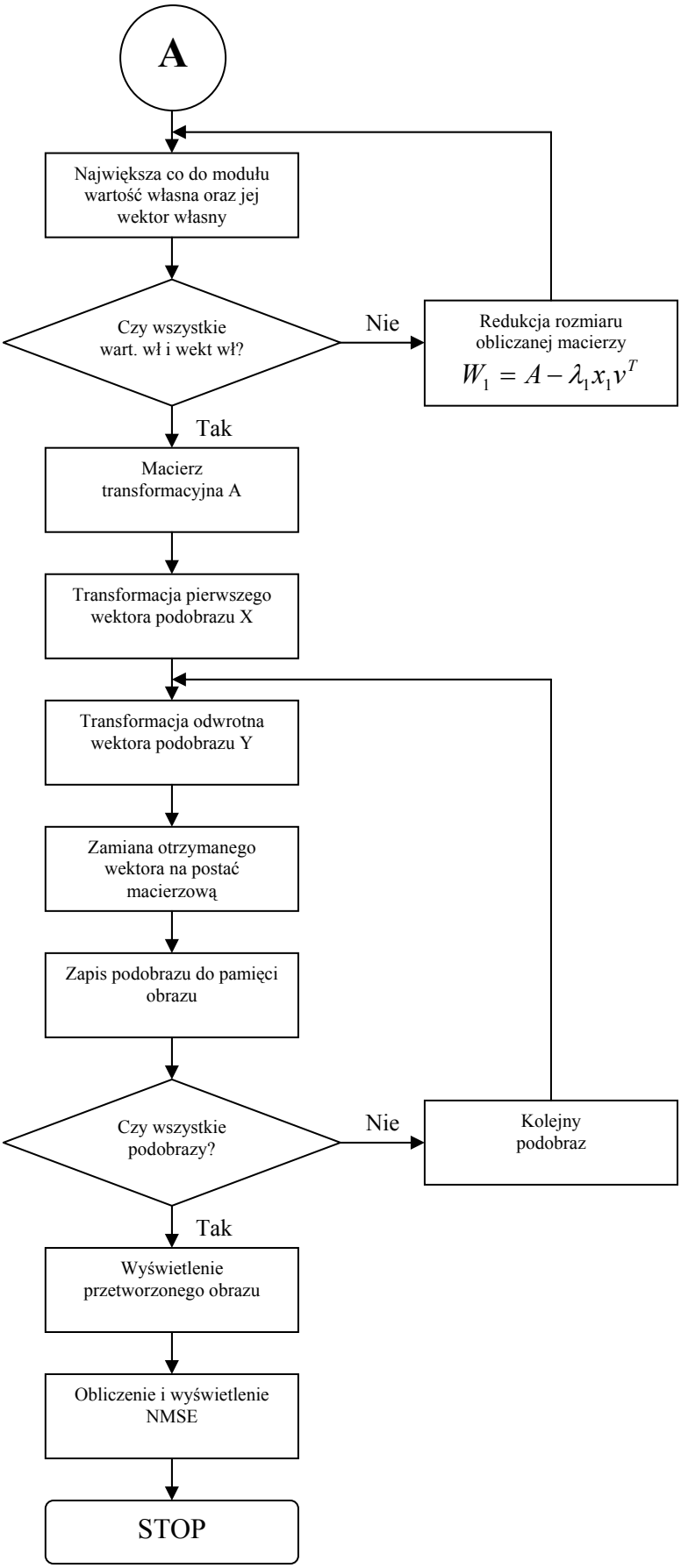


Diagram 1. Algorytm przetwarzania obrazu za pomocą transformaty KLT

Do obliczenia największych, co do modułu, wartości własnych oraz stowarzyszonych z nimi wektorów własnych zastosowano następujące procedury numeryczne zaczerpnięte z książki „Algorytmy numeryczne”[8].

```
////////////////////////////////////////////////////////////////
//procedure Rayleigh(S :MacierzSym; N :integer; var y :WierszTyp;
//                 MxIt :integer; Dok :double; var lambda :double);
//
//{Oblicza maksymalną co do modułu wartość własną lambda macierzy
//symetrycznej S stopnia N oraz stowarzyszony z nią wektor własny y
//metodą Rayleigha.
//Pozostałe oznaczenia:
//                    MxIt - maksymalna liczba iteracji.
//                    Dok - dokładność obliczeń.
////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////
//procedure Hotelling( S :MacierzSym; N,K : integer;
//                     var B :macierzTyp; var lambda :WierszTyp);
//
//{Oblicza kolejne co do modułu wartości własne i stowarzyszone
//z nimi wektory własne macierzy symetrycznej S stopnia N metoda
//Hotellinga.
//Oznaczenia:
//   K - liczba obliczanych wartości i wektorów własnych.
//   B - macierz której K kolejnych wierszy stanowią wektory własne.
//   lambda - wektor którego K kolejnych współrzędnych to wartości
//            własne macierzy S.
////////////////////////////////////////////////////////////////
```

Ich kod źródłowy jest dostępny na załączonej do ww książki dyskietce, jak również w dołączonym do pracy na płycie CD kodzie źródłowym programu „Kompresja Neuronowa”. Powyższa procedura obliczania wektorów własnych metodą Hotellinga została wybrana ze względu na swoją prostotę implementacji oraz szybkość działania. Metoda Rayleigha użyta tutaj jako pomocnicza, jest metodą odnajdywania największej, co do modułu wartości własnej macierzy symetrycznej.

Następnie obliczony jest stowarzyszony z nią wektor własny. W przypadku, gdy chcemy policzyć kolejne, największe, co do modułu wartości własne, dokonujemy redukcji obliczanej macierzy. Pod aktualnie obliczoną wartość własną podstawiane jest zero. W tym wypadku druga w kolejności wartość własna staje się największą i zostaje obliczona. Czynność powtarzamy, aż do uzyskania wszystkich potrzebnych nam wartości własnych i wektorów własnych. Taką metodę podejścia zaproponował Hotelling, co przy przetwarzaniu podobrazów o większych rozmiarach ma zasadnicze znaczenie, ze względu na szybkość działania. Dlaczego bowiem obliczać 256 wektorów własnych dla podobrazu 16x16 pikseli, skoro na przykład potrzeba tylko pierwszych 16? Optymalizacja pod tym względem daje nam w krótkim czasie, liczonym w sekundach, wyniki nawet dla względnie dużych rozmiarów podobrazu.

Rozdział 4
Sieci neuronowe

Historia powstania i rozwoju sieci neuronowych. Opis zasad ich budowy, działania i funkcjonowania. Przykład zastosowania w programie „Kompresja Neuronowa”

4.1 Krótki rys historyczny

Dystans, jaki dzieli nas od zwierząt to prawie dokładnie sześć cali, czyli tyle ile jest pomiędzy uszami. Stwierdzenie to, w sposób nieco humorystyczny oddaje sedno różnicy pomiędzy ludźmi, a resztą żywych organizmów na ziemi. A polega ona na budowie i funkcjonalności naszego mózgu. Korzystamy z niego codziennie, w każdej chwili, nawet, gdy śpimy, a jednocześnie jest on chyba najsłabiej poznanym obiektem, jaki znany jest dzisiejszej nauce. Dlatego też od pewnego czasu, a dokładniej od 1943 roku, kiedy to McCulloch i Pitts przedstawili w postaci układu arytmetyczno-logicznego pierwszy formalny model neuronu, ludzie nauki zaczęli próby zgłębienia tajemnicy mózgu i procesów w nim zachodzących przy pomocy elektronicznych maszyn liczących, które w dzisiejszych czasach zostały zastąpione przez superkomputery o ogromnych mocach obliczeniowych. Jednak nawet tak potężne maszyny nie są w stanie dorównać człowiekowi w czynnościach dla nas tak prostych i oczywistych jak rozpoznawanie innych osób, zapachów, rozumienie mowy, taniec, śpiew. Dlaczego tak się dzieje, oto zagadka, która nadal nie jest do końca rozwiązana. Jedną z prób symulacji procesów zachodzących w mózgu jest modelowanie pojedynczych neuronów, a następnie budowanie z nich sieci. Mózg ludzki składa się z ok. 10^{11} neuronów, a każdy z neuronów posiada około 10^4 synaps doprowadzających do niego sygnał. Dla porównania sieć użyta do kompresji obrazu będącej przedmiotem tej pracy składa się z maksymalnie trzech warstw po 2500 neuronów i tyleż samo synaps do każdego z nich, jednak zwykle, np. dla podobrazu o rozmiarach 4x4 piksele jest to sieć 16x4x16 neuronów z 4 do 16 synaps dla każdego z nich. Widać na tym przykładzie, że jest to naprawdę niewiele w porównaniu z mózgiem przeciętnego „zjadacza chleba”.

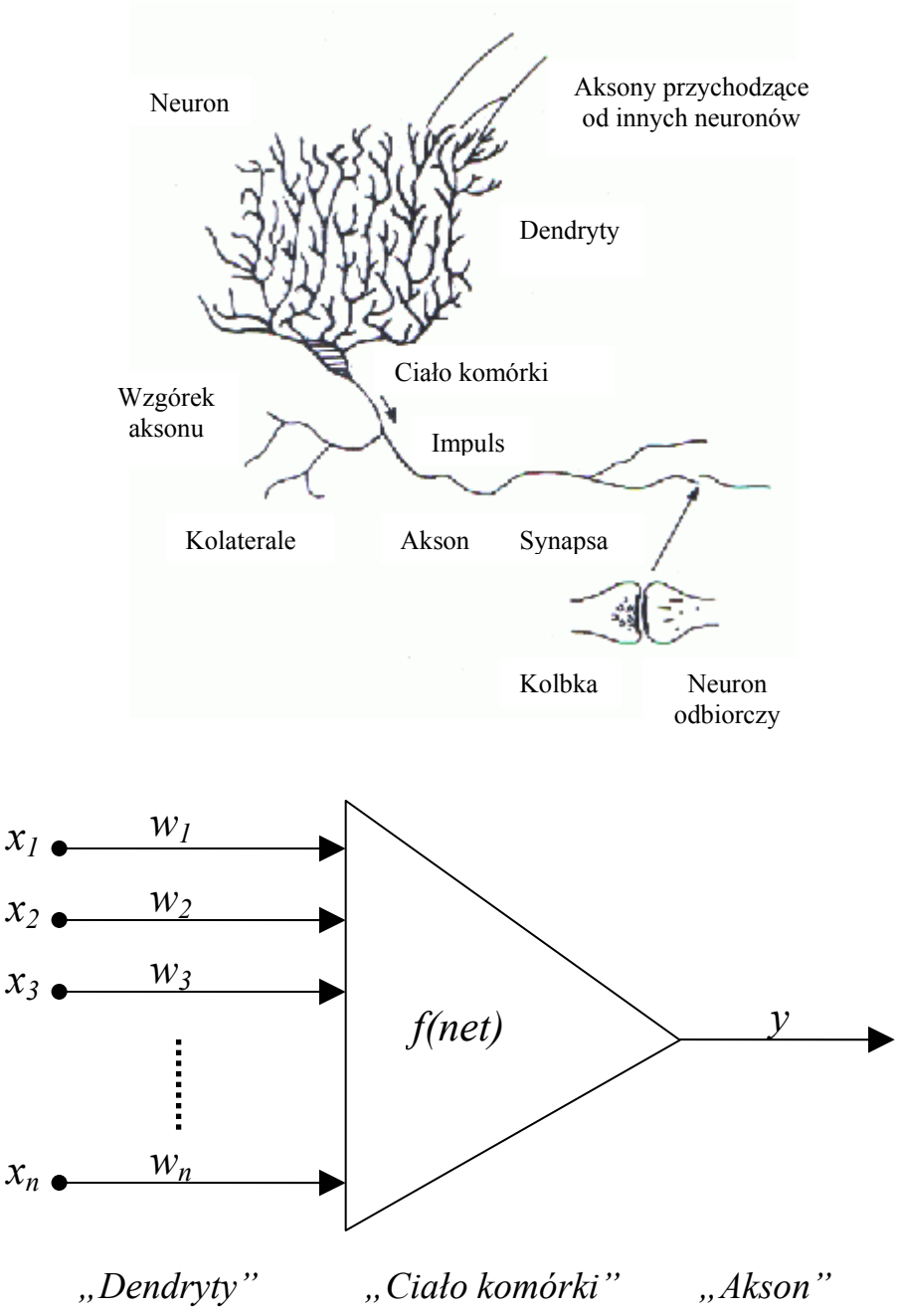
Najważniejsze prace oraz przełomowe odkrycia w dziedzinie sieci neuronowych przedstawia poniższa tabela.

Data	Opis wydarzenia
1943	McCulloch i Pitts przedstawiają pierwszy formalny opis modelu neuronu w postaci układu arytmetyczno-logicznego.
1949	Odkrycie przez Donalda Hebba możliwości przechowywania informacji w strukturze połączeń pomiędzy neuronami. Pierwsza propozycja metody uczenia polegająca na zmianach wag tych połączeń – *Reguła Hebba*.
1954	Minsky buduje pierwszą sieć neuronową.
1958	Rosenblatt buduje *perceptron* – maszynę klasyfikującą obrazy i stosującą do nauki modyfikacje wag połączeń pomiędzy neuronami.
1960	Budowa urządzenia ADALINE i jego rozwinięcia MADALINE do rozpoznawania obrazów, przewidywania pogody i sterowania adaptacyjnego.
1960, 1962	Widrow i Hoff proponują nową efektywną metodę uczenia sieci.
1965, 1968	Nilsson wydaje monografię podsumowującą osiągnięcia w dziedzinie sieci neuronowych.
1969	Minsky i Papert wydają książkę poddającą wątpliwości możliwość uczenia sieci wielowarstwowych, co zapoczątkowało długi okres stagnacji w tej dziedzinie.
1972, 1977	Amari prowadzi badania nad uczeniem się sieci z elementami progowymi.
1980	Fukushima i Miyaka rozwijają strukturę sieci *neokognitron*, która przy rozpoznawaniu obrazów próbuje naśladować zachowanie organizmów żywych.
1974	Werbos proponuje nowe mechanizmy uczenia sieci warstwowych zauważone zostaną dopiero 10 lat później.
1977, 1984, 1987	Kohonen i Anderson prowadzą badania nad pamięciami asocjacyjnymi.
1982	Kohonen opracowuje sieci uczące się bez nauczyciela.
1977, 1982	Grossberg i Carpenter rozwijają teorię adaptacyjnej sieci neuronowej.
1982, 1984	Hopfield wprowadza rekurencyjną architekturę pamięci neuronowych.
1986	McClelland i Rumelhart wydają dwutomową monografię nt. równoległego przetwarzania rozproszonego.
Lata późniejsze	Gwałtowny rozwój badań nad sieciami neuronowymi, ich zastosowaniem w przemyśle, medycynie, życiu codziennym i technice wojskowej. Metody optymalizacji, nieliniowe procesy złożone, procesy chaotyczne i algorytmy genetyczne podlegające rekombinacjom, mutacjom i selekcji.

Tabela 3. Zestawienie najważniejszych prac i przełomowych odkryć w dziedzinie badań nad budową i zastosowaniem sztucznych sieci neuronowych na podstawie [5].

4.2 Zasada działania

Podstawą do budowy sztucznych sieci neuronowych jest model neuronu zbudowany na wzór jego biologicznego odpowiednika.



Rys.6. Schemat neuronu biologicznego (na podstawie Rys. 2.2. ze strony 34 w książce [5]) oraz schemat jego funkcjonalnego odpowiednika zastosowanego w niniejszej pracy.

W neuronie biologicznym sygnały pochodzące z zewnątrz (od receptorów rejestrujących zdarzenia i sygnały pochodzące zarówno z zewnątrz, jak i z wewnątrz organizmu) i z wewnątrz, to znaczy od siebie samego (sprzężenie zwrotne) lub od innych neuronów są transmitowane przez sieć dendrytów (około 10^4 dla każdego neuronu) do ciała komórki, które znajduję się w środku tej pajęczyny połączeń. Tam wszystkie sygnały są sumowane i po przekroczeniu pewnego progu wytwarzany jest impuls, który wysyłany jest w postaci sygnału nerwowego poprzez akson lub kolaterale do innych neuronów lub ośrodków decyzyjnych/wykonawczych. Impulsy te mogą być hamujące lub pobudzające dla innych neuronów.

Podobnie dzieje się w modelu sztucznego neuronu. Sygnały wejściowe $\boldsymbol{x_i}$ zostają przemnożone przez odpowiadające im wagi $\boldsymbol{w_i}$, a następnie sumowane w łączny sygnał pobudzenia neuronu ***net*** i przetwarzane w wynik ***y*** funkcją progową zwaną także funkcją aktywacji ***f(net)***.

$$net = \sum_{i=1}^{n} x_i w_i$$

gdzie:

net	-	łączny sygnał pobudzenia
x_i	-	sygnał pobudzenia na wejściu i
w_i	-	waga sygnału na wejściu i
n	-	liczba wejść

Funkcje aktywacji mogą przyjmować wartości dyskretne lub ciągłe, przy czym obie z nich mogą być unipolarne lub bipolarne. Pierwszą w historii zaproponowaną przez McCulloch & Pitts [12] była funkcja unipolarna dyskretna opisana wzorem:

$$y^{k+1} = \begin{cases} 1, gdy \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0, gdy \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$

gdzie:

k	-	kolejne momenty czasu
T	-	wartość progowa

Gdy y_k będzie przyjmować wartości (-1,1), to wtedy funkcja ta będzie nazywać się bipolarną.

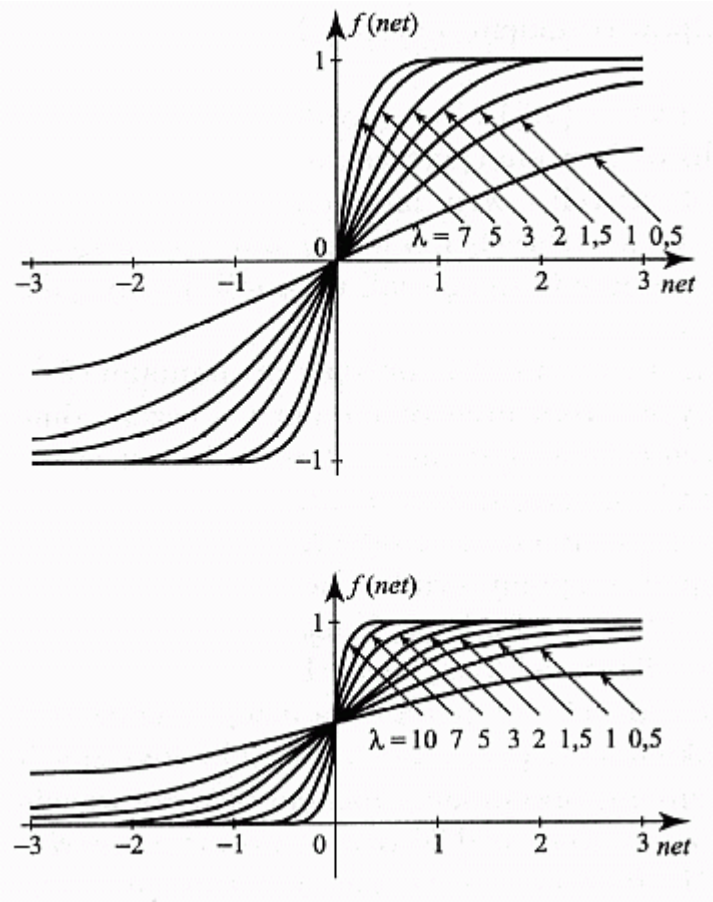
Najczęściej jednak spotyka się ciągłą funkcję aktywacji, która przyjmuje różne postacie funkcji sigmoidalnej. Wśród nich najpopularniejsze są dwie postacie:

- Unipolarna, ciągła funkcja aktywacji:

$$f(net) = \frac{1}{1 + \exp(-\lambda net)}$$

- Bipolarna, ciągła funkcja aktywacji:

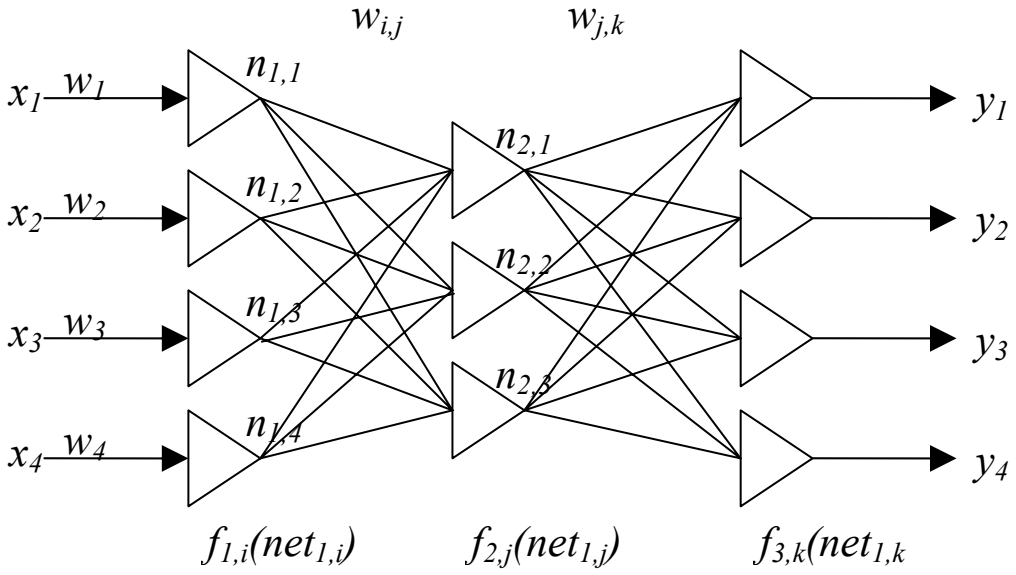
$$f(net) = \frac{2}{1 + \exp(-\lambda net)} - 1$$



Rys.7. Sigmoidalne, ciągłe funkcje aktywacji neuronu. U góry bipolarna, a na dole unipolarna. Źródło rys.2.5. str.40 [5].

W programie „Kompresja Neuronowa" zastosowano obie z powyższych funkcji, jednak wersja bipolarna daje dużo lepsze efekty i jest przy tym stabilniejsza. Dodatkowo w kodzie źródłowym pozostawiono przygotowane miejsce do bardzo prostej rozbudowy o funkcje dyskretne. Jednak ze względu na bardzo ograniczony zakres możliwych wartości, które otrzymuje się na wyjściu, nie nadają się one do kompresji obrazów. Dla przykładu, dla małego podobrazu o rozmiarze 2x2 piksele o 256 możliwych do przyjęcia wartościach, mamy na wyjściu sieci wektor 4 neuronów. Przyjmują one wartości binarne a więc mogą reprezentować tylko 2^4 stanów, podczas gdy dla tego podobrazu powinno to być $(2^8)^4$,czyli 2^{32} wartości.

Z takich właśnie neuronów konstruuje się sztuczne sieci neuronowe. Mogą one przyjmować struktury jedno i wielowarstwowe. W zależności od obszarów zastosowania i funkcji, jakie mają spełniać, sieci takie mogą przyjmować różne formy oraz wykorzystywać różne funkcje aktywacji.



Rys.8. Przykład trójwarstwowej sieci neuronowej.

Dla sieci neuronowej z rys.8. algorytm przetwarzania przedstawiono na poniższym diagramie.

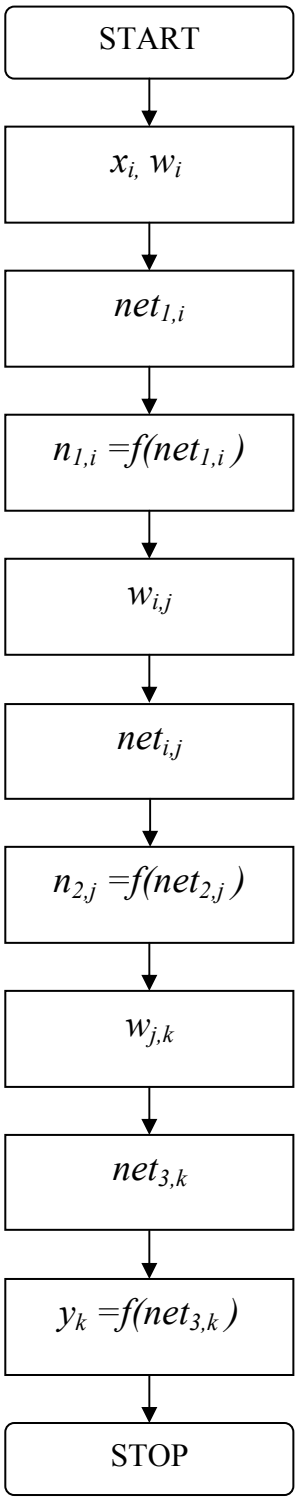


Diagram 2. Algorytm przetwarzania podobrazu za pomocą sieci neuronowej.

W diagramie 2 zastosowano następujące oznaczenia:

Dla warstwy wejściowej:

- Wektor danych wejściowych

$$x^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}$$

- Wektor wag

$$w_i^T = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix}$$

- Pobudzenie łączne dla neuronów

$$net_{1,i} = \begin{cases} net_{1,1} = x_1 w_1 \\ net_{1,2} = x_2 w_2 \\ net_{1,3} = x_3 w_3 \\ net_{1,4} = x_4 w_4 \end{cases}$$

- Wartości wyjściowe neuronów

$$n_{1,i} = \begin{cases} n_{1,1} = f(net_{1,1}) = \dfrac{2}{1+\exp(-\lambda net_{1,1})} - 1 \\ n_{1,2} = f(net_{1,2}) = \dfrac{2}{1+\exp(-\lambda net_{1,2})} - 1 \\ n_{1,3} = f(net_{1,3}) = \dfrac{2}{1+\exp(-\lambda net_{1,3})} - 1 \\ n_{1,4} = f(net_{1,4}) = \dfrac{2}{1+\exp(-\lambda net_{1,4})} - 1 \end{cases}$$

Dla warstwy środkowej (ukrytej):

- Macierz wag

$$w_{i,j} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \\ w_{4,1} & w_{4,2} & w_{4,3} \end{bmatrix}$$

- Pobudzenie łączne dla poszczególnych neuronów

$$net_{2,j} = \begin{cases} net_{2,1} = \sum_{i=1}^{4} n_i w_{i,1} \\ net_{2,2} = \sum_{i=1}^{4} n_i w_{i,2} \\ net_{2,3} = \sum_{i=1}^{4} n_i w_{i,3} \end{cases}$$

- Wartości wyjściowe neuronów

$$n_{2,j} = \begin{cases} n_{2,1} = f(net_{2,1}) = \dfrac{2}{1+\exp(-\lambda net_{2,1})} - 1 \\ n_{2,2} = f(net_{2,2}) = \dfrac{2}{1+\exp(-\lambda net_{2,2})} - 1 \\ n_{2,3} = f(net_{2,3}) = \dfrac{2}{1+\exp(-\lambda net_{2,3})} - 1 \end{cases}$$

Dla warstwy wyjściowej:

- Macierz wag warstwy wyjściowej

$$w_{j,k} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} & w_{4,1} \\ w_{1,2} & w_{2,2} & w_{3,2} & w_{4,2} \\ w_{1,3} & w_{2,3} & w_{3,3} & w_{4,3} \end{bmatrix}$$

- Pobudzenie łączne dla neuronów warstwy wyjściowej

$$net_{3,k} = \begin{cases} net_{3,1} = \sum_{i=1}^{3} n_i w_{i,1} \\ net_{3,2} = \sum_{i=1}^{3} n_i w_{i,2} \\ net_{3,3} = \sum_{i=1}^{3} n_i w_{i,3} \\ net_{3,4} = \sum_{i=1}^{3} n_i w_{i,4} \end{cases}$$

- Wartości wyjściowe neuronów

$$y_k = \begin{cases} y_1 = f(net_{3,1}) = \dfrac{2}{1+\exp(-\lambda net_{3,1})} - 1 \\ y_2 = f(net_{3,2}) = \dfrac{2}{1+\exp(-\lambda net_{3,2})} - 1 \\ y_3 = f(net_{3,3}) = \dfrac{2}{1+\exp(-\lambda net_{3,3})} - 1 \\ y_4 = f(net_{3,4}) = \dfrac{2}{1+\exp(-\lambda net_{3,4})} - 1 \end{cases}$$

Widać tutaj podobieństwo do transformaty KLT opisanej w rozdziale 3. Podobieństwo to jest na tyle duże, że jeśli założyć, że funkcja aktywacji jest równa pobudzeniu łącznemu $f(net) = net$, to wtedy możemy obliczyć wagi sieci po prostu podstawiając macierz transformacyjną pod macierz wag. Zależność ta działa również w drugą stronę, tzn. ucząc sieć dwuwarstwową z taką samą liczbą neuronów we wszystkich warstwach otrzymujemy macierz wektorów własnych macierzy kowariancji wektora wejściowego w postaci wag pomiędzy warstwą pierwszą i drugą, oraz tę samą macierz transponowaną (odwrotną) w postaci wag pomiędzy drugą i trzecią.

4.3 Uczenie sieci neuronowych

Jedną z podstawowych cech odróżniających sieci neuronowe od typowego programu sekwencyjnego jest sposób implementacji wiedzy i algorytmów rozwiązujących stawiane im zadania. W tradycyjnych programach proces ten odbywa się jeszcze na etapie kodowania wcześniej zaprojektowanego, ściśle opisanego algorytmu, natomiast w sieciach neuronowych wiedza jest „przyswajana" w procesie uczenia. Mamy dwa podstawowe rodzaje metod uczenia, a mianowicie „bez nauczyciela" lub „z nauczycielem". W pierwszej metodzie sieć sama modyfikuje swoje wagi w zależności od tego, co jest podawane na wejściu. Sygnałem uczącym jest sygnał wejściowy sieci. Wykorzystuje się tutaj zjawisko znane z biologii, gdzie wiadomo, że jeśli jakiś neuron często i systematycznie pobudza inny neuron, to połączenie pomiędzy nimi staje się mocniejsze, przez co wzrasta skuteczność tego pobudzenia. Przykładem praktycznej realizacji tego stwierdzenia jest reguła Hebba, którą jej twórca ogłosił już w 1949 roku [13].

Wynik działania neuronu określa wzór:

$$y_i = f(w_i^T x)$$

Przyrost wartości wektora wag:

$$\Delta w_i = \beta \cdot f(w_i^T x) \cdot x$$

gdzie:

Δw_i	-	przyrost wartości wagi *i*
β	-	współczynnik uczenia
$f(w_i^T x)$	-	funkcja aktywacji obliczona od łącznego pobudzenia
x	-	wektor wartości wejściowych
$w_i^T x$	-	łączne pobudzenie neuronu

Modyfikując w ten sposób wagi wejść, sieć uczy się np. rozróżniać klasy obiektów/danych. Jest to jednak metoda zdecydowanie wolniejsza od drugiej z nich – z nauczycielem.

Metoda nauki „z nauczycielem" polega na podawaniu danych wejściowych do sieci, a następnie badaniu, czy wynik otrzymany po przetworzeniu jest zgodny z wynikiem, jaki chcemy uzyskać. Jeśli „nauczyciel" zdecyduje, że wynik uzyskany za bardzo różni się od oczekiwanego, to następuje korygowanie wag według odpowiedniego algorytmu i proces jest powtarzany. Podstawowym kryterium oceny działania neuronu jest poziom błędu średniokwadratowego, a metodą nauki wyprowadzoną jako wynik minimalizacji tego błędu jest *Reguła delta* [5].

Dla bipolarnej, ciągłej funkcji aktywacji:

$$y = f(net) = \frac{2}{1 + \exp(-\lambda net)} - 1$$

oraz jej pochodnej równej:

$$f'(net) = \frac{2\exp(-net)}{[1 + \exp(-net)]^2} - 1 = \frac{1}{2}(1 - y^2) \quad \text{dla} \quad \lambda = 1$$

otrzymujemy wzór na korekcję wag:

$$\Delta w_i = \eta(d_i - y_i) f'(net_i) x = \frac{1}{2}\eta(d_i - y_i)(1 - y_i^2)$$

gdzie:

η	-	współczynnik korekcji
d_i	-	wartość żądana na wyjściu neuronu
y_i	-	wartość otrzymana na wyjściu neuronu
$f'(net_i)$	-	pochodna funkcji aktywacji
x	-	wektor danych wejściowych
Δw_i	-	przyrost wartości wagi *i*

Do zastosowań programistycznych stosuje się między innymi gradientową metodę wyznaczania minimum funkcji wielu zmiennych. W naszym przypadku funkcją wielu zmiennych jest błąd $E(w)$. Na początku przyjmuje się dowolny (lub określony według pewnych zasad) punkt *w*. Następnie oblicza się dla niego gradient $\nabla E(w)$. W końcu nową wartość *w* uzyskujemy przesuwając go w kierunku ujemnego gradientu – kierunku najszybszego spadku wartości funkcji. Algorytm ten można zapisać w postaci

$$w^{k+1} = w^k - \eta \nabla E(w^k)$$

gdzie:

k	-	numer kroku uczenia
η	-	współczynnik uczenia
w^k	-	wektor wag w kroku *k*

Gradient przyjmuje tutaj postać:

$$\nabla E(w) = -\frac{1}{2}\eta(d - y)(1 - y^2)x$$

Stąd otrzymujemy algorytm uczenia – korekcji wag dla neuronu z bipolarną, ciągłą funkcją aktywacji:

$$w^{k+1} = w^k + \frac{1}{2}\eta(d^k - y^k)(1 - y^{k2})x^k$$

Reguła delta odnosi się do sieci jednowarstwowych. Można ją jednak zmodyfikować i rozszerzyć również na sieci wielowarstwowe. Wtedy mówimy

o *metodzie propagacji wstecznej błędu*.[5][7][10] Jej odkrycie przyniosło renesans w dziedzinie wielowarstwowych sieci neuronowych po okresie zastoju, jaki nastąpił w wyniku opublikowania książki [14], w której panowie Minsky i Papert udowadniali brak możliwości uczenia sieci wielowarstwowych. Nazwa metody pochodzi od kolejności obliczania błędów w kolejnych warstwach sieci i modyfikowania wartości wag. Sygnał błędu jest obliczany od warstwy ostatniej do pierwszej i w takiej też kolejności są aktualizowane wagi.

Sygnał błędu dla warstwy wyjściowej

$$\delta_{wyj}^{k+1} = \frac{1}{2}(d^k - y^k)(1 - y^{k2})$$

Uaktualnienie wag pomiędzy warstwą ukrytą (środkową), a wyjściową

$$w_{wyj}^{k+1} = w_{wyj}^{k} + \eta \delta_{wyj}^{k} y^k$$

Sygnał błędu dla warstwy środkowej

$$\delta_{sro}^{k+1} = \frac{1}{2}(1 - y^{k2}) \sum_{k=1}^{K} \delta_{wyj}^{k} w^k$$

Uaktualnienie wag pomiędzy warstwą wejściową, a środkową

$$w_{sro}^{k+1} = w_{sro}^{k} + \eta \delta_{sro}^{k} x^k$$

W przypadku większej liczby warstw ukrytych, dla każdej z nich postępujemy podobnie jak dla wyżej opisanej warstwy środkowej.

Algorytm wstecznej propagacji błędu jest jednak bardzo wolno zbieżny, dlatego też stosuje się różne metody przyspieszające proces uczenia. Metod tych jest sporo, a ich opis wykracza poza ramy tej pracy. Warto jednak wspomnieć o ich istnieniu i najskuteczniejszych z nich [10], a są to:

- metoda zmiennej metryki zwana BFGS (Broyden-Goldfarb-Fletcher-Shanno),
- algorytm Levenberga-Marquardta,
- metoda gradientów sprzężonych z regularyzacją Møllera,
- algorytmy heurystyczne QPROP i RPROP.

Optymalizują one zarówno długość, jak i kierunek wektora, o jaki zmieniają się wagi podczas każdej iteracji uczenia sieci. Skuteczność optymalizacji wprowadzanej przez powyższe algorytmy przedstawia tabela nr.4.

Nazwa algorytmu	Średni czas	Relacja do najszybszego
Levenberg-Marquardt	1.14	1.00
BFGS	5.22	4.58
Gradientów sprzężonych z regularyzacją Møllera	6.09	5.34
Zmienny współczynnik uczenia	27.69	24.29
Testowana sieć trzywarstwowa 1-5-1 użyta do obliczenia funkcji sinus. Zakończenie algorytmu po zmniejszeniu błędu średniokwadratowego do wartości 0.002.		

Tabela 4. Zestawienie szybkości kilku algorytmów uczących na podstawie materiałów z internetu [19].

W programie „Kompresja Neuronowa" zastosowano trzy metody przyspieszenia procesu uczenia, pozostawiając jednak miejsce dla powyższych, bardziej skomplikowanych, ale jednocześnie skuteczniejszych metod.

Pierwszą z wykorzystanych metod jest optymalizacja współczynnika uczenia WSP, który jest dobierany automatycznie w zależności od aktualnego procesu uczenia. Polega ona na zwiększaniu współczynnika uczenia w momencie, gdy różnica błędu średniokwadratowego w dwóch kolejnych iteracjach jest niewielka, i jego zmniejszaniu, podczas gdy ta różnica jest znaczna. Takie podejście powoduje spowolnienie procesu uczenia dla bardzo dużych zmian, a jego przyspieszenie, gdy proces już się ustabilizował i staje się bardzo powolny.

Drugą metodą jest dodanie czynnika momentu. Powoduje to „wygładzenie" procesu uczenia i wyhamowanie dla większych wartości współczynnika uczenia oraz przyspieszenie tego procesu dla wartości mniejszych.

Trzecia polega na przybliżeniu wartości początkowych wag, na podstawie macierzy transformacyjnej otrzymanej w wyniku transformaty KLT. Niestety dokładne przybliżenie można zastosować tylko w momencie, gdy funkcja aktywacji neuronu jest funkcją jednostkową lub, co najwyżej liniową. W przypadku nieliniowej funkcji aktywacji, dającej zdecydowanie lepsze rezultaty w zakresie stawianych tutaj przed nią zadań, nie jest to takie oczywiste, ale można się pokusić o przybliżone ustawienie wag, które będzie początkowo i tak zdecydowanie lepsze niż wartości losowe, co zostanie wykazane w następnym rozdziale.

X	wejście	wagi	net	f(net)		wagi	net	net(f(net))	f(net)	Y
5	0.0196	4.8533				32.3556				
36	0.1412	-4.2200	-0.2992	-0.0449		0.5333	0.8707	0.1306	0.0196	4.9949
17	0.0667	0.5600				30.4889				
25	0.0980	1.6733								
						-28.1333				
5	0.0196	0.0800				-16.6222	6.2751	0.9412	0.1402	35.7627
36	0.1412	-2.4933	-1.0561	-0.1571		30.1333				
17	0.0667	-1.9667								
25	0.0980	-5.8600				3.7333				
						-13.1111	1.6146	0.2421	0.0363	9.2573
5	0.0196	4.5733				-3.7333				
36	0.1412	4.5200	0.5264	0.0788						
17	0.0667	-0.5600				11.1556				
25	0.0980	-1.6733				-39.0667	4.8069	0.7209	0.1077	27.4664
						-11.1556				

Tabela 5. Tabela obliczeń dla optymalizacji wartości wag transformatą KLT (Opracowanie własne).

Wyjaśnienie do tabeli 5. – opis poszczególnych kolumn

1. „X” – przykładowe wartości wektora wejściowego X,
2. „wejście” – unormowane wartości wektora X,
3. „wagi” – wagi obliczone na podstawie KLT,
4. „net” – łączne pobudzenie neuronu w warstwie środkowej obliczone jako suma iloczynów odpowiednich wartości z kolumn „wejście” i „wagi”.
5. „f(net)” – funkcja aktywacji $f(net) = \frac{2}{1+\exp(-\lambda net)} - 1$ obliczona na łącznym pobudzeniu „net”.
6. Kolumny z warstwy drugiej obliczone analogicznie do pierwszej.
7. „Y” – wektor wartości wyjściowych.

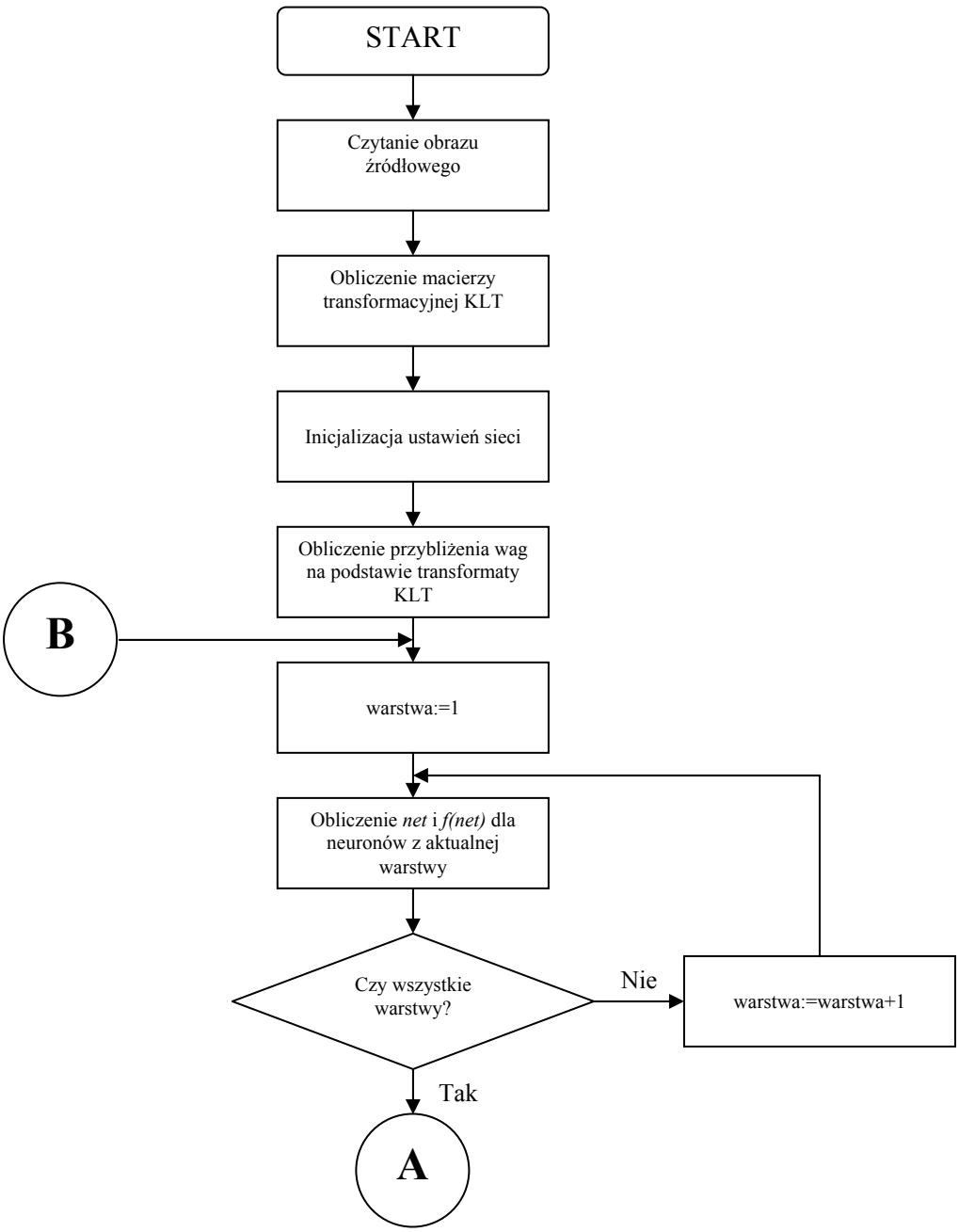
Do obliczenia wag początkowych użyte zostało przekształcenie wzoru na funkcję aktywacji

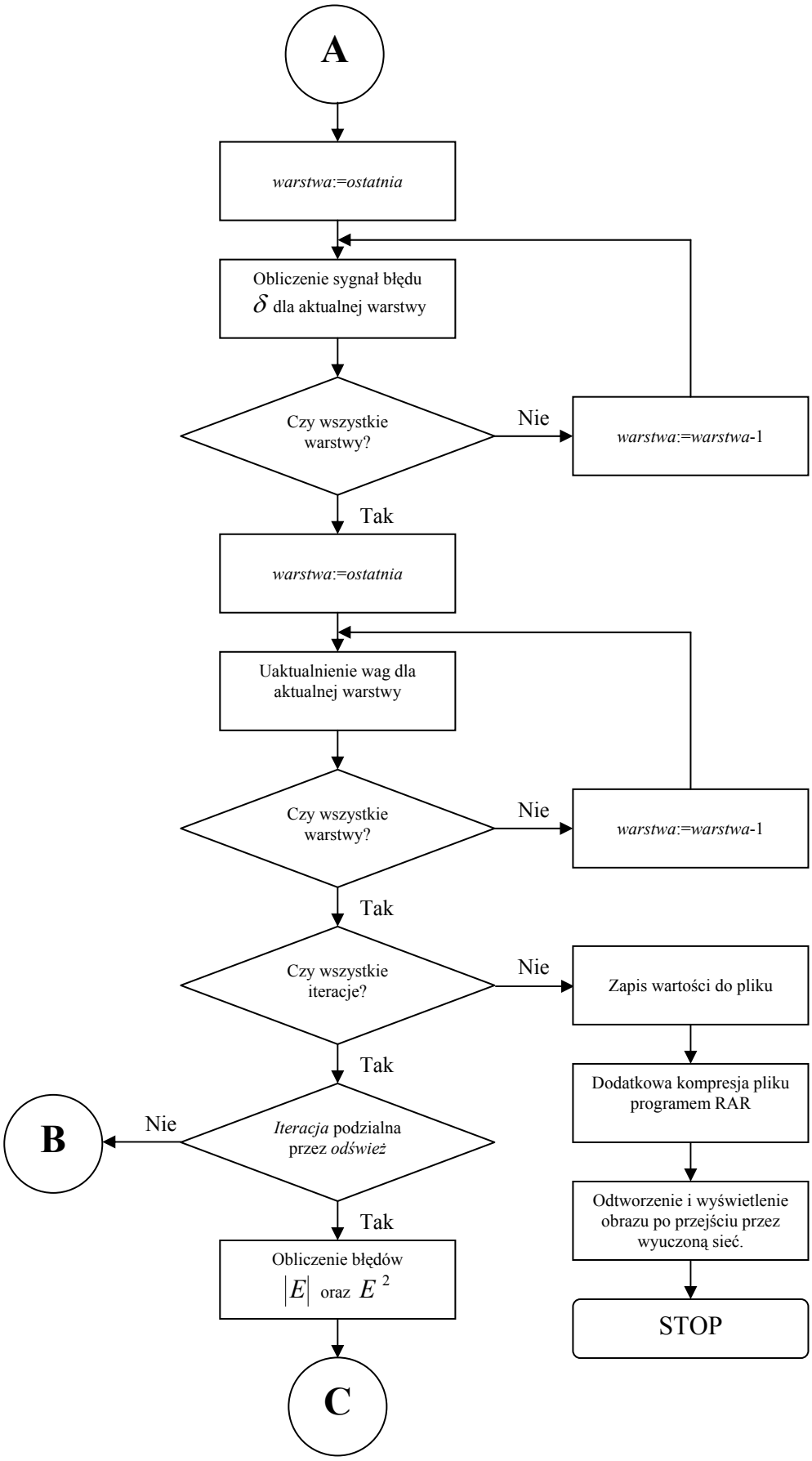
$$f(net) = \frac{2}{1+\exp(-\lambda net)} - 1 \quad \Rightarrow \quad net = -\frac{1}{\lambda}\ln\left(\frac{2}{f(net)+1} - 1\right)$$

Wartości wag zmienione zostały w taki sposób, aby po obliczeniu funkcji *f(net)* otrzymać wartość równą tej, którą otrzymano w wyniku transformacji KLT.

4.4 Algorytm przetwarzania obrazu za pomocą sieci neuronowej

W programie „Kompresja Neuronowa” zastosowano następujący algorytm przetwarzania obrazu przez sieć neuronową.




A
warstwa:=ostatnia
Obliczenie sygnał błędu
δ dla aktualnej warstwy
Czy wszystkie warstwy?
Nie
warstwa:=warstwa-1
Tak
warstwa:=ostatnia
Uaktualnienie wag dla aktualnej warstwy
Czy wszystkie warstwy?
Nie
warstwa:=warstwa-1
Tak
Czy wszystkie iteracje?
Nie
Zapis wartości do pliku
Tak
Dodatkowa kompresja pliku programem RAR
B
Nie
Iteracja podzialna przez odśwież
Tak
Odtworzenie i wyświetlenie obrazu po przejściu przez wyuczoną sieć.
Obliczenie błędów
|E| oraz E 2
STOP
C

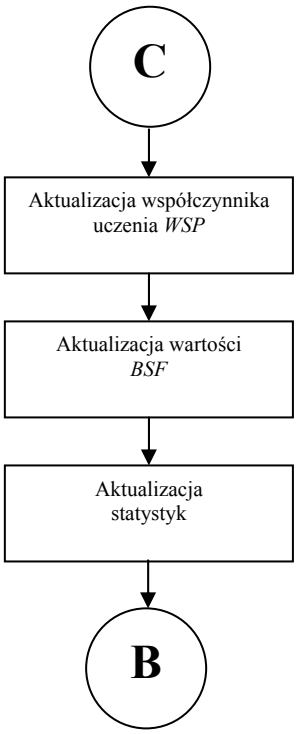


Diagram 3. Algorytm przetwarzania obrazu za pomocą sieci neuronowej.

4.5 Istotniejsze fragmenty kodu programu

Poniżej przedstawione zostaną najistotniejsze fragmenty kodu źródłowego. Dla celów edukacyjnych zostały one celowo uproszczone w taki sposób, aby najlepiej oddać zasadę ich działania.

```
Obliczenie przybliżenia wag
na podstawie transformaty
KLT
```

Jednym z zadań tej pracy było przeprowadzenie testów na temat wpływu początkowego przybliżenia wag sztucznej sieci neuronowej. Zadanie to, było by zagadnieniem prostym, gdyby nie fakt, że zastosowano nieliniową funkcję aktywacji w każdym z neuronów. W takim wypadku zachodzi konieczność zastosowania pewnego przekształcenia, które pozwoli na przybliżone ustalenie wartości początkowych wag pomiędzy warstwą wejściową i środkową, jak również środkową i wyjściową całej sieci. Zadanie to realizuje poniższy kod.

```
////////////////////////////////////////////////////
//Generowanie wag warstw środkowej i wyjściowej
//    WEJ     - długość wektora danych wejściowych
//    SRO     - liczba neuronów warstwy środkowej
//    WYJ     - liczba neuronów warstwy wyjściowej
//    w1[i,j] - wagi pomiędzy warstwą wejściową i środkową
//    w2[i,j] - wagi pomiędzy warstwą środkową i wyjściową
//    WSP1    - współczynnik uczenia
//    B[i,j]  - wektor transformacyjny obliczony z KLT
////////////////////////////////////////////////////
for i:=0 to WEJ do for j:=0 to SRO do
      w1[i,j]:=(-1/WSP1)*ln(-1+2/(B[j,i]+1));
for i:=0 to SRO do for j:=0 to WYJ do
      w2[i,j]:=(-1/WSP1)*ln(-1+2/(B[i,j]+1));
```

Obliczenie *net* i *f(net)* dla neuronów z aktualnej warstwy

Podstawowymi wartościami obliczanymi w procesie przetwarzania danych, jak również uczenia sztucznej sieci neuronowej są wartości pobudzenia łącznego neuronu ***net*** oraz wartość funkcji aktywacji neuronu ***f(net)***. W programie „Kompresja Neuronowa", w funkcji *net()* obliczane jest łączne pobudzenie dla neuronu oznaczonego numerem *ktory* w warstwie *warstwa*. Obliczenia przeprowadzane są w zależności od wybranej struktury sieci dla dwóch lub trzech warstw. W funkcji *aktywacja()* natomiast zaimplementowano cztery możliwości obliczenia wartości funkcji aktywacji neuronu, a mianowicie

- bipolarną ciągłą $$f(net) = \frac{2}{1+\exp(-\lambda net)} - 1$$

- unipolarną ciągłą $$f(net) = \frac{1}{1+\exp(-\lambda net)}$$

- bipolarną dyskretną przyjmującą wartości „-1" lub „1"
- unipolarną dyskretną przyjmującą wartości „0" lub „1".

Dodatkowo poniżej przedstawiono przykład ich wykorzystania do obliczania wartości na wyjściu neuronów poszczególnych warstw sieci.

```
//////////////////////////////////////////////////
// Obliczanie wartości net oraz f(net).
//          warstwa – numer warstwy neuronów
//          ktory   - numer neuronu w warstwie
//          WSP1    - współczynnik w funkcji aktywacji
//////////////////////////////////////////////////
function net(ktory,warstwa:integer):double;
var
   i    : integer;
   suma : double;
begin
   suma:=0;
   case warstwa of
      0: for i:=0 to wejsc do suma:=suma+we[i]*w0[i,ktory];
      1: for i:=0 to WEJ do suma:=suma+n0[i]*w1[i,ktory];
      2: for i:=0 to SRO do suma:=suma+n1[i]*w2[i,ktory];
   end; //case warstwa
   net:=suma;
end; // function net();


function aktywacja(ktory,warstwa:integer):double;
begin
   aktywacja:=0;
   case funkcja_a of
      //bipolarna ciągła
      1: aktywacja:=2/(1+exp(-WSP1*net(ktory,warstwa)))-1;
      //unipolarna ciągła
      2: aktywacja:=1/(1+exp(-WSP1*net(ktory,warstwa)));
      //bipolarna dyskretna
      3: if net(ktory,warstwa)>=1 thenaktywacja:=1
         else aktywacja:=-1;
      //unipolarna dyskretna
      4: if net(ktory,warstwa)>=1 then aktywacja:=1
         else aktywacja:=0;
   end{case};
end; //function aktywacja();
```

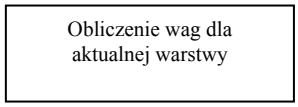
Wywołanie funkcji *net* oraz *f(net)*

```
////////////////////////////////////////////////////
// Obliczenie wartości na wyjściu neuronów poszczególnych warstw
//    n0[i] – neurony warstwy pierwszej (wejściowej)
//    n1[i] – neurony warstwy drugiej (środkowej)
//    n2[i] – neurony warstwy trzeciej (wyjściowej)
////////////////////////////////////////////////////
//wartości na wyjściu pierwszej warstwy
for i:=1 to WEJ do n0[i]:=aktywacja(i,0)
//wartości na wyjściu drugiej warstwy
for i:=1 to SRO do n1[i]:=aktywacja(i,1);
//wartości na wyjściu trzeciej warstwy
for i:=1 to WYJ do n2[i]:=aktywacja(i,2);
```

Obliczenie sygnał błędu δ dla aktualnej warstwy

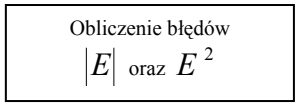
W procesie uczenia sztucznej sieci neuronowej metodą wstecznej propagacji błędu konieczne jest obliczanie w każdej iteracji sygnału błędu dla każdego neuronu w sieci. Zadanie to jest realizowane przez poniższy fragment kodu.

```
//Sygnał błędu dla warstwy wyjściowej
for i:=0 to WYJ do
   swyj[i]:=0.5*(wy[i]-n2[i])*(1-n2[i]*n2[i]);
//Sygnał błędu dla warstwy ukrytej
for i:=0 to SRO do begin
   pom:=0;
   for j:=0 to WYJ do pom:=pom+swyj[j]*w2[i,j];
      ssro[i]:=0.5*(1-n1[i]*n1[i])*pom;
end; //for i
//Sygnał błędu dla warstwy wejściowej
for i:=0 to WEJ do begin
   pom:=0;
   for j:=0 to SRO do pom:=pom+ssro[j]*w1[i,j];
      swej[i]:=0.5*(1-n0[i]*n0[i])*pom;
end; //for i...
```



Kolejnym etapem w procesie uczenia sieci jest modyfikacja wag połączeń pomiędzy neuronami we wszystkich warstwach. Zadanie to jest realizowane podobnie jak w przypadku wyznaczania sygnału błędu, od warstwy ostatniej (wyjściowej) do pierwszej (wejściowej). Współczynnik uczenia oznaczony jest jako *WSP*, natomiast *WSP2* jest współczynnikiem określającym udział czynnika momentum w procesie uczenia. Tablice oznaczone jako *_bak[]* przechowują wartości odpowiednich wag z poprzedniej iteracji

```
//Uaktualnienie wag warstwy wyjściowej
for i:=0 to SRO do for j:=0 to WYJ do begin
   w2[i,j]:=w2[i,j]+WSP*(swyj[j]*n1[i])+WSP2*(w2[i,j]-w2_bak[i,j]);
end; //for i
//Uaktualnienie wag warstwy środkowej
for i:=0 to WEJ do for j:=0 to SRO do begin
   w1[i,j]:=w1[i,j]+WSP*(ssro[j]*n0[i])+WSP2*(w1[i,j]-w1_bak[i,j]);
end; //for i
//Uaktualnienie wag warstwy wejściowej
for i:=0 to wejsc do for j:=0 to WEJ do begin
   pom:=w0[i,j]+WSP*(swej[j]*we[i])+WSP2*(w0[i,j]-w0_bak[i,j]);
end; //for i
```



Podstawową wartością będącą miarą jakości algorytmu jest wartość błędu średniokwadratowego przypadającego na jeden piksel. Do tego celu obliczana jest suma kwadratów różnic pomiędzy obrazem wzorcowym, którego wartości zapisane są w tablicy *obraz[]* oraz obrazem obliczonym, którego wartości pobierane są bezpośrednio z pamięci obrazu w obiekcie *form7.Image1*. Suma ta dzielona jest następnie przez liczbę wszystkich pikseli w obrazie wzorcowym.

Dodatkowo do celów informacyjnych oblicza się sumaryczny błąd bezwzględny.

```
////////////////////////////////////////////////////
//Obliczanie błędu bezwzględnego E oraz średniokwadratowego E2 //dla
całego obrazu.
//Wartość punktu[i,j] - form7.Image1.Canvas.Pixels[i,j] and $ff
//Obraz wzorcowy      - obraz[i,j]
////////////////////////////////////////////////////
E:=0;E2:=0;
for i:=1 to wex do for j:=1 to wey do begin
   x:=(form7.Image1.Canvas.Pixels[i,j] and $ff);
   y:=obraz[i,j];
   E:=E+abs(x-y);
   E2:=E2+(x-y)*(x-y);
end;//for j
E2:=E2/(wex*wey);
```

Aktualizacja współczynnika uczenia *WSP*

W procesie uczenia sieci bardzo istotnym parametrem, który może być zmieniany przez użytkownika programu jest współczynnik uczenia *WSP*. Przyjęcie zbyt dużej wartości tego współczynnika powoduje brak możliwości dostrojenia się sieci, natomiast zbyt mała wartość spowoduje znaczne spowolnienie procesu uczenia. Aby uniknąć obu tych przypadków wprowadzono do programu dynamiczną zmianę *WSP* dokonywaną na podstawie analizy procesu uczenia, a w szczególności zmian wartości błędu średniokwadratowego. Dokładny opis wpływu odpowiedniego doboru *WSP* zostanie przedstawiony w dalszej części pracy.

```
////////////////////////////////////////////////////
//Obliczanie błędu bezwzględnego E oraz średniokwadratowego E2 //dla
całego obrazu.
//Wartość punktu[i,j] - form7.Image1.Canvas.Pixels[i,j] and $ff
////////////////////////////////////////////////////
E_old:=E_akt;
E_akt:=E2;
if (E_akt<=0.9*E_old) and (WSP<0.9) then WSP:=WSP*1.05;
if (E_akt>1.2*E_old) and (WSP>0.001) then WSP:=WSP*0.85;
```

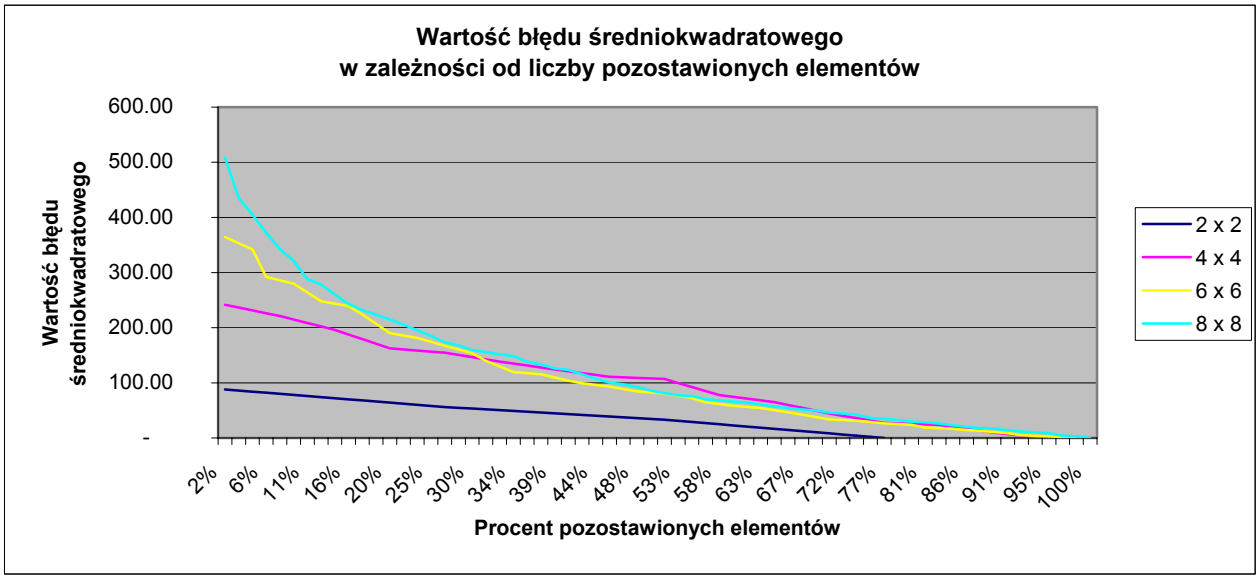
Rozdział 5
Eksperymenty i wnioski

Omówione tu zostaną szczegółowo badania obu metod kompresji obrazów nieruchomych oraz ich wyniki. Kryterium jakości metody będzie minimalizacja unormowanego błędu średniokwadratowego.

5.1 Metoda KLT

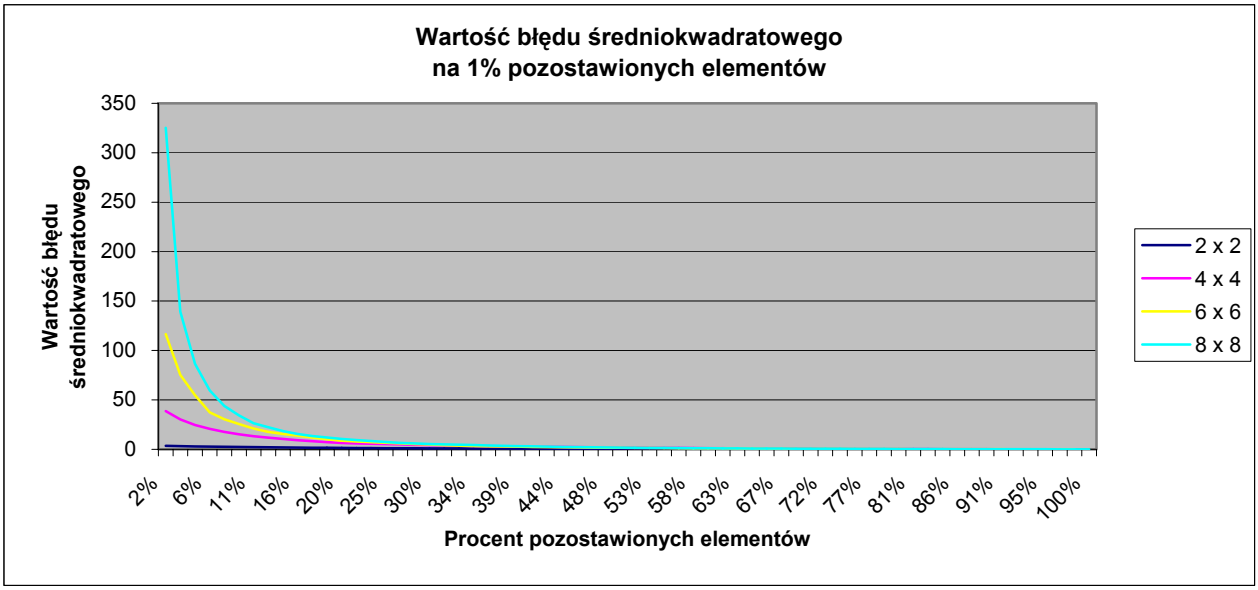
Pierwszą z badanych metod kompresji jest metoda transformacyjna wykorzystująca transformatę KLT. Jest ona najlepsza spośród metod transformacyjnych pod względem minimalizacji błędu średniokwadratowego.

W celu porównania skuteczności i jakości kompresji przeprowadzono badania dla różnych wielkości podobrazów. Dla łatwiejszego zobrazowania wyników na osi *X* wykresu przedstawiono procent pozostawionych elementów przekształcenia. Do analizy użyto popularnego w tego typu testach obrazu „Lena".



Rys.9. Zależność wartości błędu średniokwadratowego od liczby pozostawionych elementów w macierzy transformacyjnej dla różnych rozmiarów podobrazu.

Na rysunku nr 10 natomiast przedstawiono wartość błędu przypadającą na jeden procent pozostawionych po transformacji elementów.



Rys.10. Wartości błędu średniokwadratowego przypadającej na jeden procent pozostawionych elementów w macierzy transformacyjnej.

Dla lepszego zobrazowania jakości metody poniżej przedstawiono przetransformowane obrazy dla podobrazu o rozmiarach 4 x 4 piksele dla różnej liczby pozostawionych elementów.

a) KLT 4x4x1 b) KLT 4x4x2

c) KLT 4x4x4 d) KLT 4x4x8

e) KLT 4x4x12 f) KLT 4x4x16

Rys.11. Jakość odwzorowania obrazu po przetransformowaniu KLT dla różnej liczby pozostawionych elementów transformacyjnych.

Podobnie dla różnych wielkości podobrazu transformata KLT wykazuje różną skuteczność. Poniżej na rysunku 12 przykład zastosowania dla podobrazów 2x2, 4x4, 6x6, 8x8, 16x16. Procedury umieszczone w programie „Kompresja Neuronowa" umożliwiają przetwarzanie podobrazów o maksymalnych rozmiarach 19x19 pikseli, jednak ze względu na bardzo długi czas oczekiwania na wynik, który liczony jest wielu minutach na komputerze klasy P4 z zegarem 1.6GHz, nie ma ona na dzień dzisiejszy zastosowania praktycznego. Jakkolwiek szybki rozwój technik cyfrowych, a w szczególności wzrost szybkości komputerów klasy PC już niedługo umożliwi swobodne korzystanie z tak dużych, a nawet większych podobrazów.

a) KLT 2x2x2

b) KLT 4x4x8

c) KLT 6x6x18

d) KLT 8x8x32

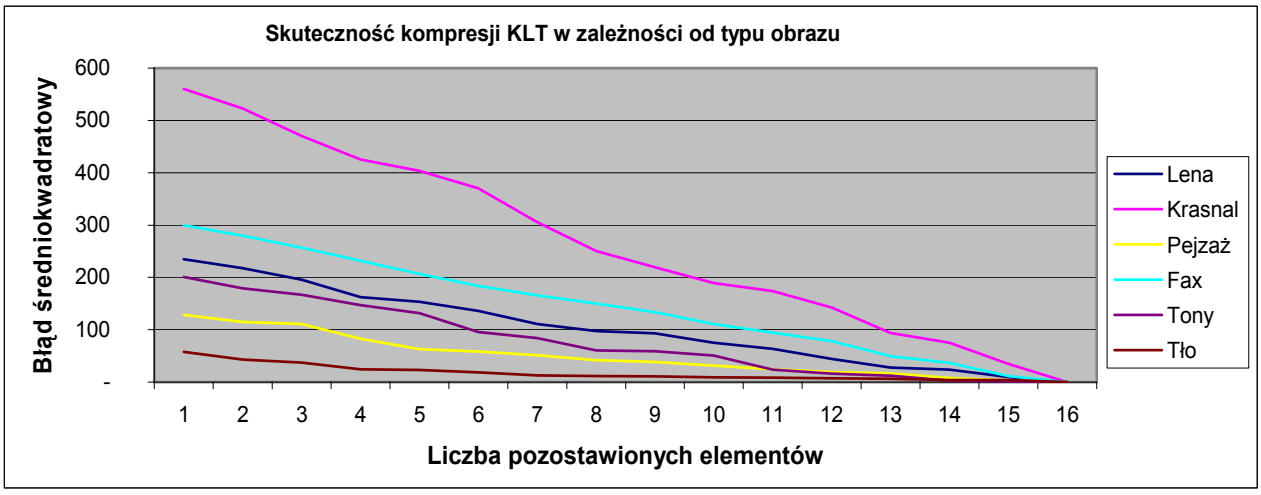
e) KLT 16x16x128

Rys.12. Jakość odwzorowania obrazu po przetransformowaniu KLT dla różnej wielkości podobrazów.

Dodatkowym elementem, mającym wpływ na jakość kompresji obrazu jest jego zawartość. Program zachowuje się różnie dla różnych typów danych, takich jak pejzaż, portret (Lena, Krasnal), tło, przejścia tonalne, fax.

Liczba elementów	Lena	Krasnal	Pejzaż	Faks	Tony	Tło
1	235.12	560.01	128.13	299.99	201.09	57.91
2	217.49	523.03	114.93	279.59	179.20	42.72
3	195.79	470.38	111.26	256.53	167.02	37.39
4	162.64	425.64	82.72	231.92	146.92	24.45
5	153.60	403.72	62.84	206.93	131.53	23.41
6	136.19	370.18	58.11	183.60	95.60	18.38
7	111.30	305.67	51.43	165.25	84.30	12.84
8	97.23	250.05	41.80	149.86	60.63	11.50
9	93.06	219.59	38.74	133.26	59.17	10.64
10	75.30	189.01	31.68	111.11	50.86	9.27
11	63.60	173.71	24.27	94.39	24.00	8.03
12	44.42	142.17	19.47	78.55	16.13	6.83
13	27.35	93.37	16.60	49.45	12.22	6.38
14	23.92	75.09	7.91	36.31	2.48	3.97
15	9.48	34.43	3.52	11.34	1.19	3.56
16	-	-	-	-	-	-

Tabela 6. Tabela zależności wartości błędu średniokwadratowego dla różnych typów obrazów przetworzonych transformatą KLT (Opracowanie własne).



Rys.13. Wartość błędu średniokwadratowego w zależności od typu obrazu.

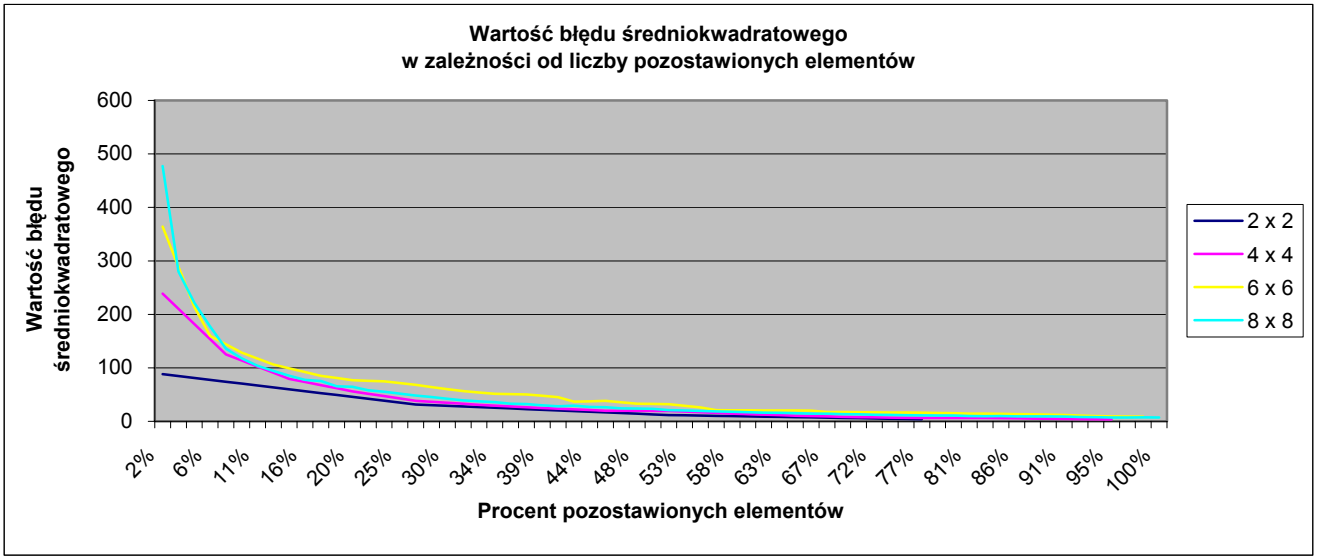
Z powyższego zestawienia i przeprowadzonych testów wynika fakt, że metoda kompresji obrazu transformatą KLT nie nadaje się do wszystkich rodzajów obrazu. Najlepiej zachowuje się ona wobec obrazów, w których dominują obszary mało zróżnicowanego tła („Pejzaż", „Tło"). Przy obrazach o większym zróżnicowaniu, czy też przejściach tonalnych („Tony", „Lena") widać już jego mniejszą

skuteczność. Natomiast z obrazami o znacznym zróżnicowaniu treści występującej w obrazie „Krasnal" opisywana transformata radzi sobie zdecydowanie najsłabiej. Zaskoczeniem jednak jest tutaj bardzo słabe możliwości kompresji obrazów typu „Faks", które przecież nie zawierają wielu szczegółów w porównaniu z „Krasnalem". Wytłumaczeniem tutaj mogą być występujące ostre kontury.

Zasadniczym atutem opisywanej metody jest jej szybkość. Dla podobrazów o rozmiarach do 8x8 pikseli czas przetwarzania obrazu jest podczas kompresji równy najwyżej kilka sekund. Dla dekompresji czas ten jest wręcz niemierzalny dla każdej osiągalnej wielkości podobrazu na przeciętnym komputerze klasy PC dostępnym w dzisiejszych czasach (rok 2003).

5.2 Metoda NN

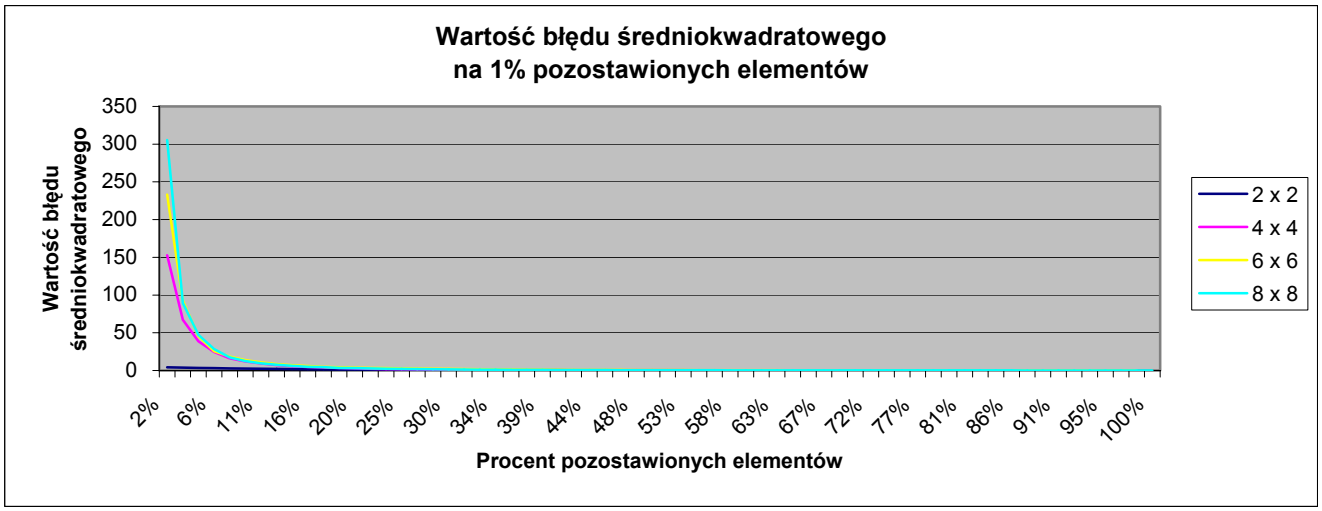
Drugą metodą kompresji badaną w niniejszej pracy jest kompresja za pomocą sieci neuronowej. Podobnie jak dla transformaty KLT, tak i tutaj przeprowadzono badanie jakości kompresji dla różnych wielkości podobrazów wydzielonych z obrazu oryginalnego, jakim jest obraz „Lena".



Rys.14. Zależność wartości błędu średniokwadratowego od liczby neuronów w warstwie środkowej.

Jak widać jakość kompresji, której wyznacznikiem jest błąd średniokwadratowy obrazu przetworzonego przez sieć w stosunku do obrazu oryginalnego jest w bardzo dużym stopniu zależna od liczby neuronów pozostawionych w warstwie środkowej. Wartość akceptowalną dla ludzkiego oka potrafi ona odwzorować już za pomocą

20% neuronów. Jest to wskaźnik bardzo dobry w porównaniu z metodą KLT, od której w tym wypadku potrafi być trzykrotnie lepsza. Również na poniższym wykresie widać wyraźnie, że wartość błędu przypadająca na jeden procent pozostawionych elementów ma charakterystykę znacznie lepszą od KLT.



Rys.15. Wartość błędu średniokwadratowego przypadająca na 1% pozostawionych neuronów w warstwie środkowej.

Podobnie jak dla KLT, tak i tutaj dla lepszego zobrazowania jakości metody poniżej przedstawiono obrazy przetworzone, a następnie odtworzone przez sieć neuronową. Obraz został podzielony na podobrazy o rozmiarach 4 x 4 piksele. Dla przykładu podano obrazy przetworzone przez sieć o różnej liczbie neuronów w warstwie środkowej.

a) NN 4x4x1 b) NN 4x4x2

c) NN 4x4x4 d) NN 4x4x8


e) NN 4x4x12 f) NN 4x4x16


Rys.16. Jakość odwzorowania obrazu przez sieć neuronową dla różnej liczby neuronów w warstwie środkowej.

Podobnie jak transformata KLT tak i sieć neuronowa wykazuje różną skuteczność kompresji obrazu dla różnych wielkości podobrazów. Poniżej dla przykładu przedstawiono skuteczność kompresji dla obrazu Lena podzielonego na podobrazy o rozmiarach 2x2, 4x4, 6x6, 8x8, 16x16 pikseli. Program „Kompresja Neuronowa" umożliwia przetwarzanie obrazów o rozmiarach dochodzących do 48x48 pikseli z możliwością zwiększenia tej wartości, jednak wraz ze zwiększaniem rozmiarów podobrazów zwiększa się również rozmiar sieci przetwarzającej dany podobraz, a tym samym moc komputera potrzebna do jego obsłużenia. Jednocześnie spada również szybkość zbieżności algorytmu i konieczne jest wykonanie większej liczby kroków uczących.

a) NN 2x2x2

b) NN 4x4x8

c) NN 6x6x18

d) NN 8x8x32

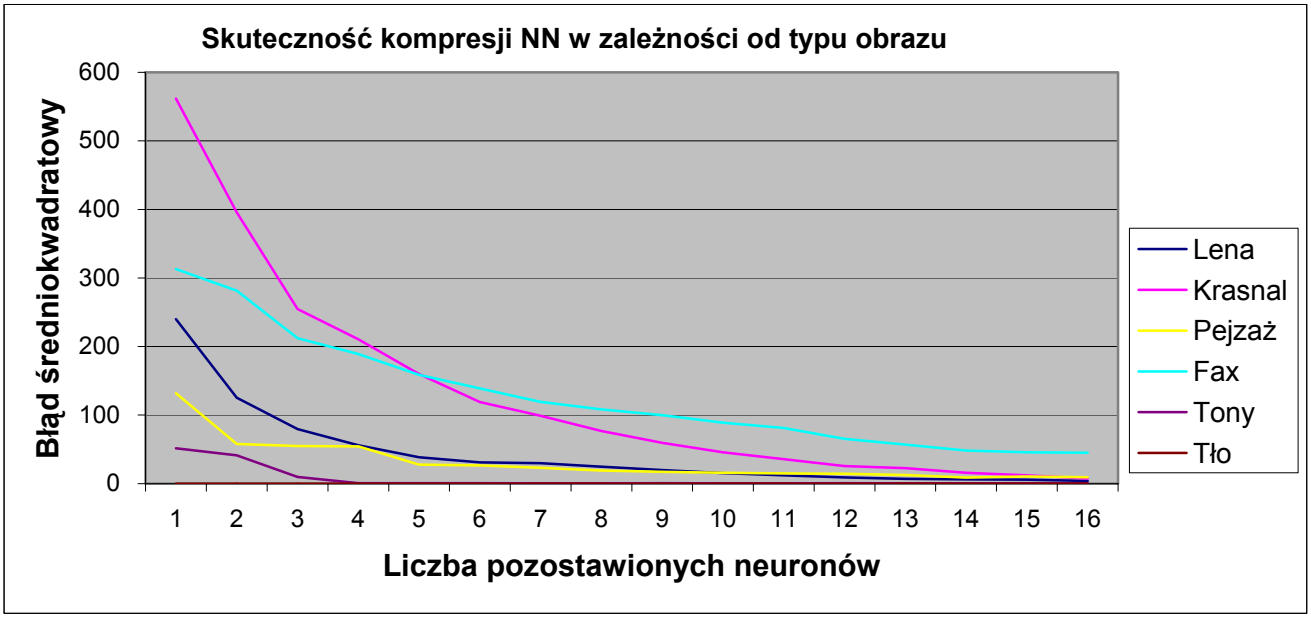
e) NN 16x16x128

Rys.17. Jakość odwzorowania obrazu po przetworzeniu przez sieć neuronową dla różnej wielkości podobrazów.

Jakość kompresji zależy również od rodzaju obrazu, jaki jest przetwarzany. Testy przeprowadzono dla tego samego zestawu obrazów jak dla KLT.

Liczba neuronów	Lena	Krasnal	Pejzaż	Faks	Tony	Tło
1	239.96	561.43	131.79	313.00	51.56	0.00
2	125.23	395.62	57.91	281.39	41.20	0.00
3	79.42	254.67	54.89	212.26	9.73	0.00
4	55.99	210.62	54.31	189.13	0.77	0.00
5	38.42	160.06	27.79	158.83	0.70	0.00
6	30.95	118.89	26.53	139.02	0.64	0.00
7	29.94	98.76	23.19	119.06	0.54	0.00
8	24.78	76.65	19.16	108.30	0.58	0.00
9	19.79	59.37	17.17	99.75	0.70	0.00
10	15.28	45.77	15.77	88.91	0.45	0.00
11	12.16	35.72	14.82	81.08	0.53	0.00
12	8.98	25.57	14.40	65.28	0.61	0.00
13	6.96	22.66	12.51	56.93	0.58	0.00
14	6.19	15.77	8.95	48.19	0.52	0.00
15	5.92	12.23	10.41	45.63	0.63	0.00
16	4.01	7.47	9.30	45.05	0.61	0.00

Tabela 7. Tabela zależności wartości błędu średniokwadratowego dla różnych typów obrazów przetworzonych siecią neuronową (Opracowanie własne).



Rys.18. Wartość błędu średniokwadratowego w zależności od typu obrazu przetworzonego siecią neuronową.

Sieć neuronowa podobnie jak transformata KLT różnie zachowuje się w przypadku różnych rodzajów obrazu. Dla obrazów takich jak „Krasnal” czy „Faks”, w których kontury są silnie zaznaczone skuteczność kompresji jest

najmniejsza. Dla średnio skomplikowanych, w których kontury, faktura i tło występują w mniej więcej równych proporcjach, np. dla obrazu „Lena" stopień kompresji i jakość odtworzonego obrazu są na akceptowalnym poziomie. Przy przetwarzaniu obrazu „Pejzaż" widać już siłę kompresji w porównaniu z KLT, w stosunku, do którego w szerokim spektrum pozostawionych elementów jest nawet trzykrotnie lepsza. Prawdziwą siłę jednak metoda ta pokazuje w przypadku obrazów jednorodnych, lub prawie jednorodnych, takich jak „Tony" czy „Tło", gdzie jest wprost bezkonkurencyjna i deklasuje KLT. Dodatkowym atutem tej metody może być jej szybkość w porównaniu do metod transformacyjnych. Bardzo długi proces uczenia sieci, który na współczesnym (rok 2003) komputerze klasy PC zajmuje nawet kilka godzin, jest dokonywany jednokrotnie, a później za jego pomocą w czasie poniżej sekundy można kompresować inne obrazy podobnej klasy. Proces dekompresji również zajmuje czas poniżej jednej sekundy.

Do poniższych testów wykorzystano obrazy testowe przedstawione na rys. 19.

19a – obraz „Lena" bardzo często wykorzystywany przy porównywaniu metod kompresji (źródło z internetu „The rest of Lenna story") [20],

19b – Weronika Puchalska w wieku 3 miesięcy (źródło własne),

19c – zdjęcie ze spływu kajakowego wykonane w 2003 roku (źródło własne),

19d – strona 30 tej pracy skanowana z rozdzielczością 150 dpi,

19e – obraz „Tony" wygenerowany na potrzeby testów, wartości pikseli w każdym wierszu są jednakowe, a w kolejnych wierszach zwiększają się o 1,

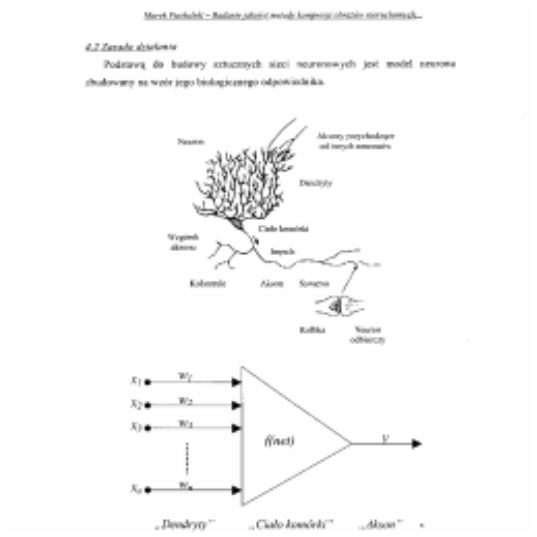
19f – obraz „Tło" wygenerowany na potrzeby testów, wartość każdego piksela ustawiona została na 128.

a) „Lena"

b) „Krasnal"

c) „Pejzaż”

d) „Faks”



c) „Tony”

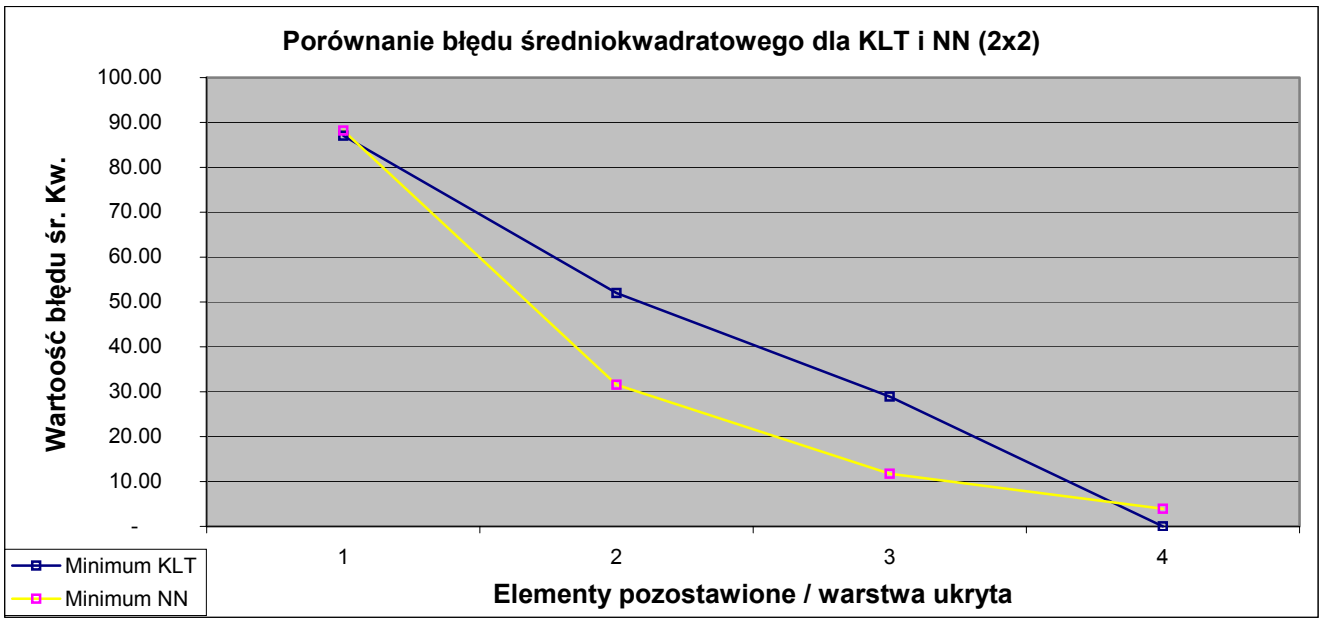
d) „Tło”

Rys.19. Obrazy wykorzystane do badania wpływu rodzaju obrazu na jakość przetwarzania zarówno transformatą KLT, jak i siecią neuronową.

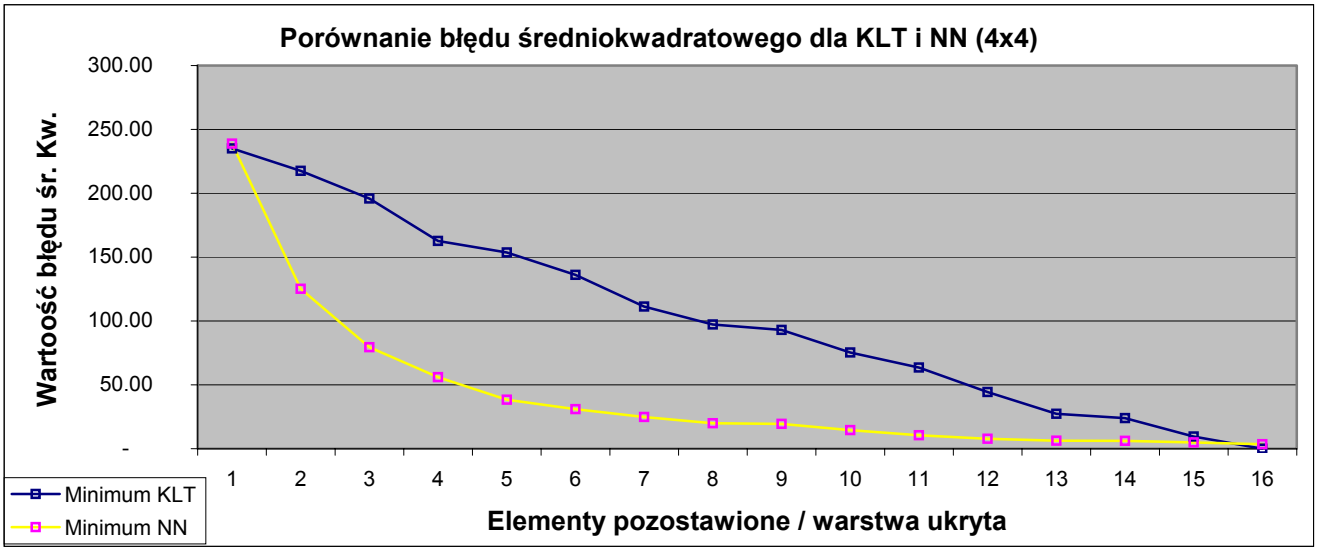
5.3 Porównanie metody KLT z NN

W tej pracy opisane są i testowane dwie metody kompresji obrazów. Pierwsza z nich, to transformata Karhunena-Loeve opierająca się o przekształcenia macierzowe, druga natomiast polega na przekształceniu obrazu przez sieć neuronową. Obie opierają się na podobnych założeniach i to do tego stopnia, że biorąc pod uwagę sieć dwuwarstwową z jednostkowymi funkcjami aktywacji w neuronach, można przyjąć, że wagi pomiędzy wejściem sieci, a warstwą środkową odpowiadają macierzy wektorów własnych macierzy kowariancji będącej podstawą

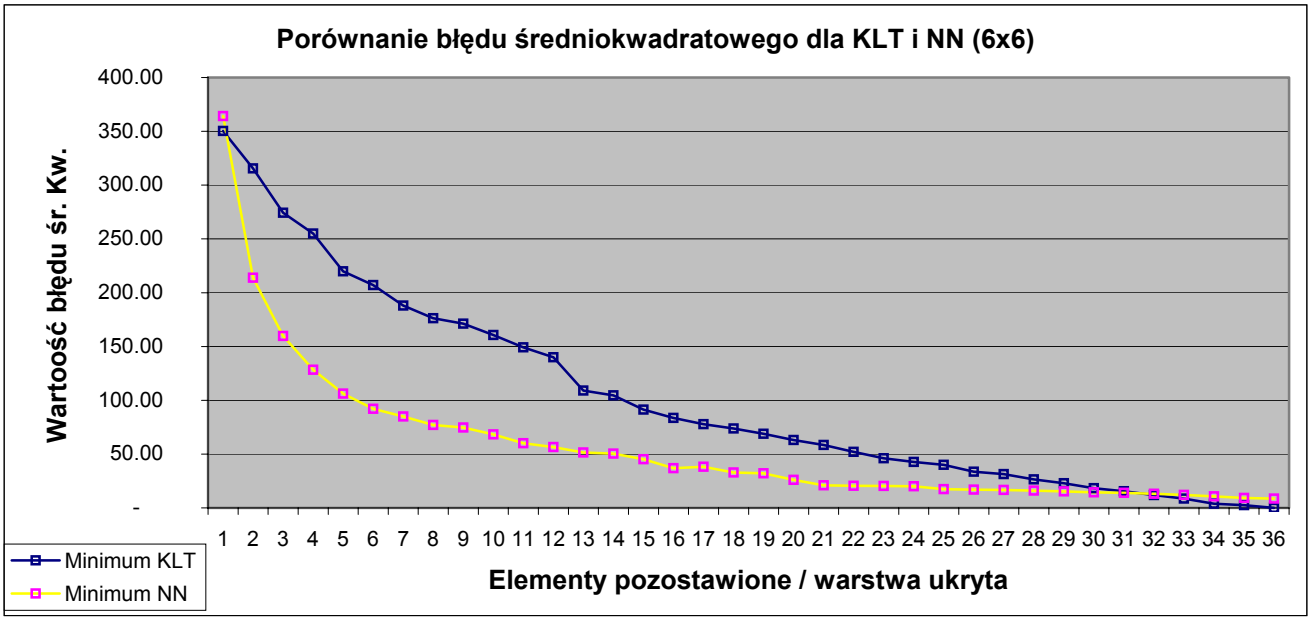
transformaty KLT, a wartości pomiędzy warstwą środkową i wyjściową przyjmują dokładnie wartości powyższej macierzy transponowanej. Jednak ze względu budowę każdego neuronu użytego do budowy sieci w programie „Kompresja Neuronowa", w którym zaimplementowaną nieliniową funkcję aktywacji podejście takie nie jest możliwe i wartości macierzy transformacyjnej nie odpowiadają już wartościom wag w sieci. W takim wypadku można się pokusić co najwyżej o pewne przybliżenie wartości początkowych, od których sieć zaczyna „naukę". Zastosowanie nieliniowej funkcji aktywacji powoduje znaczny wzrost efektywności algorytmu pod względem zarówno minimalizacji błędu średniokwadratowego odtworzonego obrazu, jak również stopnia kompresji. Dlatego też metoda kompresji obrazów nieruchomych za pomocą sieci neuronowej jest w wielu przypadkach znacznie skuteczniejsza od transformaty KLT, co pokazują wyraźnie poniższe wykresy dla obrazu „Lena".



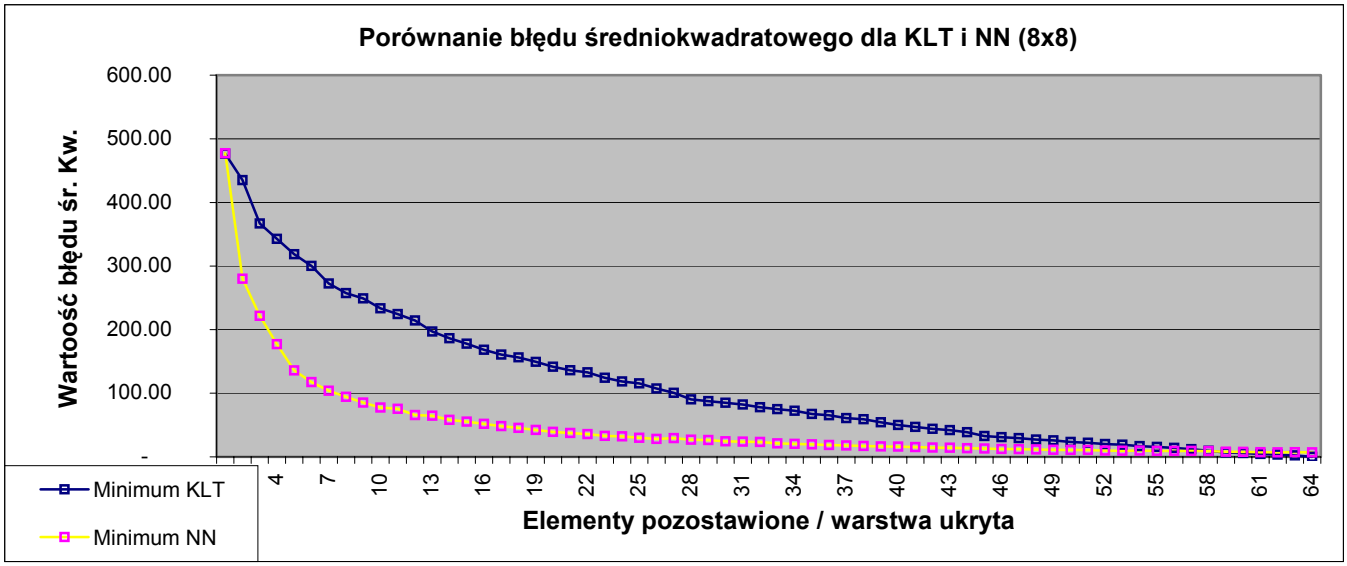
Rys.20. Porównanie błędu średniokwadratowego po przetworzeniu przez sieć neuronową oraz KLT dla podobrazu o rozmiarach 2x2 piksele.



Rys.21. Porównanie błędu średniokwadratowego po przetworzeniu przez sieć neuronową oraz KLT dla podobrazu o rozmiarach 4x4 piksele.



Rys.22. Porównanie błędu średniokwadratowego po przetworzeniu przez sieć neuronową oraz KLT dla podobrazu o rozmiarach 6x6 pikseli.



Rys.23. Porównanie błędu średniokwadratowego po przetworzeniu przez sieć neuronową oraz KLT dla podobrazu o rozmiarach 8x8 pikseli.

Na wszystkich powyższych rysunkach widać zdecydowaną przewagę sieci neuronowej poza sytuacją, kiedy pozostawiamy jeden element/neuron, lub też, gdy pozostawiamy wszystkie elementy/neurony. W pierwszym przypadku transformata KLT liczy po prostu średnią dla każdego wydzielonego podobrazu i jest w tym zwykle lepsza od sieci NN. Dla wszystkich elementów przekształcenie KLT jest bezstratne i odtwarza zawsze obraz wzorcowy w przeciwieństwie do sieci, gdzie standardowo wprowadzane są pewne szumy. Natomiast praktycznie w całej pozostałej przestrzeni liczby elementów pozostawionych widzimy, że sieć NN znacznie szybciej dąży do minimalizacji błędu średniokwadratowego. Wynikają z tego następujące dwa fakty:

- Przy takiej samej liczbie neuronów w warstwie środkowej, co elementów macierzy transformacyjnej KLT, błąd średniokwadratowy jest mniejszy dla sieci NN, a tym samym jakość odtworzonego obrazu jest lepsza.

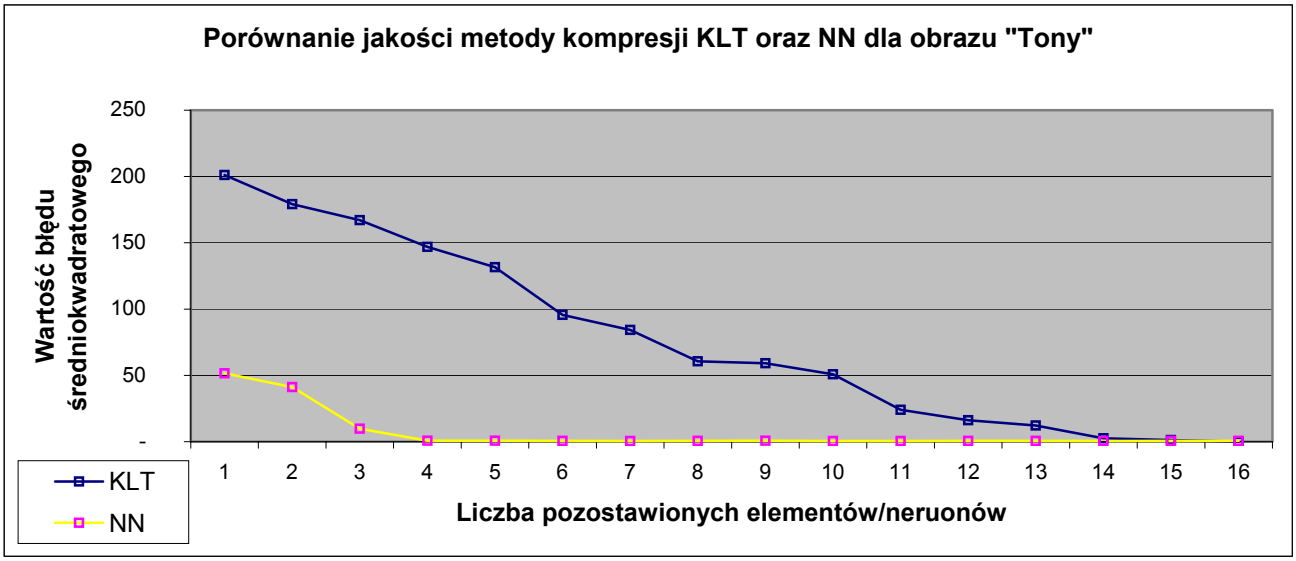
- Przy takim samym błędzie średniokwadratowym odtworzonego obrazu potrzeba znacznie mniej neuronów od elementów transformaty KLT, a tym samym stopień kompresji jest lepszy dla sieci NN.

Najlepiej przewagę sieci NN widać na przykładzie obrazów „Tło” i „Tony”. W pierwszym przypadku sieć natychmiast rozpoznaje, że jest to obraz jednorodny i za każdym razem błąd średniokwadratowy jest równy zero. W drugim natomiast widać, że sieć rozpoznaje zależność pomiędzy poszczególnymi podobrazami

i wykorzystując to minimalizuje błąd prawie do zera. Przypadek ten obrazuje poniższa tabela wraz z wykresem.

Liczba neuronów / elementów	KLT	NN
1	201.09	51.56
2	179.20	41.20
3	167.02	9.73
4	146.92	0.77
5	131.53	0.70
6	95.60	0.64
7	84.30	0.54
8	60.63	0.58
9	59.17	0.70
10	50.86	0.45
11	24.00	0.53
12	16.13	0.61
13	12.22	0.58
14	2.48	0.52
15	1.19	0.63
16	-	0.61

Tabela 8. Porównanie wartości błędu średniokwadratowego dla transformaty KLT oraz sieci neuronowej przy przetwarzaniu obrazu „Tony". (Opracowanie własne).



Rys 24. Porównanie wartości błędu średniokwadratowego dla KLT oraz sieci neuronowej przy przetwarzaniu obrazu „Tony". (Opracowanie własne).

Widać z powyższego przykładu, w szczególnych typach obrazów zawierających przejścia tonalne, sieć neuronowa deklasuje transformatę KLT oferując nawet

kilkaset razy lepszą jakość odtworzonego obrazu przy 25% pozostawionych neuronach warstwy środkowej.

Ponadto porównując czas przetwarzania jednej i drugiej metody można przypuszczać na początku, że KLT jest znacznie szybsze od sieci NN, ponieważ liczy praktycznie od razu i w ciągu kilku, kilkunastu sekund podaje wynik, podczas gdy sieć zaczyna naukę i dostraja się w trakcie sesji uczenia, która trwa od kilku sekund do wielu godzin, w zależności od ustawionych parametrów. Jednak po nauczeniu taka sieć może być wykorzystywana ponownie do kompresji tego samego typu obrazu. W takim przypadku to ona właśnie wygrywa w wyścigu, ponieważ gotowy obraz podaje w ciągu ułamków sekund ponownie deklasując KLT.

Jak widać z powyższych przykładów, warto poświęcić więcej czasu na zrozumienie zasad jakie rządzą sieciami neuronowymi aby skutecznie wykorzystać drzemiący w nich potencjał i o wiele skuteczniej i szybciej robić to, co do tej pory było domeną metod transformacyjnych, a w szczególności najlepszej z nich jaką jest transformata Karhunena – Loeve (KLT).

5.4 Dodatkowe badania nad zachowaniem się NN

Sieć neuronowa to bardzo czułe narzędzie i wynik jej pracy zależny jest od wielu czynników i parametrów, jakimi można ją opisać. W wielu wypadkach od tych ustawień zależy wręcz, czy dana sieć w ogóle będzie spełniała swoje zadanie. Do podstawowych parametrów opisujących sieć i proces jej uczenia należą:

- Struktura sieci
  - architektura sieci
  - liczba warstw w wielowarstwowej sieci neuronowej
  - liczba neuronów w każdej z warstw
  - liczba połączeń pomiędzy neuronami oraz ich wagi
  - typ funkcji aktywacji neuronów
- Proces uczenia
  - współczynnik uczenia w funkcji aktywacji
  - współczynnik uczenia w aktualizacji wag
  - liczba powtórzeń sesji uczących
  - typ danych wejściowych
  - algorytm poszukiwania rozwiązania (optymalizacje)
  - wartość początkowa wag

Poniżej przedstawiony zostanie wpływ poszczególnych parametrów na zachowanie sieci i procesu jej uczenia. Do celów badawczych przyjęty zostanie jeden z wielu modeli, a mianowicie trójwarstwowa sieć neuronowa, bez sprzężeń zwrotnych, uczona za pomocą algorytmu propagacji wstecznej błędu. Warstwa wejściowa i wyjściowa będą składały się z 16 neuronów połączonych ze zmienną liczbą neuronów warstwy środkowej za pomocą wag korygowanych w procesie uczenia „z nauczycielem". Sieć uczona będzie odwzorowania danych wejściowych w wyjściowe z możliwie najmniejszym błędem średniokwadratowym. Zbiorem danych wejściowych będzie obraz „Lena" podzielony na podobrazy o rozmiarach 4x4 piksele.

- *Porównanie zależności błędu średniokwadratowego od liczby neuronów w warstwie środkowej*.

Stopień kompresji i jakość odtworzonego obrazu jest ściśle zależna od liczby neuronów w warstwie środkowej sieci. Jest ona regulowana w zależności od wyniku jaki chcemy uzyskać. Zwiększając ten parametr uzyskujemy coraz lepszą jakość odtworzonego obrazu wyrażoną minimalnym błędem średniokwadratowym. Maksymalna liczba neuronów w warstwie środkowej jest równa liczbie neuronów w pozostałych warstwach. Natomiast minimalna wartość tego parametru to jeden. Stopień kompresji jest tutaj wyrażony liczbą neuronów w warstwie środkowej S, do liczby neuronów w warstwie wejściowej N i oznacza jaki procent rozmiaru obrazu wzorcowego zajmuje obraz po kompresji.

$$Kompr. = \frac{S}{N}$$

Dla przykładu, jeśli mamy podobraz o rozmiarach 4x4 piksele i pozostawimy w warstwie środkowej 4 neurony, to współczynnik kompresji będzie równy 25%, a to oznacza, że zamiast 100kB na dysku twardym, plik obrazu zajmie teoretycznie tylko 25kB. Oczywiście zmniejszając liczbę elementów warstwy środkowej pogarsza się jakość odtworzonego obrazu, co widać wyraźnie na Rys. 14 ze strony 56 tej pracy. Rozsądny kompromis pomiędzy jakością, a stopniem kompresji osiąga

się pozostawiając w warstwie środkowej już od 10% - 20% neuronów w stosunku do ich liczby w warstwie wejściowej. Najlepiej jednak pozostawiać ok. 25% neuronów, co daje obraz „miły" dla oka, bez widocznych znacznych zniekształceń, natomiast pozostawienie połowy neuronów daje w efekcie bardzo dobry obraz bez widocznych zniekształceń, co prezentują przykłady dla różnych rozmiarów podobrazów przedstawione na rys.17. Dla przykładu poniżej przedstawiono dwa obrazy „Lena" przetworzone przez sieć, w której pozostawiono 12.5% (64-8-64) oraz 25% neuronów (64-16-64).

Rys 25. Obraz „Lena" (podobraz 8x8) przetworzony przez sieć 64-8-64 (po lewej) oraz 64-16-64 (po prawej).

- *Wpływ zastosowania klasyfikacji podobrazów na stopień i jakość kompresji.*

W większości przypadków naturalne obrazy, jakie poddawane są kompresji zbudowane są różnych klas elementów. Pierwsza z nich, to kontury zaznaczające obiekty i przenoszące najwięcej informacji (przede wszystkim o kształtach tych obiektów), a tym samym wymagające najlepszego odwzorowania. Druga to faktura nadająca konturom treść (rodzaj materiału, z jakiego jest zrobiony obiekt). Ostatnią z nich jest tło wypełniające zwykle większość całego obrazu, a będące jednocześnie najmniej zróżnicowanym obszarem w całym obrazie i tym samym przenoszące najmniej informacji. Korzystając z powyższej wiedzy można zróżnicować liczbę neuronów w warstwie środkowej i przyjąć, że dla konturów będzie ich więcej, dla faktury średnio, a dla tła najmniej. Postępując w ten sposób zachowujemy najlepiej

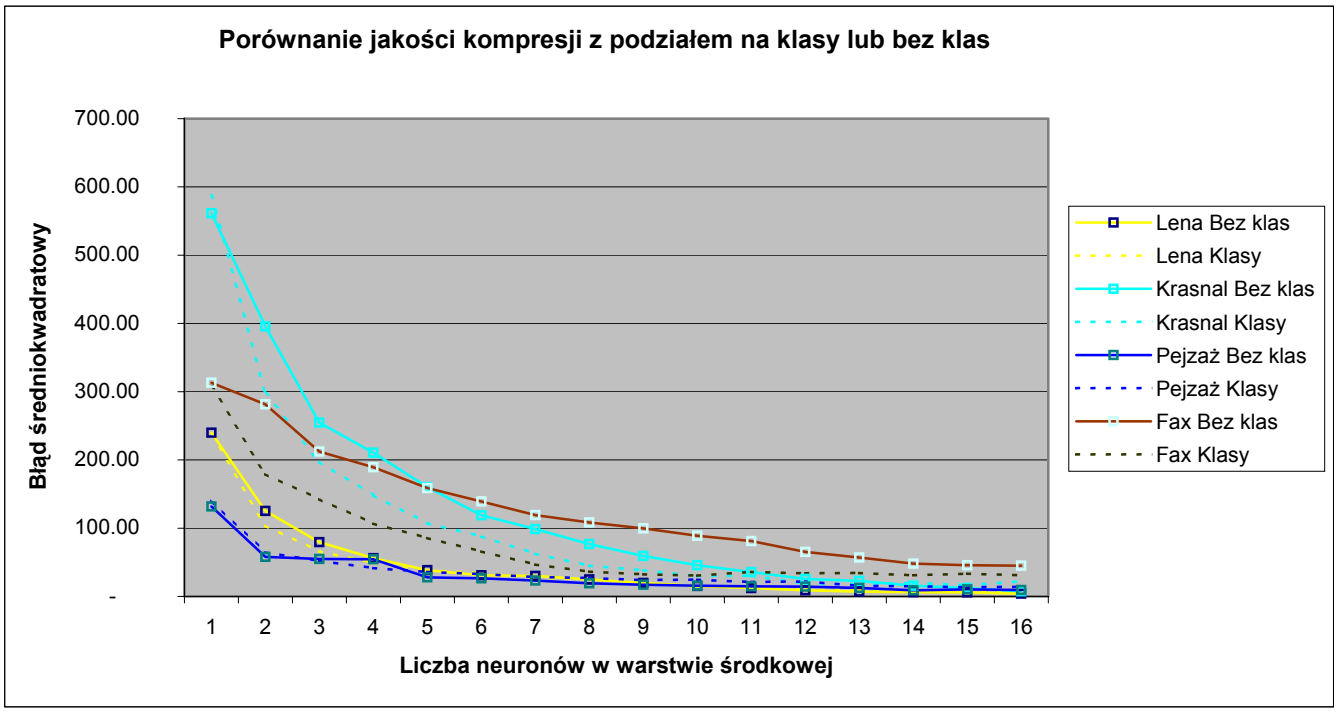
podobrazy najistotniejsze, a najsłabiej te, które są najmniej istotne. Klasyfikację podobrazów przeprowadza się w następujący sposób. Wyszukiwana jest największa i najmniejsza wartość jasności każdego piksela w podobrazie, a następnie na podstawie różnicy wartości pomiędzy nimi dokonywany jest przydział do odpowiedniej klasy. Oczywiście dla podobrazów o różnych rozmiarach należy ustawić różne wartości progowe. Dodatkowo do celów testowych zaimplementowano w programie możliwość podziału na różne klasy podobrazów za pomocą sieci neuronowej. Jakkolwiek jest to ciekawe zadanie dla sieci neuronowej, to jednak selekcja czysto numeryczna jest zdecydowanie prostsza i szybsza. Sieci neuronowe natomiast dużo lepiej nadają się do rozpoznawania i klasyfikacji konkretnych obiektów na obrazach, z którymi powyższa metoda nie daje sobie rady, tzn. np. „czy obiekt na obrazie jest jabłkiem, czy gruszką?". Na rys.26 przedstawiono porównanie kompresji obrazu, w której zastosowano podział na klasy podobrazów, z kompresją bez takiego podziału. Natomiast na rysunku 54 pokazano przykładowy podział na podobrazy.

Liczba neuronów	Lena			Krasnal		
	Bez klas	Neurony	Klasy	Bez klas	Neurony	Klasy
1	239.96	1-1-1	236.96	561.43	1-1-1	87.82
2	125.23	1-2-3	103.69	395.62	1-2-3	297.78
3	79.42	1-3-5	65.96	254.67	1-3-5	197.41
4	55.99	2-4-6	49.22	210.62	1-4-7	148.11
5	38.42	3-5-7	38.47	160.06	1-5-9	107.15
6	30.95	3-6-9	34.02	118.89	2-6-9	87.97
7	29.94	3-7-11	27.34	98.76	3-7-11	62.71
8	24.78	6-8-10	25.18	76.65	3-8-13	45.12
9	19.79	6-9-12	19.91	59.37	2-9-16	38.17
10	15.28	5-10-15	14.40	45.77	4-10-16	29.70
11	12.16	8-11-14	13.32	35.72	7-11-15	30.45
12	8.98	9-12-15	9.76	25.57	8-12-16	18.45
13	6.96	10-13-16	9.44	22.66	9-13-16	17.93
14	6.19	12-14-16	7.87	15.77	13-14-15	20.03
15	5.92	14-15-16	8.61	12.23	14-15-16	16.44
16	4.01	16-16-16	6.77	7.47	16-16-16	22.13

Tabela 9. Porównanie wyników uzyskanych kompresją z podziałem na klasy podobrazów, oraz bez takiego podziału dla obrazów „Lena" i „Krasnal".

Liczba neuronów	Pejzaż			Faks		
	Bez klas	Neurony	Klasy	Bez klas	Neurony	Klasy
1	131.79	1-1-1	140.42	313.00	1-1-1	307.82
2	57.91	1-2-3	64.59	281.39	1-2-3	179.24
3	54.89	2-3-4	52.12	212.26	1-3-5	142.58
4	54.31	2-4-6	41.61	189.13	1-4-7	106.63
5	27.79	2-5-8	35.78	158.83	1-5-9	85.71
6	26.53	6-6-6	32.81	139.02	1-6-11	65.96
7	23.19	4-7-10	28.45	119.06	1-7-13	46.56
8	19.16	7-8-9	26.98	108.30	2-8-14	36.43
9	17.17	8-9-10	23.97	99.75	2-9-16	32.43
10	15.77	10-10-10	24.32	88.91	4-10-16	30.40
11	14.82	10-11-12	21.18	81.08	7-11-15	35.92
12	14.40	11-12-13	21.92	65.28	8-12-16	33.74
13	12.51	13-13-13	15.40	56.93	11-13-15	34.14
14	8.95	14-14-14	14.87	48.19	12-14-16	30.98
15	10.41	14-15-16	13.47	45.63	14-15-16	33.41
16	9.30	16-16-16	14.00	45.05	16-16-16	30.99

Tabela 10. Porównanie wyników uzyskanych kompresją z podziałem na klasy podobrazów, oraz bez takiego podziału dla obrazów „Pejzaż" i „Faks".



Rys 26. Porównanie skuteczności kompresji obrazu z klasyfikacją podobrazów na „tło", „fakturę" i „kontury" oraz bez niej.

Element klasyfikacji na klasy podobrazów można w przyszłości rozwinąć o możliwość analizy obrazów kolorowych i ewentualnej dodatkowej analizy zależności pomiędzy kanałami kolorów.

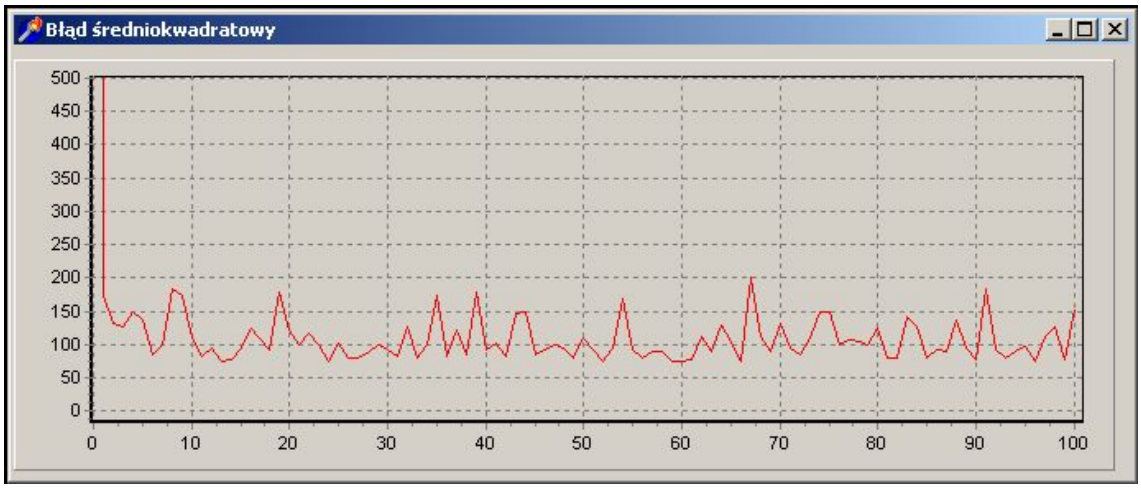
- *Zależność współczynnika kompresji, jakości obrazu i szybkości treningu sieci od współczynnika uczenia*.

Zgodnie ze wzorem na przyrost wartości wag

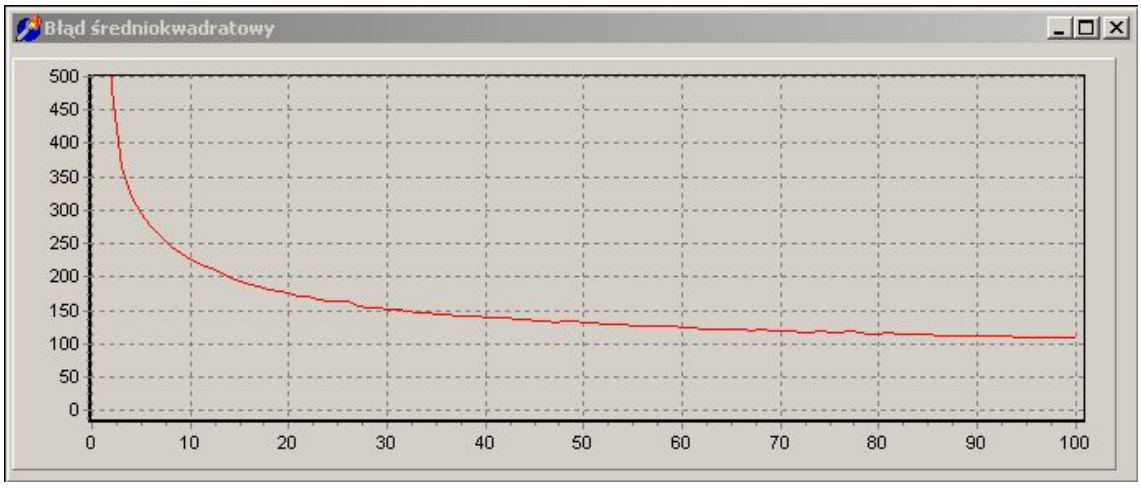
$$\Delta w_i = \beta \cdot f(w_i^T x) \cdot x$$

definiujemy zmienną β jako współczynnik uczenia sieci. W zależności od jego wartości wagi sieci zmieniane są mniej lub bardziej dynamicznie. O ile w początkowej fazie treningu warto ustawić go w zakresie wartości *dużych*, aby początkowe ustawienie wag na wartości losowe było szybko zbieżne w kierunku minimum globalnego, o tyle w końcowej fazie będzie ono przeszkadzało w precyzyjnym dostrojeniu sieci i wtedy warto ustawiać go na wartości *małe*. Wartości *duże* i *małe* są pojęciami względnymi i mocno zależą od struktury sieci i jej rozmiarów. Przyjmuje się, iż optymalnym ustawieniem współczynnika uczenia jest odwrotność liczby neuronów w danej warstwie. Dla przykładu, jeśli w warstwie środkowej jest 5 neuronów, to współczynnik ten powinien przyjmować wartości w okolicy 0.2. Warto przy tym ustawiać go osobno dla każdej warstwy sieci. Jeśli jednak stosuje się dynamiczne dostrojenie, to na początku warto ustawić wartości wyższe dwu-, trzy-, a czasami nawet pięciokrotnie. Takie podejście powoduje, że w początkowej fazie strojenia sieć bardzo szybko zmierza do minimum globalnego omijając po drodze minima lokalne, a w jego okolicy, gdy współczynnik zaczyna dynamicznie maleć, zwalnia i precyzyjnie dostraja się do wartości optymalnej. Na poniższych rysunkach przedstawiono zależność zmian błędu średniokwadratowego w zależności od wielkości współczynnika uczenia dla treningu sieci na obrazie „Lena” podzielonym na podobrazy 4x4 oraz 100 000 kroków uczących. Dynamiczna zmiana jego wartości opiera się na następującej zasadzie:

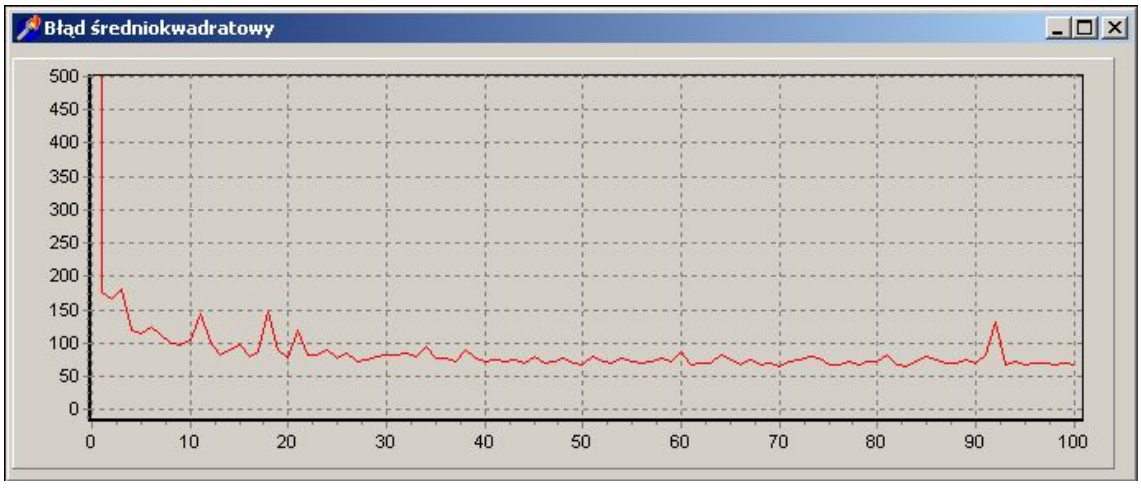
- jest zmniejszany, gdy wzrost błędu średniokwadratowego w stosunku do wartości poprzedniej jest większy niż 20%,
- jest zwiększany, gdy błąd średniokwadratowy zmniejsza swoją wartość w stosunku do wartości poprzedniej o 10%.



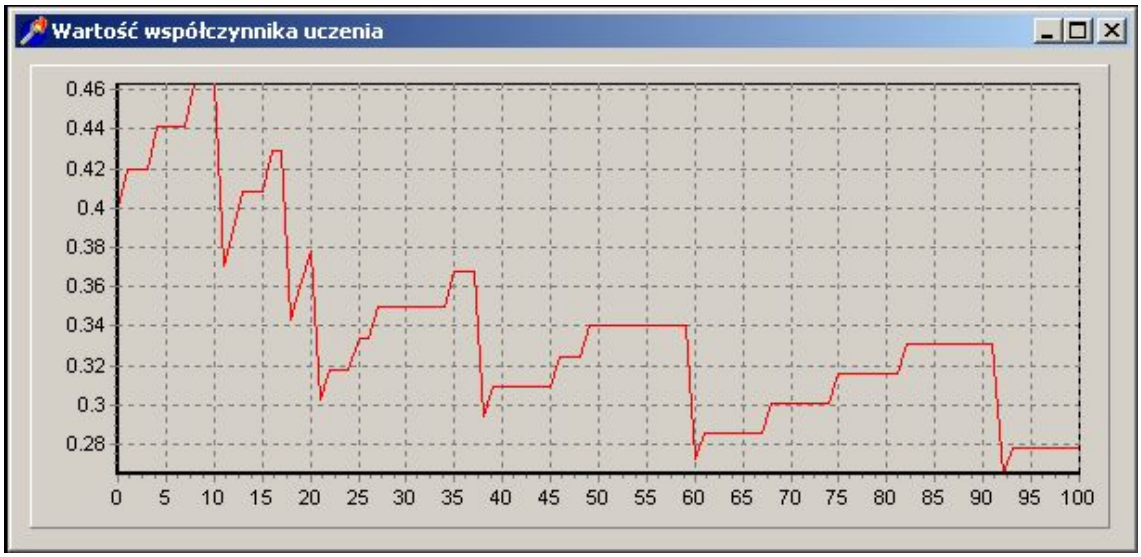
Rys 27. Współczynnik uczenia zbyt duży $\beta = 0.9$ - brak dostrojenia sieci.



Rys 28. Współczynnik uczenia zbyt mały $\beta = 0.01$ - bardzo wolna zbieżność.



Rys 29. Współczynnik uczenia dobierany dynamicznie $\beta = 0.5$ - bardzo szybka zbieżność na początku i precyzyjne dostrojenie pod koniec procesu.



Rys 30. Wykres zmian współczynnika uczenia dobierany dynamicznie. Wartość początkowa $\beta = 0.5$.

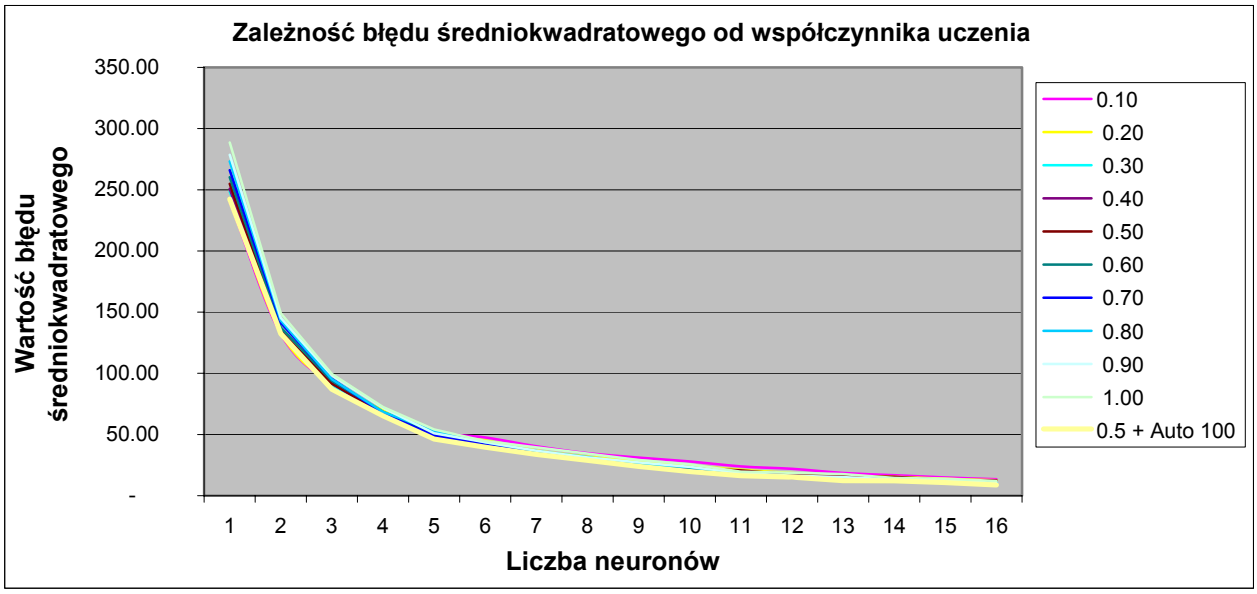
Odpowiedni dobór współczynnika uczenia ma również decydujący wpływ na końcowy wynik całego procesu uczenia i przetwarzania obrazu. Dla zobrazowania tego czynnika poniżej przedstawiono wykres zależności osiąganego wyniku w postaci wartości błędu średniokwadratowego dla współczynnika z zakresu 0.1 - 1.0, a także dla dynamicznie dobieranego z wartością początkową równą 0.5. Pozostałe parametry ustawione identycznie jak dla rys 27-30.

Liczba neuronów	0.10	0.20	0.30	0.40	0.50
1	240.27	243.26	248.61	250.90	254.88
2	132.24	133.34	137.12	132.68	134.52
3	91.65	92.00	91.43	92.01	91.53
4	69.57	69.93	66.50	65.79	67.86
5	52.49	49.77	48.92	50.38	47.95
6	47.61	45.11	41.83	42.64	41.09
7	40.38	36.95	34.33	34.83	34.84
8	34.83	31.96	29.11	29.74	29.33
9	31.15	27.82	24.43	23.89	25.96
10	28.02	24.75	21.79	20.84	22.71
11	23.89	21.22	19.19	17.16	20.72
12	21.94	18.52	16.93	15.05	17.25
13	18.62	16.39	14.91	13.76	16.04
14	16.78	15.49	13.78	11.72	15.43
15	14.96	13.87	12.54	11.15	12.86
16	13.44	12.73	10.78	9.63	12.53

Tabela 11. Wartość błędu średniokwadratowego w zależności od wartości współczynnika uczenia i liczby neuronów w warstwie środkowej.

Liczba neuronów	0.60	0.70	0.80	0.90	1.00	0.5 + Auto 100
1	260.27	266.02	273.29	278.53	288.60	242.39
2	137.50	141.24	143.16	145.33	149.04	132.30
3	93.92	95.48	95.18	98.37	99.49	87.03
4	68.54	67.25	68.57	71.83	72.45	65.28
5	50.51	49.29	52.91	51.00	54.09	46.31
6	40.46	41.95	43.64	43.66	44.54	39.65
7	36.27	36.48	36.49	36.73	39.45	33.68
8	29.55	29.83	31.52	31.23	34.34	28.93
9	24.75	24.34	25.65	27.22	29.21	24.00
10	21.82	20.69	23.36	23.50	25.58	19.73
11	17.39	17.24	17.59	18.56	19.64	16.37
12	15.07	14.58	15.83	16.34	18.74	15.12
13	13.07	13.64	14.34	15.13	17.61	12.31
14	12.00	12.44	12.16	13.36	14.50	12.11
15	10.65	10.53	11.31	13.07	13.92	10.75
16	9.23	9.10	9.45	10.37	12.04	8.80

Tabela 12. Wartość błędu średniokwadratowego w zależności od wartości współczynnika uczenia i liczby neuronów w warstwie środkowej.



Rys 31. Zależność błędu średniokwadratowego od współczynnika uczenia.

Jak widać z wykresu, przy odpowiednio dużej liczbie kroków (100 000) uczących różnice pomiędzy wynikami osiąganymi dla różnych wartości współczynników zacierają się. Wynika to z bardzo prostego faktu, a mianowicie, dla małych współczynników, sieć dostraja się powoli, ale skutecznie dąży do celu jak to widać na rys. 28. Natomiast dla jego dużych wartości punkt, jaki osiąga sieć oscyluje dookoła celu, jakim jest minimum globalne i czasami „trafia" w jego pobliże, po czym ponownie od niego odchodzi. Wprowadzenie współczynnika

dynamicznie zmienianego w zależności od aktualnego zachowania się sieci i zmian błędu średniokwadratowego powoduje początkowo bardzo szybką zbieżność procesu, taką jak w przypadku dużych wartości statycznych, a następnie w okolicy minimum globalnego jego zmniejszenie i „wygładzenie" charakterystyki takie jak w przypadku małych wartości współczynnika uczenia.

- *Liczba kroków uczenia jako element decydujący o jakości odwzorowania obrazu.*

Innym zasadniczym elementem mającym wpływ na dostrojenie sieci jest liczba kroków/iteracji uczących. Jest ona zależna od różnych czynników, a w szczególności od:

- algorytmu uczącego i jego optymalizacji
- współczynnika uczenia
- struktury i rozmiarów sieci

Podstawowym algorytmem uczącym jest w tym wypadku algorytm *propagacji wstecznej błędu*. Jest on podstawowym, ale jednocześnie bardzo wolno zbieżnym algorytmem. Dlatego stosowane są metody jego przyspieszania. Można je podzielić na sposób, w jaki tego dokonują, czyli:

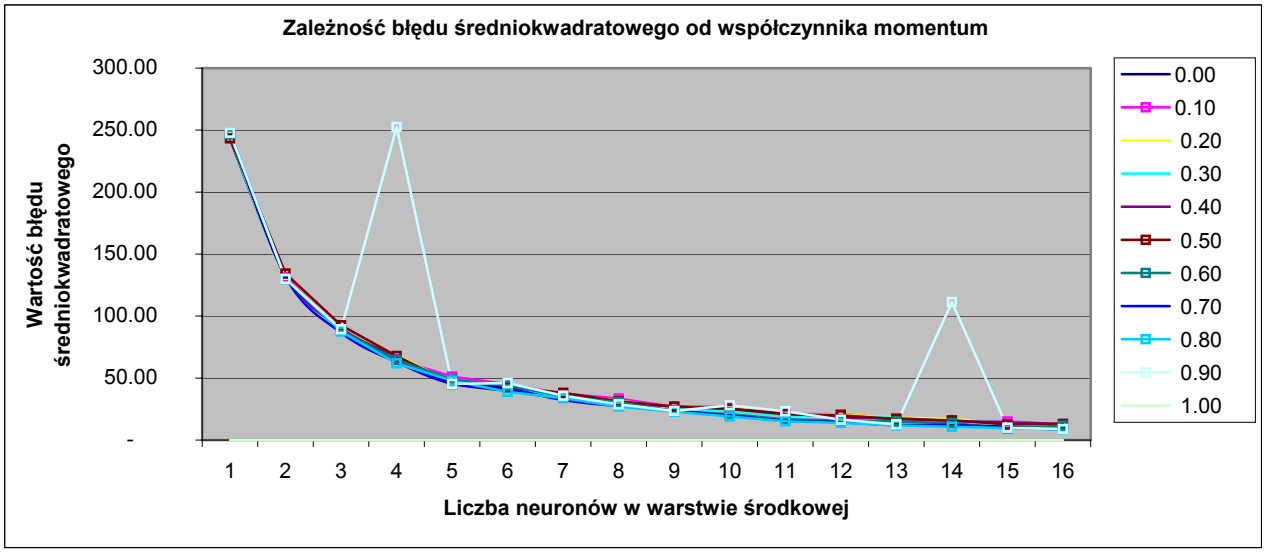
a) Algorytmy gradientowe
   - największego spadku (z zastosowaniem *momentum*)
   - zmiennej metryki
   - Levenberga – Marquardta
   - gradientów sprzężonych

b) Metody doboru współczynnika uczenia
   - dynamiczny dobór współczynnika uczenia
   - minimalizacja kierunkowa
   - reguła delta-bar-delta

c) Gradienty sprzężone z regularyzacją Møllera.

d) Algorytmy heurystyczne
   - Quickprop
   - Rprop

W związku z tym, że optymalizacja procesu uczenia nie jest tematem tej pracy, to zdecydowano się na zastosowanie jedynie dynamicznego doboru współczynnika, który w znaczący sposób wpływa na jakość kompresji, co widać na rys. 27 do 29 oraz 30, jak również eksperymentalnie na element *momentum*, którego wpływ jest tutaj przedstawiony na rys. 32 i jak widać nie wpływa on na poprawienie jakości odtworzonego obrazu, a w niektórych przypadkach wręcz ją pogarsza. Tabela wartości, jakie w wyniku procesu przetwarzania wygenerowała sieć jest dostępna na załączonej płycie CD-ROM w pliku „Porównanie KLT i NN.xls" w zakładce „Momentum".

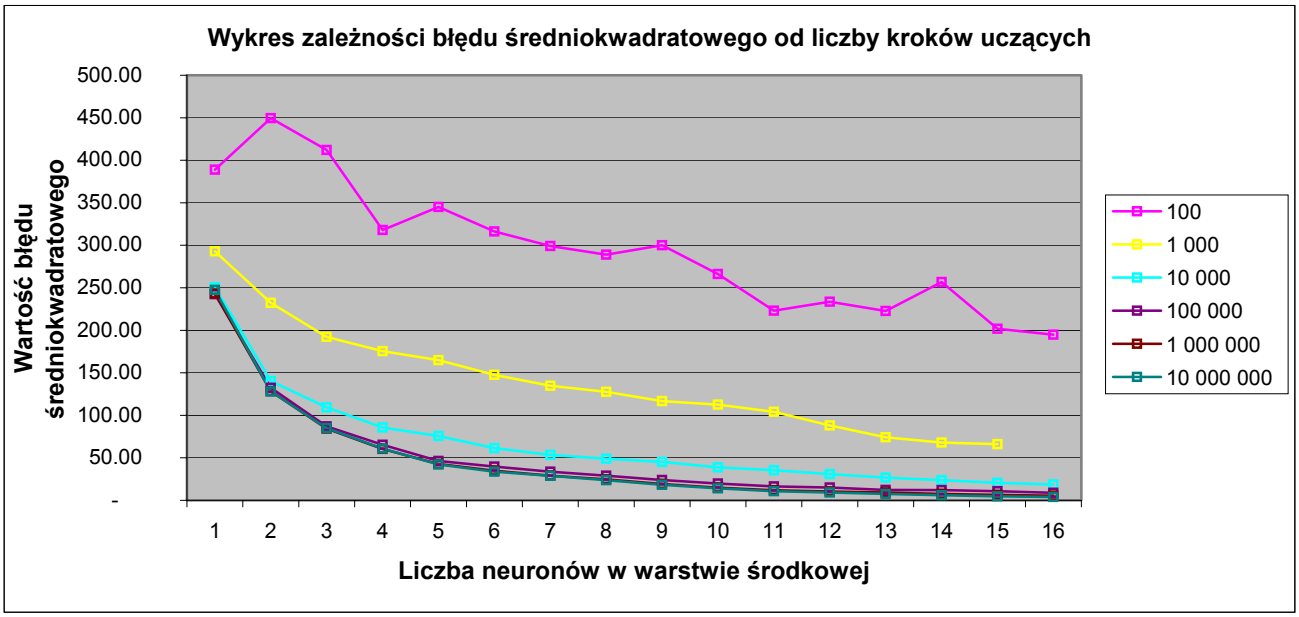


Rys. 32. Wykres zależności błędu średniokwadratowego od współczynnika momentum.

W przypadku liczby kroków uczących sytuacja wydaje się oczywista. Przy odpowiednio dobranych współczynnikach uczenia każdy kolejny krok powinien przybliżać wartości wag wszystkich połączeń do wartości optymalnych. Zbyt mała liczba kroków nie pozwala na wytrenowanie sieci w sposób wystarczający i tak jak to widać na rys. 33 dla linii odpowiadającej 100 iteracjom, wartości błędu średniokwadratowego są zdecydowanie dalekie od tego, jakie można osiągnąć już choćby dla 10 000. Bardzo szybka zbieżność na początku procesu uczenia zmniejsza się wraz z jego postępem i przy przejściu z 1 000 000 na 10 000 000 widać tylko bardzo minimalną poprawę i w efekcie zmniejszenie błędu średniokwadratowego.

Liczba neuronów	100	1 000	10 000	100 000	1 000 000	10 000 000
1	388.92	292.86	250.55	242.39	243.16	247.53
2	449.59	232.38	140.17	132.30	128.41	128.05
3	412.01	192.15	109.35	87.03	84.28	84.91
4	317.98	175.51	85.58	65.28	60.35	60.74
5	345.06	164.97	75.74	46.31	42.74	42.23
6	316.21	147.53	61.34	39.65	34.88	33.67
7	299.06	134.87	53.42	33.68	29.00	28.79
8	288.98	127.69	49.01	28.93	24.62	23.68
9	300.11	116.79	45.21	24.00	19.23	18.34
10	266.15	112.66	38.77	19.73	14.78	14.06
11	223.04	104.07	35.47	16.37	11.94	10.74
12	233.53	88.01	30.78	15.12	10.27	9.16
13	222.76	74.11	26.74	12.31	9.15	7.53
14	256.85	67.93	23.64	12.11	7.60	5.96
15	201.75	66.04	20.49	10.75	6.43	4.77
16	194.91	63.96	18.59	8.80	5.40	4.02

Tabela 13. Wartość błędu średniokwadratowego w zależności od liczby kroków uczenia i liczby neuronów w warstwie środkowej



Rys. 33. Wykres zależności błędu średniokwadratowego od liczby iteracji.

Przy podejmowaniu decyzji o liczbie kroków uczących, jakie mamy zamiar zastosować należy wziąć pod uwagę dwa podstawowe czynniki: jakość obrazu, jaki chcemy uzyskać oraz czas, jaki możemy poświęcić na strojenie sieci. Program „Kompresja Neuronowa” daje możliwość ustawienia liczby kroków w zakresie od 1 do 2^{31}, a więc praktycznie umożliwia pełne wyuczenie i przyporządkowanie optymalnych wartości dla wszystkich wag.

- *Wpływ przybliżenia wag początkowych na podstawie transformaty KLT dla sieci 2 i 3 warstwowej.*

Założeniem tej pracy było również zbadanie, jaki wpływ na ostateczny wynik działania sieci neuronowej ma przybliżenie początkowych wartości wag na podstawie transformaty KLT. Pierwotne założenia było takie, że struktura działania w/w transformaty i sieci neuronowej są podobne, a za jej pomocą można szybko obliczyć początkowe wartości wag i tym samym uniknąć wstępnego etapu treningu sieci oszczędzając na czasie. Założenie to jest słuszne dla sieci, w których funkcje aktywacji odpowiadają funkcji jednostkowej $f(net) = net$. W takim wypadku faktycznie można dokonać takiego podstawienia, a z drugiej strony jako ciekawostka, sama sieć może posłużyć jako algorytm do obliczania macierzy wektorów własnych macierzy kowariancji danych podawanych na wejście tej sieci. Jednak taki model neuronu nie pozwala na pełne wykorzystanie możliwości, jakie daje nam sieć neuronowa zbudowana w oparciu o nieliniową, ciągłą funkcję aktywacji w postaci unipolarnej

$$f(net) = \frac{1}{1 + \exp(-\lambda net)}$$

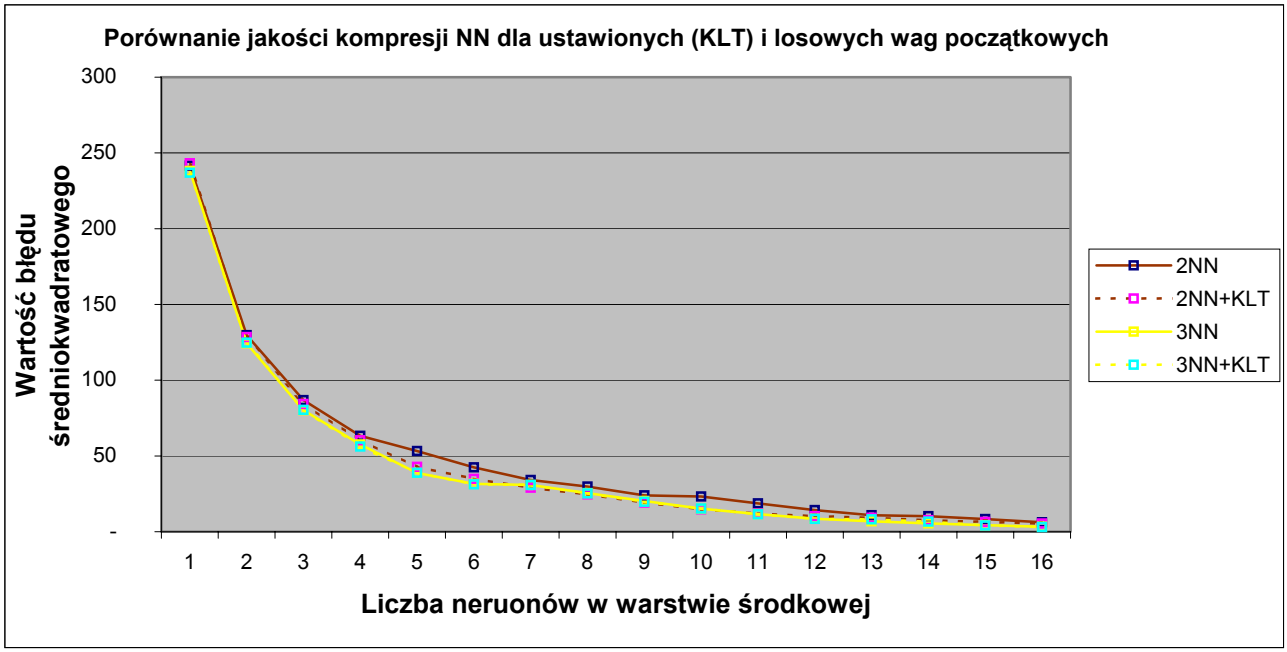
oraz również bipolarnej

$$f(net) = \frac{2}{1 + \exp(-\lambda net)} - 1$$

Pomimo takiego stanu rzeczy, do celów doświadczalnych, jak również dydaktycznych podjęto próbę przybliżenia wartości bazując na przekształceniu wartości otrzymanych w wyniku KLT w postaci macierzy transformacyjnej. W wyniku tego badania otrzymano bardzo interesujący wynik w postaci nieznacznego poprawienia dokładności dostrojenia sieci i w rezultacie poprawienia jakości przetwarzanego obrazu. Początkowo nadane wartości już w pierwszej iteracji ulegają przestrojeniu, a w następnych ulegają całkowitej zmianie, by w rezultacie została im przypisana całkowicie inna wartość. Pomimo wszystko takie wstępne ustawienie wag daje lepsze wyniki w porównaniu z wartościami losowymi, co widać na rys.34 oraz tabeli 14.

Liczba neuronów	Dwie warstwy		Trzy Warstwy	
	2NN	2NN+KLT	3NN	3NN+KLT
1	241.59	243.16	238.14	236.97
2	129.77	128.41	124.34	124.72
3	86.82	84.28	80.17	80.35
4	63.23	60.35	57.59	56.15
5	53.20	42.74	38.73	38.89
6	42.48	34.88	31.54	31.27
7	34.17	29.00	30.82	30.89
8	29.74	24.62	25.41	25.29
9	24.03	19.23	20.24	19.74
10	23.23	14.78	15.24	15.29
11	18.65	11.94	11.66	11.48
12	14.23	10.27	8.57	8.47
13	10.89	9.15	6.88	8.16
14	10.25	7.60	5.50	6.61
15	8.32	6.43	4.32	4.17
16	6.27	5.40	3.24	3.02

Tabela 14. Wpływ ustawienia wartości początkowych wag sieci na podstawie macierzy transformacyjnej przekształcenia KLT.

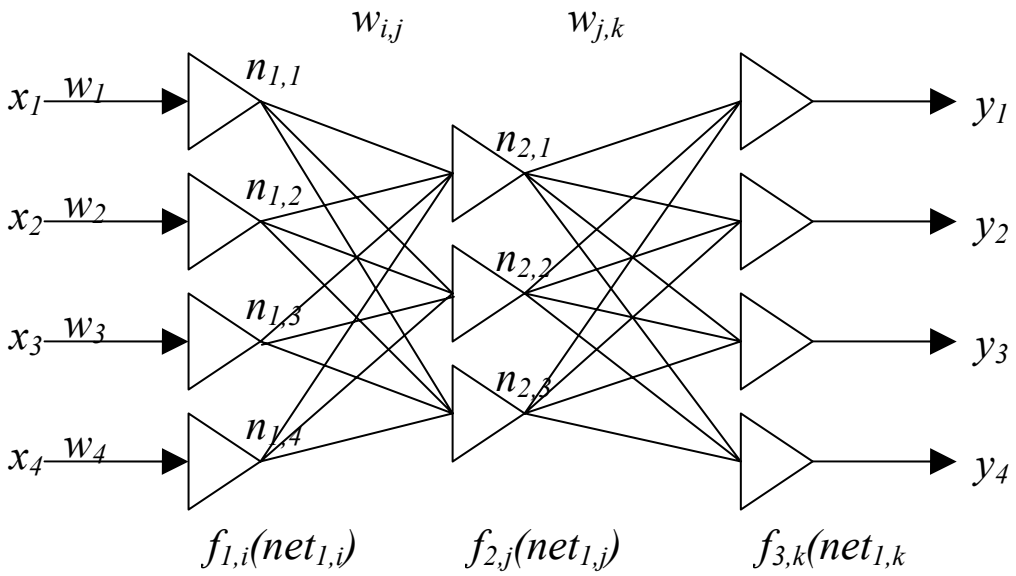


Rys. 34. Wpływ ustawienia wartości początkowych wag sieci na podstawie macierzy transformacyjnej przekształcenia KLT.

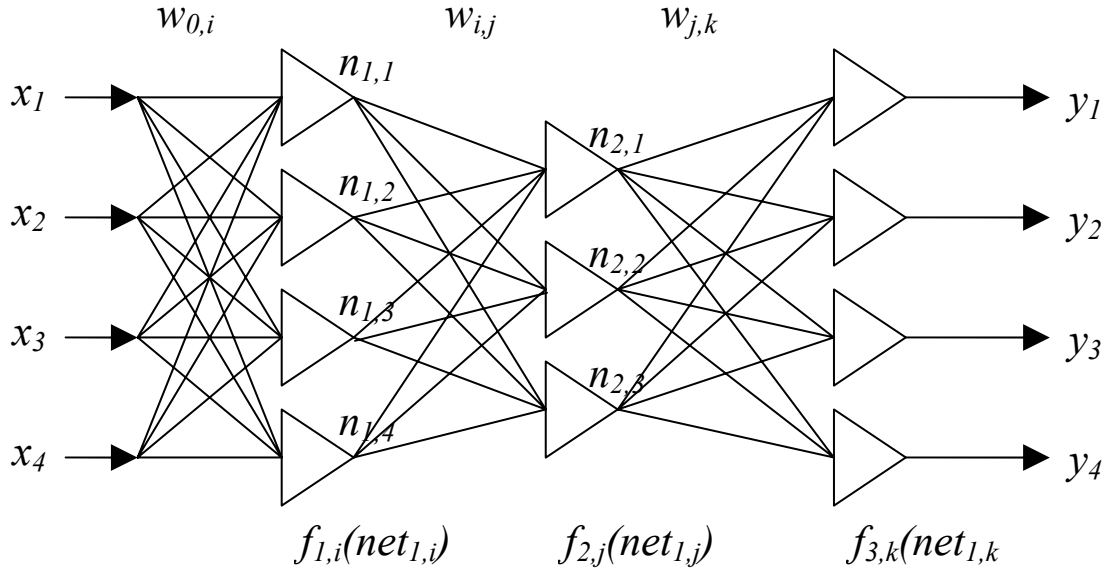
Wyraźnie widać w tabeli 14, że omawiane tutaj początkowe przybliżenie wag ma dużo większy wpływ na jakość przetwarzania dla sieci dwuwarstwowej niż trójwarstwowej. Jest to oczywiste biorąc pod uwagę, że dodatkowa warstwa sieci wprowadza swoje modyfikacje do przetwarzanych sygnałów i znacznie je zniekształca, co jeszcze bardziej uniezależnia ją od zastosowanych przybliżeń.

- *Zasada funkcjonowanie sieci dwu- i trójwarstwowej.*

Dla lepszego zobrazowania różnicy pomiędzy budową sieci dwu i trójwarstwowej poniżej przedstawiono schemat budowy obu z nich.



Rys. 35a. Schemat budowy sieci dwuwarstwowej.



Rys. 35b. Schemat budowy sieci trójwarstwowej.

W sieci dwuwarstwowej sygnały wejściowe są podawane bezpośrednio na warstwę „0”, gdzie po przejściu bez zmian przez neurony, których jednostkowa funkcja aktywacji nie zmienia ich wartości. Dopiero z tego miejsca wartości te podawane są na ważone wejścia warstwy ukrytej (środkowej), a następnie dalej zgodnie z zasadami działania sieci.

W przeciwieństwie do poprzedniej sytuacji w sieci trójwarstwowej już na samym początku podawane są na ważone wejścia neuronów warstwy „0” gdzie są sumowane w pobudzenie łączne *net* i przetwarzane przez ciągłą i nieliniową funkcję aktywacji *f(net)*. Następnie przetwarzanie przebiega identycznie jak dla sieci dwuwarstwowej.

- *Jakość kompresji w zależności od rodzaju obrazu.*

Kolejnym bardzo istotnym czynnikiem mającym wpływ na jakość kompresji, jest rodzaj przetwarzanego obrazu. Podstawowym elementem definiującym typ obrazu z punktu widzenia sieci neuronowej, jak również ogólnie kompresji obrazu jest zawartość w jego treści elementów takich jak kontury, faktura i tło. Kontury stanowią najbardziej zróżnicowane obszary obrazu, o największych skokach jasności pikseli, a tym samym są najtrudniejsze do odwzorowania i potrzebują większej liczby elementów przekształcenia KLT czy też neuronów sieci neuronowej w porównaniu do pozostałych. Z drugiej strony tło jest obszarem o najmniejszym zróżnicowaniu wartości pikseli, co pozwala poświęcić im zdecydowanie mniej elementów niż w przypadku konturów.

W celu zbadania zachowania się sieci NN, jak również transformaty KLT, przygotowano sześć najczęściej spotykanych typów obrazów (*„Lena”, „Krasnal”, „Pejzaż”, „Faks”, „Tony”, „Tło”*). Zostały one przedstawione na stronach 61 i 62 (rys. 19) tej pracy. Natomiast wyniki działania sieci na wszystkich tych przypadkach zaprezentowano w tabeli 7 oraz na rys. 18 na stronie 60.

We wszystkich przypadkach sieć zachowuje się podobnie do standardowych algorytmów kompresji, w których wykorzystuje się metody słownikowe, czy też probabilistyczne, jak również do metod kompresji stratnej.

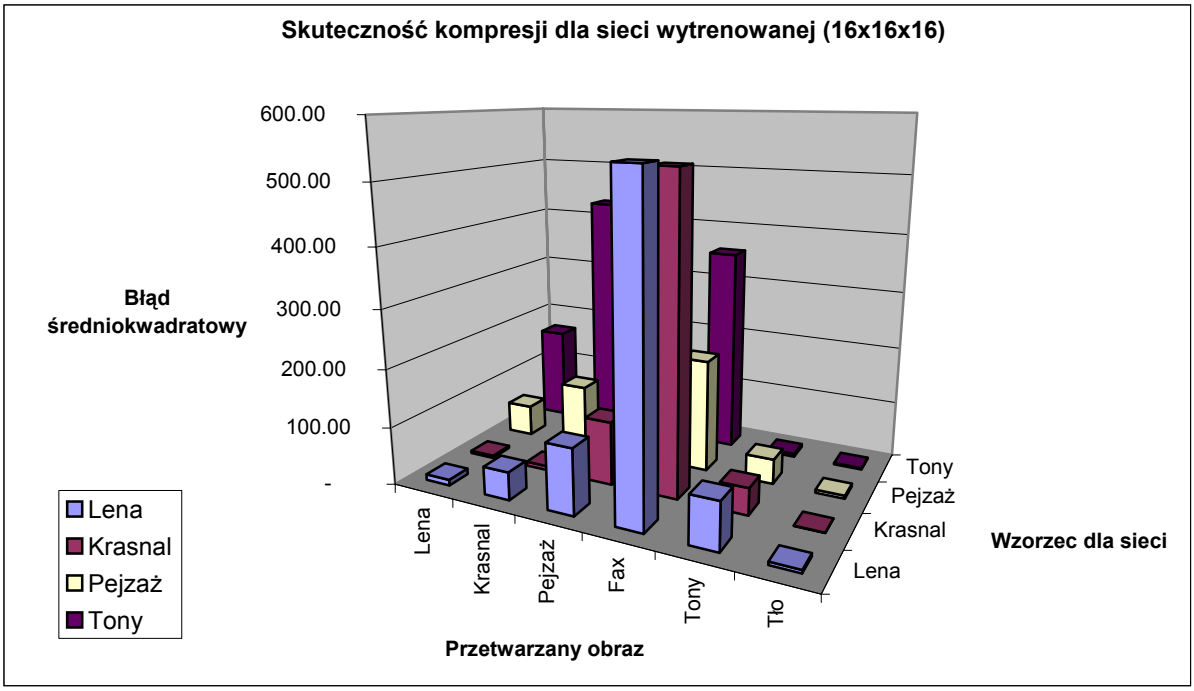
- *Skuteczność kompresji różnych obrazów za pomocą raz nauczonej sieci.*

Działanie sieci neuronowej charakteryzuje pewna bardzo ciekawa właściwość, a mianowicie taka, że dla raz wytrenowanej sieci możliwe jest przetwarzanie kolejnych obrazów tego samego typu bez konieczności ponownego treningu. Cecha ta wiąże się ściśle z kolejną własnością, jaką jest wyszukiwanie pewnych podobieństw w podawanych paczkach danych i uogólnianie ich, a także tworzenie pewnych wewnętrznych reguł, które mogą być przeniesione na inne paczki danych.

Do celów badawczych przeprowadzono szereg testów badając możliwość kompresji różnych obrazów za pomocą sieci wytrenowanej na innych wzorcach. Poniżej przedstawiono wyniki w postaci tabeli i wykresu. Obiektem badań było sześć obrazów różnych typów, a sieć dla każdego z nich dostrajana była na drodze 1 000 000 iteracji i zapisywana na dysku. Wszystkie wyniki w postaci przetworzonych obrazów, ich histogramów i wartości wag zapisane zostały na załączonym dysku CD-ROM w katalogu „Szybkie".

Skuteczność kompresji wytrenowaną siecią						
16 x 4 x 16						
Wzorzec	**Lena**	**Krasnal**	**Pejzaż**	**Faks**	**Tony**	**Tło**
Lena	**59.84**	212.72	195.45	841.80	141.14	18.66
Krasnal	73.58	**215.96**	340.20	1 228.52	176.00	20.90
Pejzaż	102.03	252.61	**43.52**	357.13	96.36	24.01
Faks	3 647.03	2 989.24	1 114.32	**200.80**	2 938.02	5 530.29
Tony	449.12	1 297.94	303.05	631.08	**18.27**	65.71
Tło	928.39	1 449.63	5 182.45	10 181.15	2 062.14	**0.05**
16 x 8 x 16						
Lena	**24.27**	88.10	160.33	710.40	99.14	7.24
Krasnal	29.77	**80.25**	201.75	831.34	92.81	9.31
Pejzaż	83.62	197.68	**30.53**	321.36	51.85	13.32
Faks	3 718.77	2 517.65	1 192.31	**135.75**	2 898.48	6 774.53
Tony	213.47	609.70	183.61	554.66	**16.94**	5.59
Tło	1 055.45	1 417.73	4 137.71	8 273.94	1 811.89	**0.01**
16 x 16 x 16						
Lena	**10.06**	48.16	112.87	542.04	80.36	5.48
Krasnal	4.23	**7.39**	110.05	528.51	46.23	0.69
Pejzaż	55.07	111.52	**14.40**	195.62	43.56	5.42
Faks	5 185.89	3 033.58	1 798.42	**53.15**	3 547.57	11 981.83
Tony	166.88	429.97	91.32	356.09	**5.35**	2.40
Tło	1 036.85	1 391.11	3 197.70	6 321.30	1 571.30	**0.01**

Tabela 15. Skuteczność kompresji różnych obrazów za pomocą raz wytrenowanej sieci. Struktury sieci zawierają odpowiednio 4, 8 i 16 neuronów w warstwie środkowej. Wzorzec, na którym była trenowana sieć jest w kolumnie „Wzorzec".



Rys. 36. Przykładowe zestawienie możliwości kompresji obrazu siecią wytrenowaną na innym wzorcu.

Aby nie zaciemniać wykresu na rys. 36 usunięto z niego wyniki otrzymane przy pomocy sieci wytrenowanej na wzorcach „Faks" oraz „Tło", które osiągały wartości zdecydowanie powyżej 1000.

Widać, że nie wszystkie obrazy można równie skutecznie w ten sposób skompresować. Lepszą skuteczność widać na obrazach podobnych, takich jak np. „Lena" i „Krasnal", dla których w skrajnym przypadku sieć wytrenowana na tym drugim i zastosowana do „Leny" dała lepsze rezultaty niż trenowana na wzorcu i to około dwukrotnie! Natomiast dla skrajnie różnych typów obrazów takich jak np. „Tło" i „Faks" stosowanie tej metody to kompletne nieporozumienie, co widać na przykładzie sieci 16x16x16 w kolumnie „Tło", gdzie przy standardowym wyniku poniżej 10, sieć wytrenowana na „Faks" wygenerowała błąd średniokwadratowy powyżej 10 000!

Dlatego też w rzeczywistym oprogramowaniu, w którym chcielibyśmy zastosować raz już wytrenowaną sieć należałoby po pierwsze wygenerować i zapisać szereg wag dla różnych obrazów wzorcowych w stosunku do tego, jakie rzeczywiste obrazy będą podawane na wejście, oraz po drugie umieścić przed programem do kompresji procedury oparte na innej sieci, a służące do rozpoznawania typu obrazu, aby w ten sposób uniknąć zdecydowanych przekłamań.

- *Rozmiar wynikowego obrazu dla różnych obrazów i różnej liczby neuronów w warstwie środkowej wygenerowanego programem „Kompresja Neuronowa”.*

Jak ogólnie wiadomo, różne rodzaje danych, a w szczególności obrazów w różnym stopniu poddają się procesowi kompresji. Również i tutaj można było się przekonać o prawdziwości tego stwierdzenia. W katalogu „Wynik kompresji” zapisano pliki wynikowe otrzymane jako rezultat końcowy kompresji obrazów różnych typów dla różnej liczby neuronów pozostawionych w warstwie środkowej. Badania przeprowadzono dla podobrazów o rozmiarach 4x4 piksele oraz liczbie iteracji równej 100 000. Porównanie wielkości plików przedstawiono w tabeli 16.

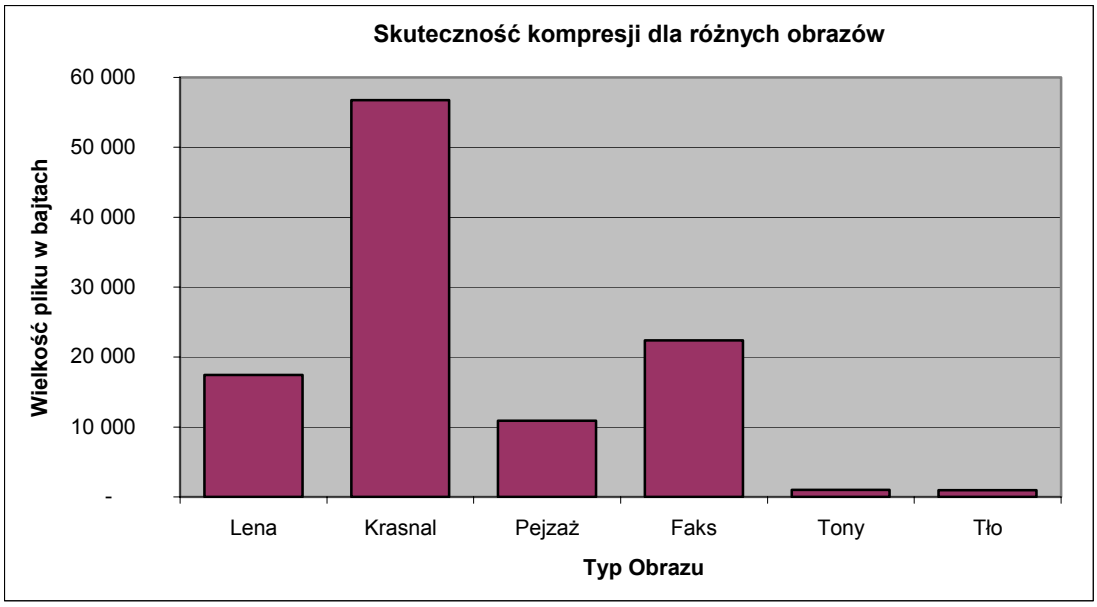
Liczba neuronów	Lena	Krasnal	Pejzaż	Faks	Tony	Tło
1	6 033	6 543	5 552	800	1 003	965
2	11 897	11 676	10 903	3 014	1 222	1 248
3	15 955	19 452	15 087	5 203	1 500	1 471
4	17 465	26 162	22 131	8 309	1 767	1 735
5	23 611	31 779	27 467	9 340	2 069	2 026
6	25 637	38 322	33 349	11 379	2 342	2 083
7	30 654	44 471	39 865	13 360	2 713	2 432
8	38 075	50 491	43 907	15 959	3 020	2 642
9	40 052	56 750	51 839	16 043	3 184	2 737
10	44 243	62 990	57 764	17 523	3 440	2 947
11	50 962	68 960	63 314	18 534	3 817	3 116
12	55 550	74 846	67 908	18 877	4 166	3 204
13	59 709	81 724	74 721	22 382	4 493	3 626
14	63 759	86 999	78 948	22 811	4 843	4 132
15	68 470	94 301	85 534	22 959	4 998	4 421
16	84 182	99 265	89 508	24 278	5 389	4 668

Tabela 16. Porównanie wielkości plików wynikowych otrzymanych w procesie kompresji danych programem „Kompresja Neuronowa”.

Porównując ją do tabeli 7 można teraz dość wyraźnie powiedzieć, w jakim stopniu każdy z powyższych obrazów przykładowych można skompresować. Załóżmy, że interesuje nas taka kompresja, w której błąd średniokwadratowy będzie przyjmował wartości pomiędzy 50 i 60. W tabeli 7 wyszukujemy odpowiednią liczbę neuronów warstwy środkowej, po czym z tabeli 17 wypisujemy odpowiedni wynik, co dokładnie pokazuje poniższe zestawienie.

	Lena	Krasnal	Pejzaż	Faks	Tony	Tło
Neurony	4	9	2	13	1	1
Bajty	17 465	56 750	10 903	22 382	1 003	965

Tabela 17. Porównanie wielkości plików wynikowych dla błędu średniokwadratowego z zakresu od 50 do 60.



Rys.37. Porównanie wielkości plików wynikowych.

Powyższe badania przeprowadzone zostały dla obrazów o rozmiarach 256x256 pikseli w 8bitowej skali odcieni szarości. Rozmiar plików wzorcowych to 64KB.

Teraz widać wyraźnie jak bardzo skuteczność kompresji jest zależna od rodzaju danych, do jakich jest stosowana. Obrazy o małej zawartości elementów takich jak krawędzie, czy kontury poddają się procesowi rewelacyjnie i przy niewielkiej utracie jakości można zmniejszyć ich rozmiar nawet kilkudziesięciokrotnie, natomiast te z dużą zawartością skomplikowanych wzorów i elementów, poddają się kompresji w bardzo niewielkim stopniu, co widać bardzo wyraźnie na rys. 37.

- *Szybkość uczenia.*

Istotnym czynnikiem każdego algorytmu kompresji danych jest jego wydajność, to znaczy czas, po jakim otrzymujemy wynik w postaci skompresowanego obrazu, czy też odtworzenia go. Parametr ten decyduje w wielu wypadkach o ogólnej przydatności danego algorytmu do niektórych zastosowań. W przypadku, gdy mamy

do czynienia z obrazami nieruchomymi, które są archiwizowane i składowane na nośnikach danych parametr ten nie jest wartością krytyczną i tutaj zdecydowanie ważniejszy jest stopień kompresji i jakość, z jaką obraz może być odtworzony. Natomiast dla transmisji obrazu ruchomego przez sieć, np. w telefonii komórkowej, przez internet, systemy monitorujące, gdzie istotnym czynnikiem jest płynność odtwarzanego obrazu, czas ma decydujące znaczenie.

Sieć neuronowa oferuje nam możliwości, które pozwalają nam na jej zastosowanie w obu powyższych przypadkach. Zarówno kompresja obrazu za pomocą raz wyuczonej sieci, jak i jego dekompresja przebiegają błyskawicznie. Jednak sam trening sieci jest procesem bardzo czasochłonnym i wymagającym bardzo szybkich komputerów.

Dla przykładu przedstawiona zostanie złożoność obliczeniowa dla jednej iteracji, dla sieci dwuwarstwowej, podobraz 4x4 piksele. Wiersz oznaczony jako *forward* odpowiada przejściu sieci „w przód", a wiersz *back* przejściu sieci „w tył". Kolumny oznaczone jako „+", „-" i „x" oznaczają odpowiednie działania arytmetyczne, a f(net) to obliczenie funkcji aktywacji, która pierwotnie zapamiętana w odpowiedniej tablicy może być natychmiast z niej odczytywana.

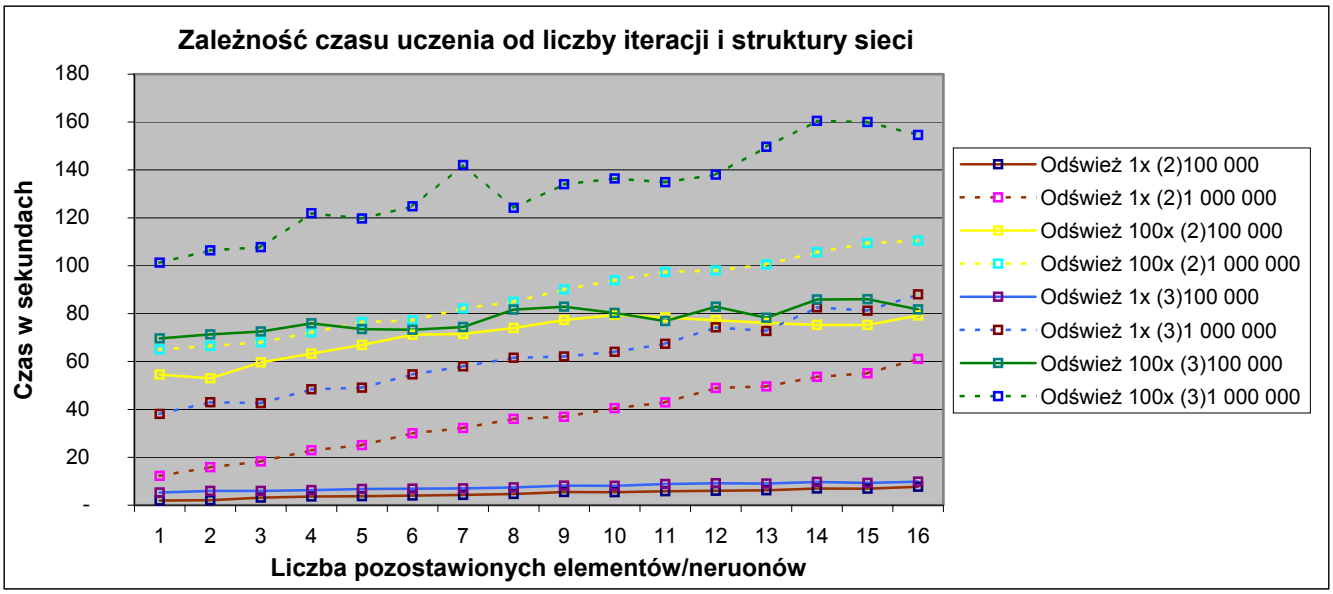
	+	x	-	f(net)
norm	128	128	-	20
back	256	604	348	-

Tabela 18. Liczba operacji matematycznych potrzebnych do zrealizowania jednego kroku uczącego.

Tak duża liczba operacji matematycznych pociąga za sobą konieczność zaangażowania ogromnej mocy obliczeniowej. Dla przykładu w tabeli 19 przedstawiono czas obliczeń dla sieci dwu- i trójwarstwowej, różnej liczby iteracji oraz częstotliwości zbierania wyników na komputerze klasy PC z procesorem Pentium IV 2.4GHz.

Liczba neuronów	Dwie warstwy				Trzy Warstwy			
	Odśwież 1x		Odśwież 100x		Odśwież 1x		Odśwież 100x	
	(2)100 000	(2)1 000 000	(2)100 000	(2)1 000 000	(3)100 000	(3)1 000 000	(3)100 000	(3)1 000 000
1	1.94	12.25	54.51	65.05	5.30	38.08	69.64	101.25
2	2.04	15.87	53.01	66.50	5.98	42.98	71.29	106.38
3	3.15	18.25	59.65	67.95	5.96	42.58	72.50	107.75
4	3.61	**22.90**	63.29	72.19	6.25	48.37	75.93	121.91
5	3.75	25.06	66.90	76.47	6.76	49.06	73.54	119.73
6	4.01	30.00	71.16	77.17	6.90	54.58	73.25	124.81
7	4.25	32.24	71.55	82.12	7.00	57.94	74.40	142.02
8	4.70	36.00	73.96	84.86	7.43	61.53	81.67	124.21
9	5.50	36.90	77.32	90.10	8.15	62.12	82.89	134.00
10	5.40	40.50	79.40	93.93	8.11	64.01	80.25	136.40
11	5.80	42.90	78.37	97.38	8.79	67.40	76.86	134.88
12	6.02	48.90	77.18	98.02	9.14	74.27	82.90	138.00
13	6.22	49.62	76.15	100.54	9.03	72.75	78.25	149.66
14	6.96	53.62	75.26	105.63	9.69	82.62	85.88	160.51
15	6.90	55.10	75.24	109.42	9.30	81.22	86.01	160.01
16	7.68	61.08	79.19	110.51	9.85	88.00	81.76	154.58

Tabela 19. Czas trwania procesu uczenia sieci w zależności od liczby iteracji, częstotliwości zbierania wyników i struktury sieci.



Rys.38. Czas trwania procesu uczenia sieci w zależności od liczby iteracji, częstotliwości zbierania wyników i struktury sieci

Obliczając średni czas jednej iteracji na podstawie tabeli 19 (pole zaznaczone na niebiesko) otrzymujemy 0.00002290s dla jednego podobrazu 4x4 piksele. Dla całego obrazu 256x256 pikseli czas ten zwiększy się do 0.09s, co daje nam 10.66 obrazu na sekundę. Jednak, jest to wartość dla całego kroku uczącego, czyli „w przód” oraz „w tył”, a w procesie kompresji siecią wyuczoną nie ma potrzeby

realizowania korekty wag - kroku „w tył”. W takim przypadku biorąc pod uwagę złożoność obliczeniową każdego z tych etapów możemy zwiększyć liczbę obrazów przetwarzanych w ciągu jednej sekundy do ok. 40, co w zupełności wystarczy do płynnego odtwarzania obrazu.

Widać, że już na tym etapie badana sieć daje nam ogromne możliwości wykorzystania jej w dzisiejszym, codziennym życiu w transmisji sygnałów cyfrowych, jakimi są obrazy, jednak dopiero implementacja dodatkowej optymalizacja procesu uczenia sieci sprawi, że obrazy te przetwarzane na bieżąco będą charakteryzowały się doskonałą jakością.

Rozdział 6
Opis programu

Przedstawiony tu zostanie program „Kompresja Neuronowa", jego szczegółowy opis oraz wymagania sprzętowe.

6.1 Wymagania sprzętowe

Symulacja działania sieci neuronowych opiera się na równoległym przetwarzaniu informacji we wszystkich węzłach sieci, tj. neuronach. W związku z tym najlepszym rozwiązaniem byłoby zastosowanie klastrów składających się z komputerów o dużej mocy obliczeniowej, jednak ze względu na znaczny koszt takiego rozwiązania i trudności natury technicznej zdecydowano się na implementację całego algorytmu w postaci programu „Kompresja Neuronowa" działającego na pojedynczym komputerze klasy PC. Komputer taki powinien jednak spełniać następujące warunki:

- Wymagania minimalne
  - system Windows (98, Me, 2000, XP, 2003)
  - 32MB pamięci operacyjnej
  - 10MB wolnego miejsca na dysku twardym
  - procesor Pentium 100MHz
  - monitor kolorowy 14"
- Konfiguracja zalecana
  - Windows 2000/XP
  - 256MB pamięci operacyjnej
  - 100MB wolnego miejsca na dysku twardym
  - procesor Pentium IV 2.4GHz
  - monitor kolorowy 17"

Lepsze parametry komputera przeznaczonego do obliczeń neuronowych oznaczają jednocześnie krótszy czas oczekiwania na rezultat w postaci wytrenowanej sieci, a w rezultacie uzyskanie większego stopnia kompresji przy wierniej odtworzonym obrazie wynikowym. Uzależnienie od wydajności komputera najlepiej widoczne jest w module analiz seryjnych, gdzie przy ustawieniu wielu różnych zmiennych oczekiwanie na wynik może trwać kilkaset tysięcy lat...

6.2 Podręcznik użytkownika

Dokumentacja programu do badań nad kompresją obrazów nieruchomych za pomocą sieci neuronowej.

Aplikacja ta powstała w ramach pracy dyplomowej na temat kompresji obrazów nieruchomych za pomocą sieci neuronowej.

W założeniu jest to narzędzie testowe pozwalające na wszelkiego rodzaju ustawienia sieci neuronowej z ciągłą zarówno bipolarną, jak i unipolarną funkcją aktywacji. Dodatkowo do celów porównawczych zaimplementowana została kompresja obrazów za pomocą transformaty KLT.

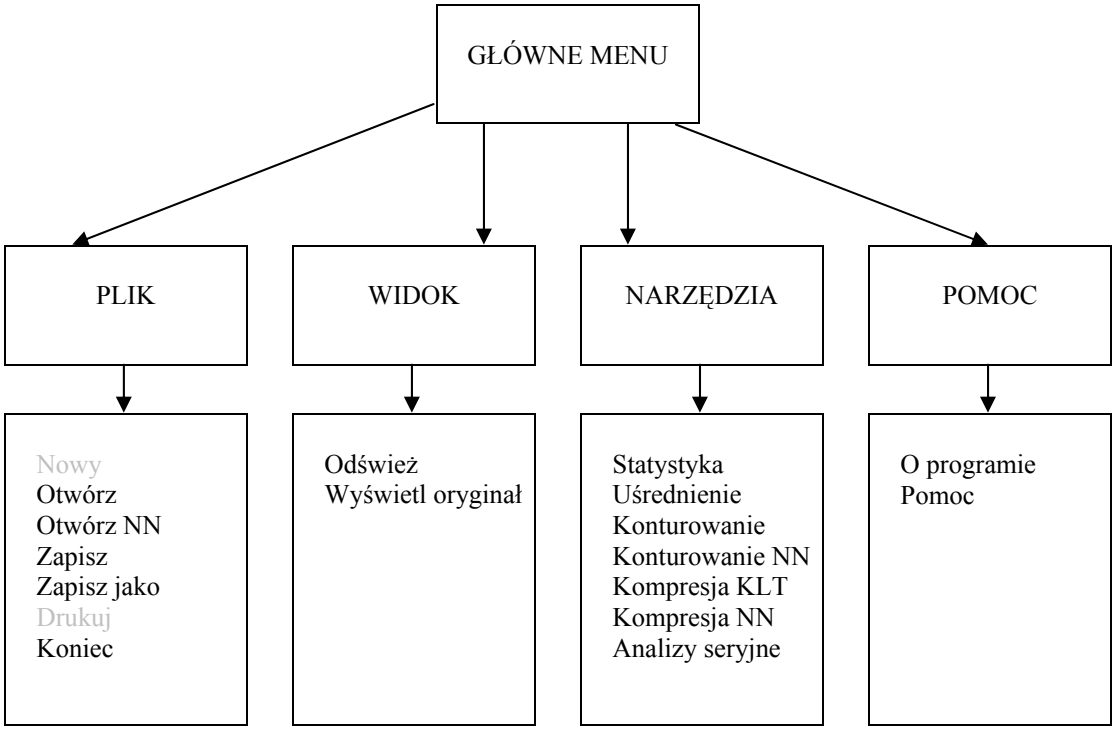
Całość powstała na bazie środowiska programistycznego Delphi 6.

Każdy z otrzymanych wyników można zapisać do pliku wykorzystując do tego celu podręczne menu ukryte pod prawym przyciskiem myszy. Dodatkowo umieszczono tam możliwość wyświetlenia odpowiednich statystyk.

Rys.39. Przykładowe menu podręczne dla wyniku kompresji KLT.

W wyniku takiego zastosowania menu podręcznego znacznie została uproszczona procedura zapisu danych wynikowych takich jak przetworzony obraz, czy jego histogram. Ponadto jest to dużo bardziej intuicyjne niż wybór odpowiedniej opcji z głównego menu.

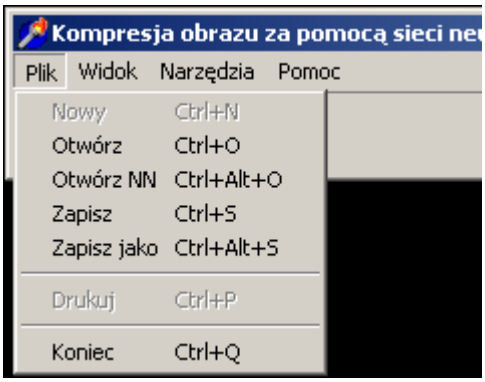
Poniżej na rysunku 40 przedstawiono schemat funkcjonalny programu



Rys. 40. Schemat funkcjonalny programu „Kompresja Neuronowa”.

POSZCZEGÓLNE FUNKCJE PROGRAMU

PLIK



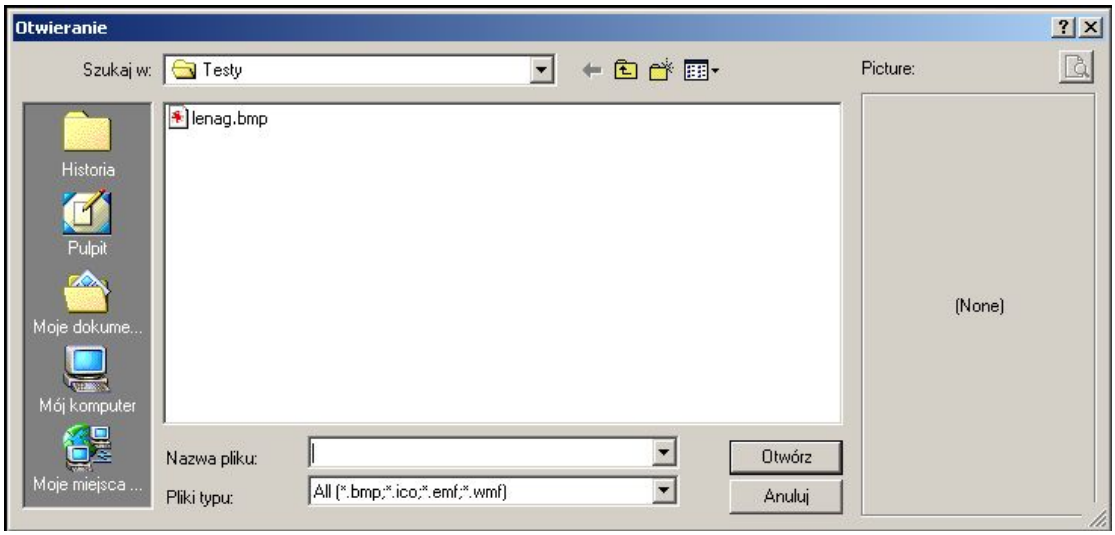
Rys.41. Menu „PLIK”.

NOWY

Tworzenie nowego obrazu – Funkcja przewidziana do przyszłej implementacji

OTWÓRZ

W celu wykonywania jakiejkolwiek późniejszej obróbki musi zostać wczytany plik graficzny. Dopuszczalne formaty to *.BMP, *.ICO, *.EMF, *.WMF. Tak wczytany obraz podlega automatycznej zamianie na postać o 8 bitowej skali odcieni szarości.



Rys.42. Otwarcie pliku graficznego do dalszego przetwarzania.

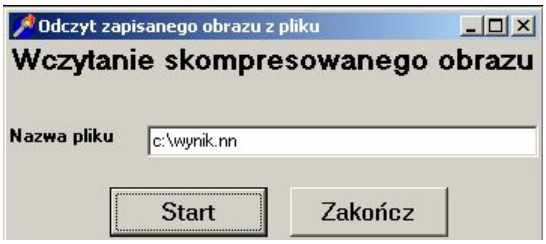
OTWÓRZ NN

Otwarcie pliku zakodowanego w procesie kompresji wymaga odpowiednich procedur dekompresyjnych zaimplementowanych w tej właśnie opcji. Kolejne etapy odtworzenia obrazu to :

- stworzenie i uruchomienie pliku wsadowego unrar.bat

  rar.exe x -c- -o+ -y wynik.nn c:\ >> Err.log

  który uruchamia program rar.exe i odkodowuje wstępnie do postaci zrozumiałej dla sieci neuronowej. Wszystkie komunikaty, w tym o błędach umieszczane są w pliku Err.log
- wczytanie parametrów sieci neuronowej oraz wartości wag warstwy wyjściowej
- wczytywanie wartości na wyjściu neuronów warstwy środkowej, a następnie obliczanie wartości kolejnych podobrazów na wyjściu sieci.
- wyświetlenie odtworzonego obrazu.



Rys.43. Odczyt skompresowanego pliku.

ZAPISZ

Zapis aktualnie otwartego obrazu pod tą samą nazwą.

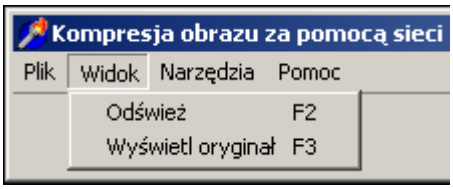
ZAPISZ JAKO

Zapis aktualnie otwartego obrazu pod wybraną nazwą w jednym z formatów *.BMP, *.ICO, *.EMF, *.WMF. Okno zapisu podobne do okna wyświetlanego podczas otwierania pliku.

DRUKUJ

Wydruk aktualnego obrazu – Funkcja przewidziana do przyszłej implementacji

WIDOK



Rys.44. Menu „Widok”.

ODŚWIEŻ

Opcja pozwalająca na przywrócenie oryginalnego obrazu po wszelkiego rodzaju operacjach zmieniających jego zawartość takich jak np. uśrednienie, czy konturowanie.

Rys.45. Wybranie opcji „Odśwież" powoduje przywrócenie obrazu oryginalnego.

WYŚWIETL ORYGINAŁ

Czasami zachodzi potrzeba wyświetlenia oryginalnego obrazu w celu porównania go do wyniku otrzymanego po przetwarzaniu poszczególnymi funkcjami programu.

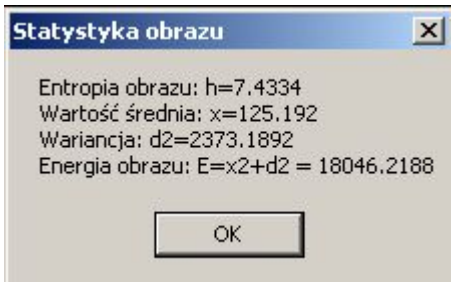
Rys.46. Wyświetlenie oryginalnego obrazu w osobnym oknie w celu późniejszego porównania go z wynikami przetwarzania.

NARZĘDZIA

Rys.47. Menu „Narzędzia".

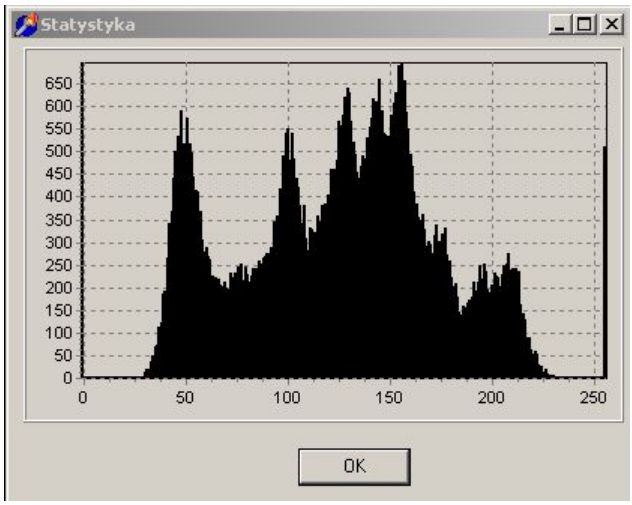
STATYSTYKA

Poza wizualną oceną wczytanego obrazu możliwe jest również obliczenie wybranych właściwości statystycznych oraz przedstawienie ich w postaci numerycznej



Rys.48. Wartości statystyczne dla obrazu „Lena”.

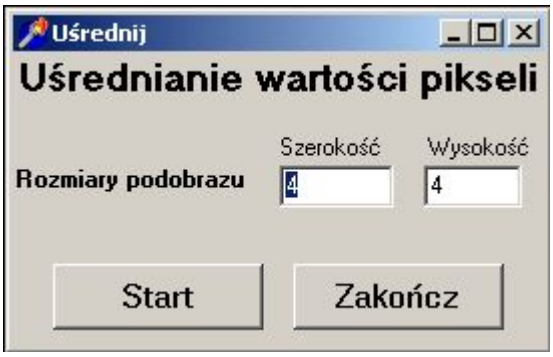
lub też w postaci wykresu histogramu



Rys.49. Histogram obrazu „Lena”.

UŚREDNIENIE

Każdy wczytany obraz można do celów testowych i poglądowych przetworzyć w ten sposób, że wybierając odpowiednie rozmiar w poziomie i w pionie dla wybranego podobrazu liczymy średnią arytmetyczną luminacji zawartych w nim pikseli, a następnie wszystkim przypisujemy tę wartość.

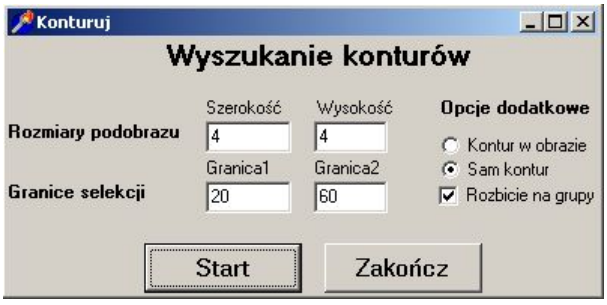


Rys.50. Panel uśredniania obrazu.

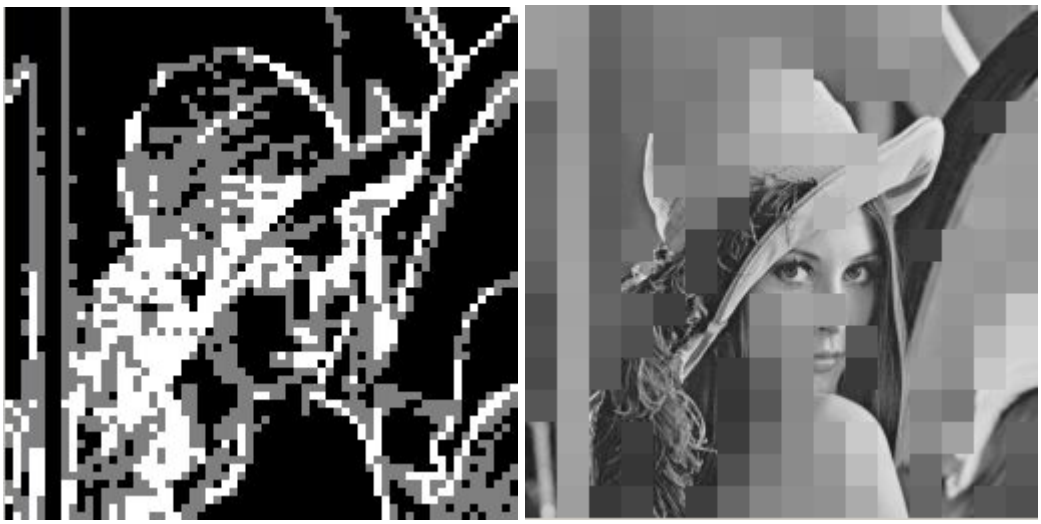
Rys.51. Wynik uśrednienie obrazu „Lena" dla podobrazów o rozmiarach 4x4.

KONTUROWANIE

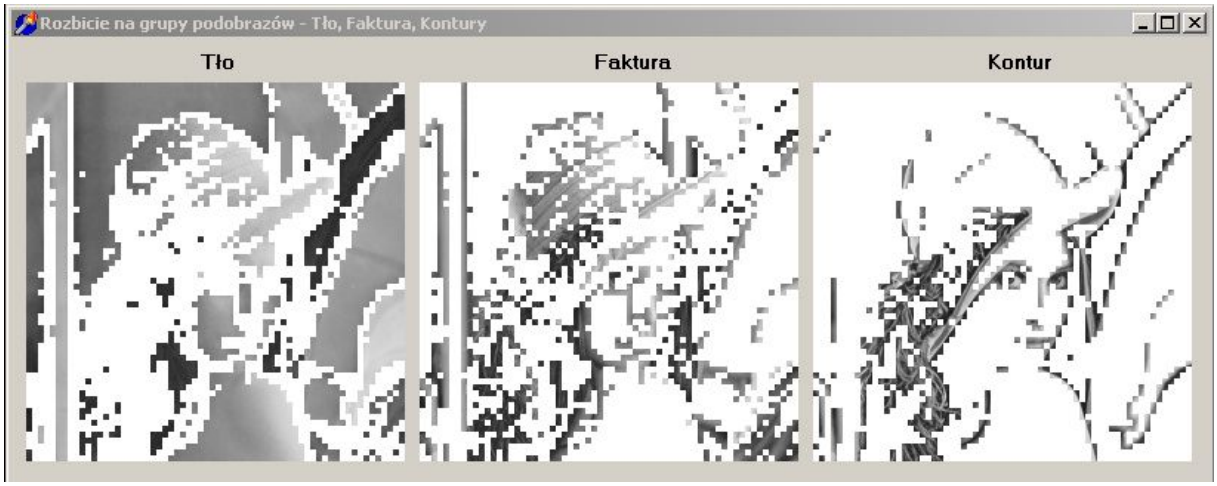
Obrazy naturalne zawierają w sobie elementy o różnej rozpiętości wartości luminacji, a tym samym o różnej podatności na kompresję. Każdy z wydzielonych podobrazów możemy zaklasyfikować do trzech podstawowych grup, a mianowicie do tła (o najmniejszej rozpiętości np. 0 - 30), faktury (o średniej rozpiętości np. 30 - 90) oraz do konturów (o największej rozpiętości 90 - 255).



Rys.52 Panel wyszukiwania konturów.



Rys.53. Wynik zastosowanie wyszukiwania konturów. Opcja „Sam kontur" dla podobrazów 4x4 (po lewej) oraz „Kontur w obrazie" dla 8x8 (po prawej).



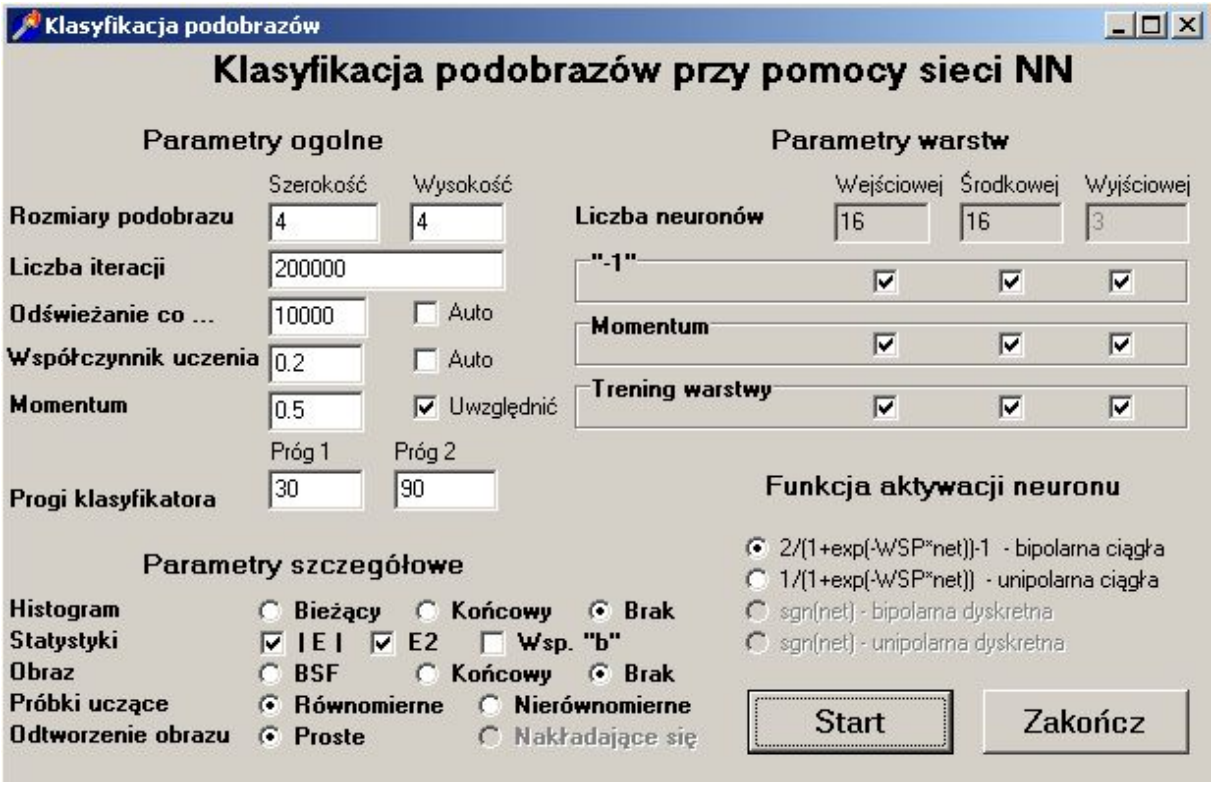
Rys.54. Wynik działania opcji „Rozbicie na grupy".

KONTUROWANIE NN

Wyszukiwanie konturów można również zrealizować wykorzystując sieci neuronowe. Jakkolwiek koszt przetwarzania jest zdecydowanie większy od obliczeń czysto numerycznych, to jednak pozwala na wyszukiwanie i grupowanie zdecydowanie bardziej złożone. Tutaj sieć uczy się rozpoznawać obrazy i bardziej odpowiednim zadaniem dla niej będzie rozpoznawanie obrazów z kamery umieszczonej przy taśmie w fabryce powideł, np. „jabłko", „gruszka" czy „śliwka" i sterowanie urządzeń do odpowiedniego ich posortowania.

Poniżej przedstawiony jest panel sterujący obsługujący tę funkcję. Na uwagę zasługuje tutaj parametr „Progi klasyfikatora", który jest odpowiedzialny za

odpowiednią rozpiętość wartości luminacji pikseli dla poszczególnych klas podobrazów. W poniższym przykładzie progi ustawione są odpowiednio na (0-30) dla tła, <30 – 90) dla faktury oraz <90 – 255) dla konturów.



Rys.55. Panel opcji „Klasyfikacja podobrazów" dla sieci neuronowej.

KOMPRESJA KLT

Jedną z grup metod kompresji obrazów nieruchomych są metody transformacyjne. Wśród nich najbardziej złożoną, lecz najlepszą pod względem minimalizacji błędu średniokwadratowego jest transformata Karhunena – Loeve. Najprościej opisać ją można jako zapis elementów powstałych poprzez mnożenie wejściowego wektora złożonego z kolumnowo zapisanych pikseli podobrazu przez macierz transformacyjną, którą jest macierz wektorów własnych estymaty macierzy kowariancji (liczonej dla wszystkich podobrazów) z wyzerowanymi wierszami odpowiadającymi najmniejszym, co do modułu wartościom własnym tejże macierzy. Dekompresja w tym wypadku jest procesem polegającym na wymnożeniu powstałych wektorów przez transponowaną macierz transformacyjną.

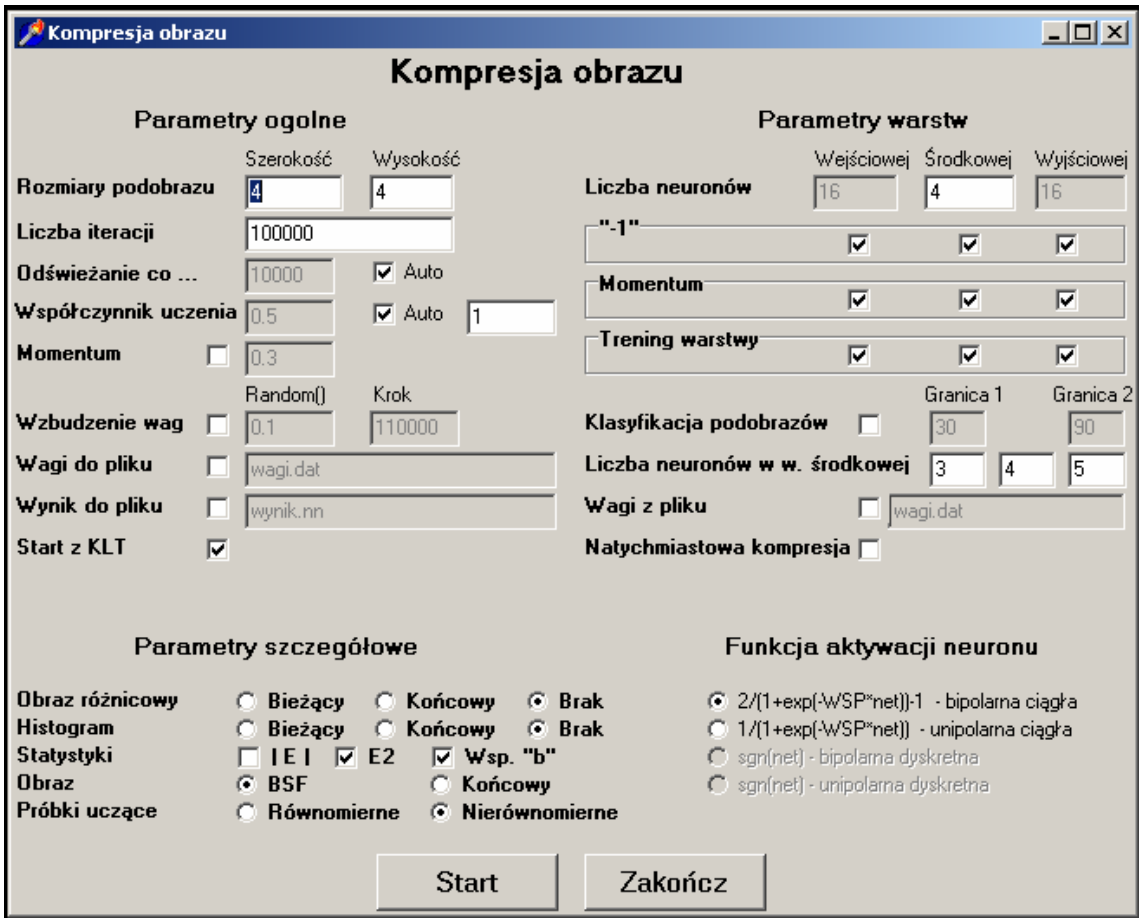
Procedura numeryczna realizująca obliczanie kolejnych największych, co do modułu wartości własnych oraz stowarzyszonych z nimi wektorów własnych została zrealizowana na bazie metody Hotellinga, która natomiast wykorzystuje metodę Rayleigh.

Rys.56. Panel sterujący „Kompresja KLT”.

Rys.57. Kompresja zrealizowana dla maksymalnych dopuszczalnych w programie ustawień, czyli 19x19 dla rozmiaru podobrazu i 128 elementach niezerowych.

KOMPRESJA NN

Kompresja za pomocą sieci neuronowej (NN) jest zasadniczym modułem tej aplikacji. Jest jednocześnie najbardziej złożonym zarówno funkcyjnie jak i programowo modułem.



Rys.58. Główny panel sterujący przetwarzaniem obrazu przez sieć neuronową.

Parametry ogólne

„Rozmiary podobrazu" – ustawienia rozmiarów przetwarzanego podobrazu w pikselach. Maksymalne możliwe do ustawienia to 48x48 jednak ilość obliczeń potrzebnych do nauczenia sieci w tym wypadku jest tak ogromna, że żaden ze współczesnych komputerów osobistych nie jest w stanie tego przetworzyć w rozsądnym czasie. Optymalne wartości maksymalne to 16x16.

„Liczba iteracji" – liczba kroków, jakie wykona sieć w celu nauczenia się obrazu. Dozwolone są wartości od 1 do 2^31 jednak rozsądne wyniki otrzymujemy już dla zakresu od 1 000 do 10 000 000 iteracji. (100 000 jest wartością optymalną).

„Odświeżanie, co" – opcja ta pozwala nam śledzić postępy w nauczaniu sieci wyświetlając, co zadaną liczbę iteracji aktualnie nauczony obraz. Dodatkowo podczas wyświetlania aktualizowane są statystyki na wykresach błędu średniokwadratowego oraz błędu bezwzględnego. Jakkolwiek jest to funkcja użyteczna i widowiskowa, to jednak znacznie spowalnia obliczenia. Opcja „Auto" ustawia automatycznie odświeżanie równe „Liczba iteracji"/100 i jest to wartość optymalna.

„Współczynnik uczenia" – Jest to współczynnik szybkości uczenia sieci. Wartość optymalna jest zależna od rozmiarów podobrazu, liczby iteracji, współczynnika momentu oraz liczby neuronów w warstwie środkowej. Parametr „Auto" jest standardowo ustawiony jako „zaznaczony".

„Momentum" – Parametr wspomagający uczenie sieci.

$$waga[i,j] := waga[i,j] + \Delta waga[i,j] + Momentum*(waga[i,j]- waga\_old[i,j])$$

„Wzbudzenie wag" – w celu uniknięcia zatrzymania się procesu uczenia w jakimś minimum lokalnym czasami stosuje się tzw. wzbudzanie wag. Pierwszy parametr określa o ile waga będzie wzbudzana (liczba losowa z przedziału od 0 do 1 razy wartość wpisana), a drugi informuje program, co ile iteracji ma wykonać wzbudzanie.

„Wagi do pliku" – zapis do pliku wag wyliczonych dla jednego obrazu wzorcowego w celu ich późniejszego wykorzystania.

„Wynik do pliku" – po przetworzeniu obrazu przez sieć do pliku tymczasowego zostają zapisane wszystkie niezbędne do odtworzenia parametry obrazu, wartości wag warstwy wyjściowej oraz wartości neuronów na wyjściu warstwy środkowej. Wszystkie wartości są przekształcane do postaci liczby całkowitej dwubajtowej ze znakiem. Następnie tworzony jest plik wsadowy rar.bat o zawartości

```
rar.exe m -m5 -y -c- -s -o+ wynik.nn temp.dat >> Err.log
```

który dodatkowo kompresuje wynik przetwarzania do pliku o nazwie podanej w odpowiednim polu.

Parametry warstw

„Liczba neuronów" – liczba neuronów w poszczególnych warstwach sieci. Dla warstwy wejściowej, jak i wyjściowej liczone są automatycznie na podstawie rozmiarów podobrazu. Natomiast dla warstwy środkowej podawane są arbitralnie przez użytkownika. Najlepiej jednak ustawiać wartości od 1/8 do 1/2 liczby wejść.

„-1" – parametr stabilizujący sieć osobno dla poszczególnych warstw.

„Momentum" – włączanie i wyłączanie efektu „momentu" dla poszczególnych warstw sieci.

„Trening warstwy" – informacja o tym, które warstwy podlegają treningowi.

„Klasyfikacja" oraz „Liczba neuronów w w. środkowej" – informacja o tym, czy w przetwarzaniu obrazu dokonać wstępnej analizy i klasyfikacji podobrazów, oraz użyć do tego celu trzech osobnych sieci neuronowych o różnych liczbach neuronów w warstwie środkowej dla różnych klas. W powyższym przykładzie progi ustawione

są odpowiednio na 30 i 90, a liczba neuronów w warstwie środkowej to 3 - tła, 4 – faktura, 5 – kontury.

„Wagi z pliku” – dobrze wytrenowaną sieć zapisaną w pliku możemy użyć do szybszej kompresji. Dodatkowo możemy tą sieć dostroić z mniejszym współczynnikiem uczenia, lecz bez potrzeby ponownego przetwarzania setek tysięcy iteracji.

„Natychmiastowa kompresja” – jeśli nie zachodzi potrzeba dostrajania sieci, to można poprzez włączenie tej opcji jedynie odczytać sieć z pliku i przetworzyć obraz natychmiast w pojedynczym kroku.

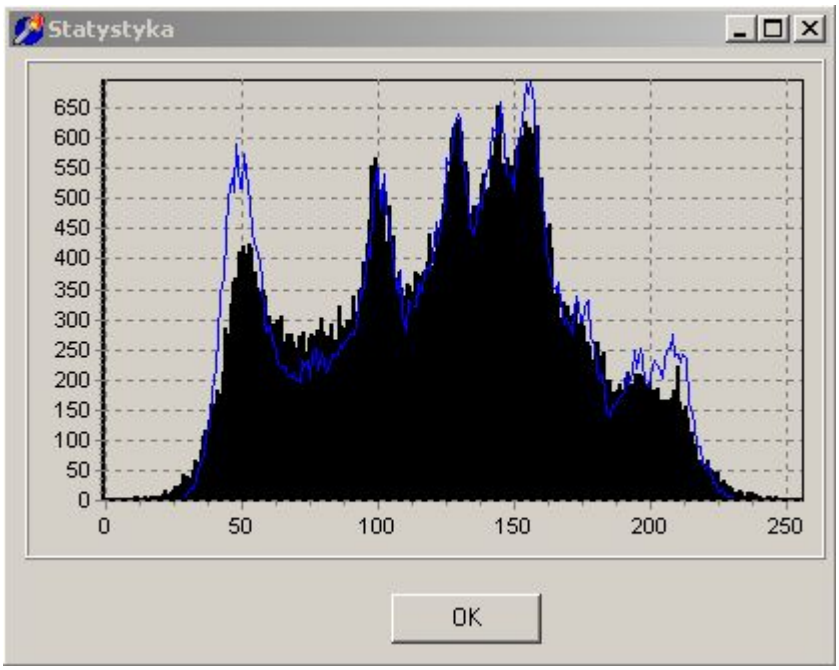
Parametry szczegółowe

„Obraz różnicowy” – czy i kiedy ma być wyświetlany obraz różnicowy przedstawiający różnicę pomiędzy obrazem przetworzonym, a oryginałem.

Rys.59. Obraz różnicowy.

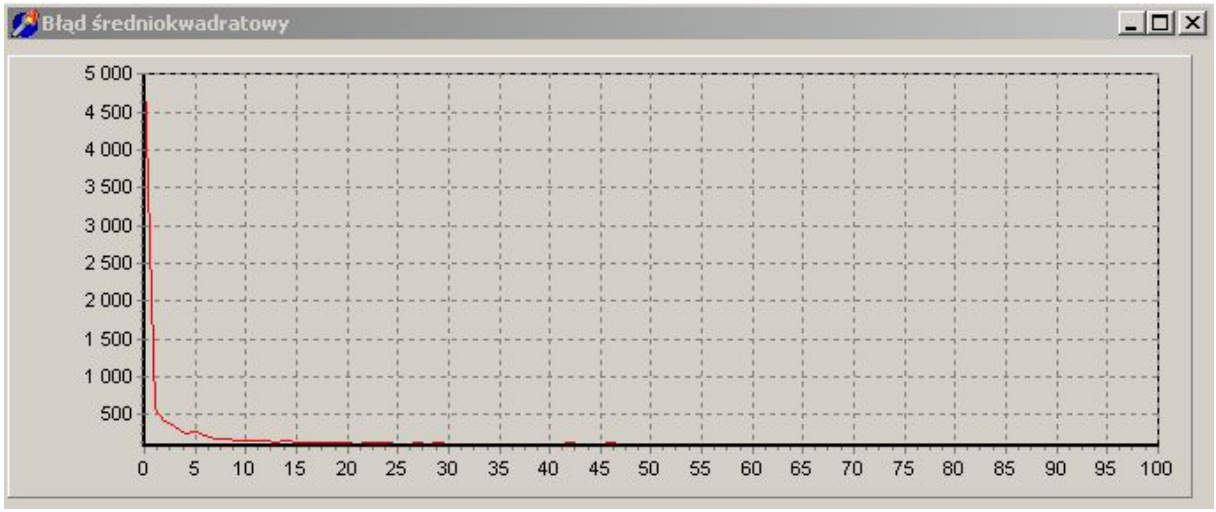
Każdy piksel przyjmuje wartość bezwzględną różnicy pomiędzy oryginałem, a po NN.

„Histogram” – czy i kiedy ma być wyświetlany histogram aktualnie przetwarzanego obrazu w porównaniu do oryginału

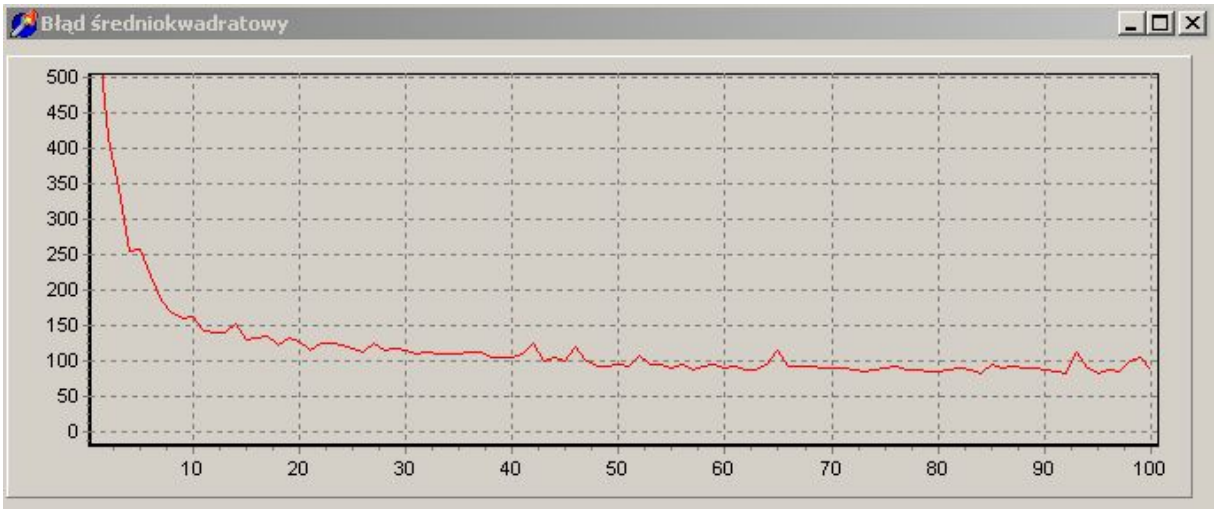


Rys.60. Histogram aktualnie przetwarzanego obrazu opisany kolorem czarnym w porównaniu do oryginału przedstawionego w kolorze niebieskim.

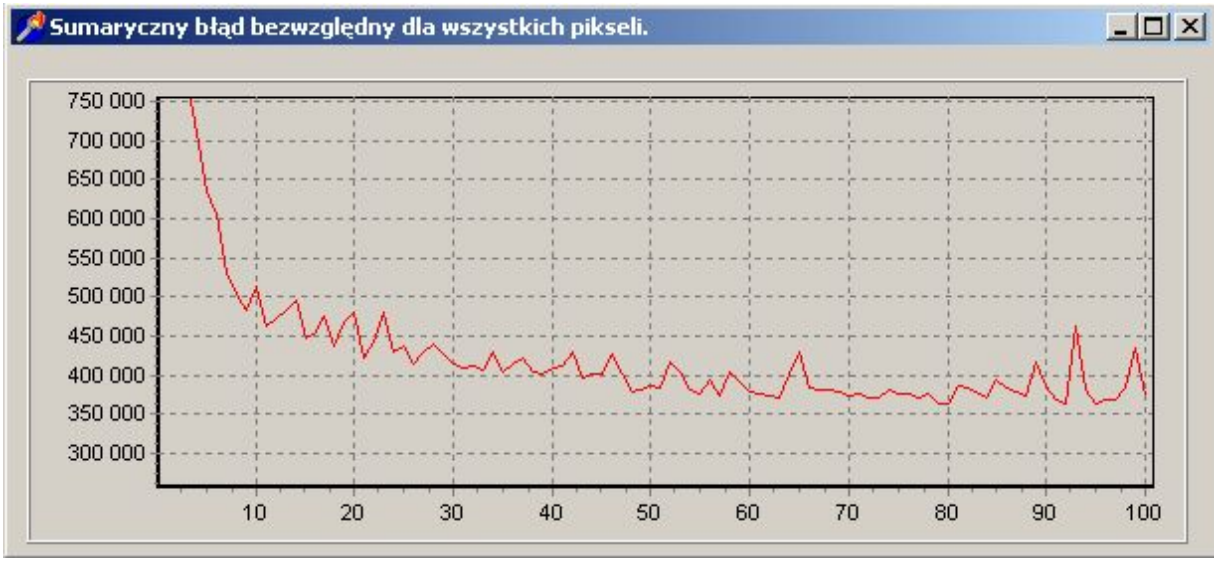
„Statystyki" – opcja stanowiąca o tym, jakie statystyki mają być zbierane podczas przetwarzania i udostępnione po jego zakończeniu. Mamy tutaj wykresy zmiany błędu średniokwadratowego oraz bezwzględnego, a także zmiany współczynnika uczenia podczas całego procesu.



Rys.61. Wykres zmian błędu średniokwadratowego w zależności od liczby kroków uczących podzielonych na sto etapów.



Rys.62. Istnieje możliwość dynamicznego powiększania fragmentów statystyki z rys. 61, czego wynikiem jest wykres z widocznymi detalami procesu uczenia.



Rys.63. Wykres zmian sumarycznego błędu bezwzględnego dla całego obrazu.

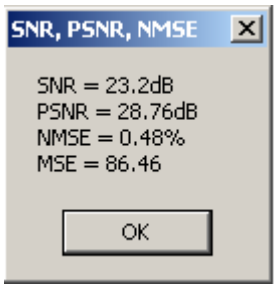
„Obraz" – jeśli zaznaczymy BSF (Best So Far), co jest zalecane to po zakończeniu nauczania wyświetlony i ew. zapisany zostanie najlepszy spośród tych obrazów jakie podczas całego procesu były wyświetlane jako aktualne. W przypadku wybrania drugiej opcji zostanie wyświetlony obraz, który faktycznie pojawił się na wyjściu sieci jako ostatni.

„Próbki uczące" – Istnieje możliwość podawania na wejściu podobrazów o ściśle określonym położeniu (odpowiadającym późniejszemu zakodowaniu i odtworzeniu), jak również o dowolnym położeniu w całym obrazie. Druga opcja jest zalecana przy większych rozmiarach podobrazów.

Funkcja aktywacji neuronu

Zaimplementowane zostały dwie ciągłe funkcje aktywacji (bipolarna i unipolarna). Jednocześnie w kodzie źródłowym jest miejsce na pozostałe dwie funkcje dyskretne, jednak ze względu na ograniczoność zakresu generowanych przez nie danych na wyjściu neuronu nie znajdują one tutaj zastosowania. Dodatkowo można w to miejsce dodać inne funkcje aktywacji, które spełnią odpowiednio swoje zadanie.

Dodatkowo w podręcznym menu pokazującym się w momencie najechania na obrazek kursorem myszy i przyciśnięciu prawego przycisku pojawią się opcje zapisu do pliku oraz statystyk. Zapis do pliku przebiega podobnie do funkcji „Zapisz jako", którą możemy wybrać z menu „Plik" w głównym menu, natomiast wybranie statystyk powoduje wyświetlenie histogramu oraz okienka z wyliczonymi parametrami NMSE (średni błąd), MSE (średni błąd przypadający na jeden piksel), SNR (odstęp sygnału od szumu) oraz PSNR (chwilowy odstęp sygnału od szumu) dla przetworzonego obrazu.

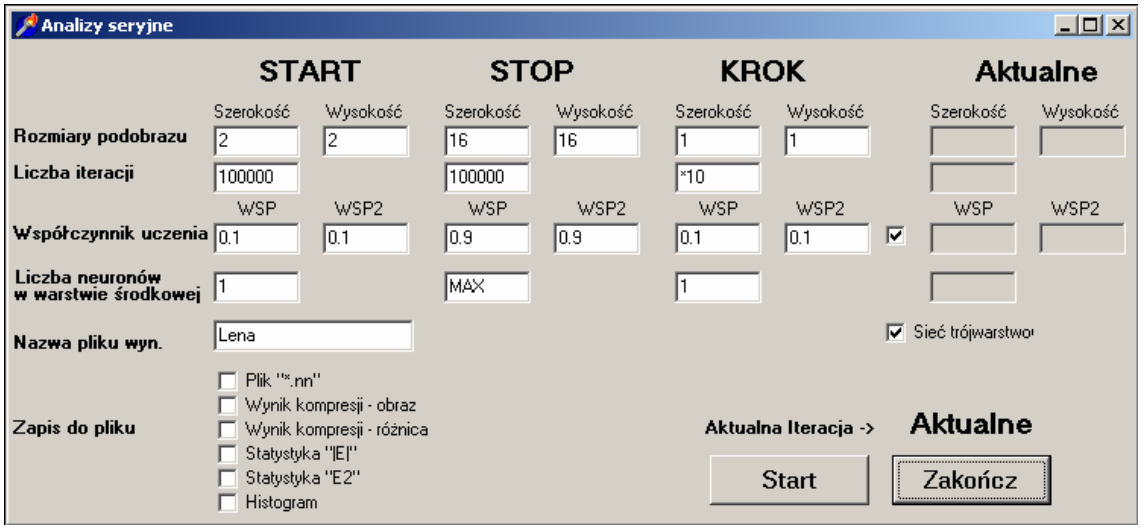


Rys.64. Dane statystyczne dla obrazu przykładowego.

Rys.65. Obraz wynikowy dla przykładowych danych przedstawionych powyżej.

ANALIZY SERYJNE

W związku z długim czasem przetwarzania dla każdego z ustawień i konieczności przeprowadzenia badań nad zachowaniem się sieci oraz możliwościami kompresji obrazów przez sieć został wprowadzony moduł analiz seryjnych, który generuje wyniki z zakresów ustawionych na formularzu



Rys.66. Panel sterowania analizami seryjnymi.

Znaczenie poszczególnych opcji jest takie same jak na formularzu „Kompresji NN". Dodatkowo parametry zostały podzielone na grupy. Ustawienia wartości początkowych START, końcowych STOP oraz KROK, o jaki wartości początkowe mają się zwiększać i zbliżać do STOP.

Aktualne – aktualne wartości parametrów, dla których trwają obliczenia.

Nazwa pliku wyn. – wyniki tekstowe analiz umieszczone zostaną w pliku o nazwie Lena_analizy.txt.

W lewej dolnej części formularza można ustawić, jakiego typu pliki będą zapisywane na dysku dla każdego przetworzenia. Natomiast za napisem „Aktualna iteracja" w miejscu tekstu „Aktualne" będzie wyświetlany postęp w aktualnym przetwarzaniu.

Uwaga na liczbę możliwych przetwarzań i czas potrzebny na ich realizację!

POMOC

O PROGRAMIE

Krótka notka o programie i jego autorze, czyli o mnie.

POMOC

Opcja do przyszłej implementacji

OD AUTORA

Mam nadzieję, że narzędzie, jakie stworzyłem pozwoli przyszłym użytkownikom lepiej zrozumieć zasadę działania sieci neuronowych oraz drzemiące w nich możliwości. Życzę Wam dużo ciekawych doświadczeń i cierpliwości dla swoich domowych komputerów oraz zapału i pomysłów w wykorzystaniu sztucznej inteligencji, a w szczególności sieci neuronowych do nowych zastosowań i badań.

Zakończenie

Symulacja zachowania rzeczywistej sieci neuronowej nie jest zagadnieniem trywialnym. Wymaga ona dużo skupienia i zwracania uwagi na wszystkie szczegóły. Wystarczy jeden drobny błąd, aby cała sieć zachowywała się bardzo niestabilnie i nie spełniała swojego zadania. W tej pracy jej zadaniem była kompresja obrazów nieruchomych. Do tego celu stworzony został program „Kompresja Neuronowa". Pozwala on na dość szczegółowe badanie zachowań sieci o różnych parametrach początkowych i uczenia. Możliwości programu oraz zakres możliwych testów znacznie wykraczają poza ramy tej pracy, a materiału wystarczy na 500 stronicową książkę, dlatego też ograniczono się tylko do najważniejszych cech, a i to tylko odkrywając „wierzchołek góry lodowej". Przetestowano i opisano następujące rzeczy:

- Jakość kompresji w zależności od liczby neuronów warstwy ukrytej.
- Jakość kompresji dla różnych rozmiarów podobrazu.
- Jakość kompresji dla różnych typów obrazów.
- Skuteczność kompresji NN w porównaniu do KLT dla różnych rozmiarów podobrazów (2x2, 4x4, 6x6, 8x8).
- Przypadek obrazu „Tony", w którym sieć neuronowa jest nawet kilkaset razy lepsza od KLT.
- Wpływ klasyfikacji podobrazów na końcowy efekt kompresji.
- Wpływ wartości współczynnika uczenia na jakość i szybkość kompresji.
- Wpływ liczby kroków uczenia oraz wartości czynnika momentum na wartość błędu średniokwadratowego.
- **Wpływ przybliżenia początkowych wartości wag na podstawie macierzy transformacyjnej KLT dla sieci dwu- i trójwarstwowych.**
- Zachowanie się sieci dwu- i trójwarstwowej.
- Możliwości przetwarzania różnych obrazów za pomocą sieci wyuczonej na jednym wzorcu.
- Stopień kompresji i rozmiar pliku wynikowego dla różnych typów obrazów w zależności od wartości błędu średniokwadratowego.
- Szybkość uczenia sieci oraz jej zależność od liczby iteracji, struktury sieci i częstotliwości zbierania wyników.

Większość z powyższych badań przeprowadzono dla popularnego obrazu „Lena" z podziałem na podobrazy o rozmiarach 4x4 piksele. Z zachowania się sieci w testach dla różnych wielkości podobrazów można wnioskować, że wyniki otrzymane dla tego szczególnego przypadku można uogólnić na pozostałe. Niestety objętość tej pracy nie pozwala na opisanie wszystkich przypadków, choć byłoby to całkiem ciekawe i pozwoliło dokładnie poznać problem, a tym samym wyznaczyć konkretne kierunki dalszego rozwoju i konkretnych zastosowań.

Dodatkowo zakres działania programu można w prosty sposób rozbudować o możliwość przetwarzania obrazów kolorowych analizując poszczególne składowe kolorów w taki sam sposób jak obrazy z odcieniami szarości, a następnie łącząc je w jeden kolorowy obraz.

W tej fazie badania i opisu problemu pliki wynikowe nie porażają stopniem kompresji i ogólnie rzecz biorąc są zdecydowanie mniejsze od popularnego formatu JPG[7], który zdominował świat grafiki. Jeśli jednak wziąć pod uwagę, że na jednym z etapów algorytmu zastosowanego w tymże formacie jest DCT[8], która jest gorsza pod względem minimalizacji błędu średniokwadratowego od najlepszej w klasie przekształceń transformacyjnych KLT. Natomiast ta ostatnia jak widać z przeprowadzonych testów jest nawet trzykrotnie gorsza od tego, co oferuje sieć neuronowa. Idąc tym tropem można pokusić się o zastąpienie DCT przez NN, co poprawi znacznie jakość obrazów zapisywanych w formacie JPG, lub przy tej samej jakości pozwoli na mocniejszą kompresję. Jedynym mankamentem jest tutaj zapotrzebowanie na ogromną moc procesora potrzebną do wytrenowania sieci i relatywnie długi czas jej realizacji. Problem ten może być tematem kolejnej pracy polegającej na implementacji i testowaniu różnych technik optymalizacji procesu uczenia, ze szczególnym uwzględnieniem metod takich jak BFGS[9], gradientów sprzężonych z regularyzacją Møllera, czy też Levenberga-Marquardta, które uważane są za najskuteczniejsze i potrafią przyspieszyć proces uczenia nawet kilkaset razy, co w przypadku dzisiejszego Pentium IV 2.4GHz oznaczałoby skrócenie czasu uczenia do ułamków sekundy, a tym samym skutecznie pozwoliłoby na jej zastosowanie w nowym algorytmie JPG.

[7] JPG – Inaczej JPEG (Joint Photographic Experts Group).
[8] DCT – Discrete Cosine Transform (Dyskretna Transformata Cosinusów)
[9] BFGS - Broyden-Goldfarb-Fletcher-Shanno

Jest to oczywiście tylko jedno z możliwych zastosowań sieci neuronowej, a przeprowadzone badania dowiodły jej wysokiej skuteczności, co w znacznej mierze może przyczynić się do rozwoju technik kompresji obrazu. Natomiast sama istota budowy i działania sieci neuronowej zbliża nas do zrozumienia zasad działania ludzkiego mózgu, a tym samym istoty naszego bytu...

Umysły niecierpliwe za stracony uważają czas użyty pożytecznie, to jest na badanie dzieł natury i spraw ludzkich.

Leonardo da Vinci

Bibliografia

I Opracowania książkowe

1. Andrzej Kiełbasiński, Hubert Schwetlick, „Numeryczna Algebra Liniowa – Wprowadzenie do obliczeń zautomatyzowanych", Wydawnictwa Naukowo Techniczne, Warszawa 1992.
2. Christopher D. Watkins, Alberto Sadun, Stephen Marenka, „Nowoczesne metody przetwarzania obrazu", Wydawnictwa Naukowo-Techniczne, Warszawa 1995.
3. Intersoftland, „Grafika PC bez tajemnic", Tłumaczenie z j. ang. przez Jarosław Mirkowski, Intersoftland, Warszawa 1995.
4. Jacek M. Żurada, „Introduction to Artificial Neural Systems", West Publishing Company, USA 1992.
5. J. Żurada, M. Barski, W. Jędruch, „Sztuczne sieci neuronowe – Podstawy teorii i zastosowania", Wydawnictwo Naukowe PWN, Warszawa 1996.
6. Joanna Chromiec, Edyta Strzemieczna, „Sztuczna inteligencja – Metody konstrukcji i analizy systemów eksperckich", Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994.
7. John Hertz, Anders Krogh, Richard G. Palmer, „Wstęp do teorii obliczeń neuronowych", Wydawnictwa Naukowo-Techniczne, Warszawa 1993.
8. Kazimierz Wanat, „Algorytmy numeryczne – wybór metod numerycznych", Wydawnictwo DIR, Gliwice 1993.
9. Ryszard Tadeusiewicz, „Sieci neuronowe", Akademicka Oficyna Wydawnicza RM, Warszawa 1993.
10. Stanisław Osowski, „Sieci neuronowe w ujęciu algorytmicznym", Wydawnictwa Naukowo-Techniczne, Warszawa 1996.
11. Steve Teixeira, Xavier Pacheco, „Delphi 4 – Vademecum Profesjonalisty" Tom 1 i 2, Wydawnictwo Helion, Gliwice 1999.
12. W.S. McCulloch, W.H. Pitts, „A logical calculus of the ideas imminent in nervous activity, Bull. Math. Biophysics, 1943.
13. D.O. Hebb, "The organization of Behaviour, a Neuropsychological Theory", Wiley, New York 1949.
14. M. Minsky, S. Papert, "Perceptrons", MIT Press, Cambridge Mass. 1969.

II Czasopisma

15. Cykl artykułów zamieszczonych w czasopiśmie „Enter", Warszawa 199x.

III Źródła internetowe

16. „Sieci neuropodobne", http://brain.fuw.edu.pl/jarek/sieci.html
17. Krzysztof Sierota, „Sieci neuronowe", Laboratorium z Sieci Neuronowych, http://akson.sgh.waw.pl/~ksiero/NN
18. „Sieci neuronowe", http://republika.pl/edward_ch/pod_prakt.html
19. Backpropagation (Neural Network Toolbox), http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/backpr13.shtml
20. "The rest of the Lenna story", http://imagets10.univ.trieste.it/lena/lenna.html

IV Inne opracowania

21. Hanna Pempera, praca dyplomowa pt. „Analiza metod kompresji obrazów nieruchomych za pomocą transformacji Karhunena – Loeve", Politechnika Gdańska 1992.
22. Roman Rykaczewski, „Kompresja obrazów nieruchomych za pomocą sztucznej sieci neuronowej", str. 447, Krajowe Sympozjum Telekomunikacji, ATR w Bydgoszczy, 6-8 września 1995.
23. Martin F. Møller, „A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Computer Science Department University of Aarhus, Denmark 13 November 1990.
24. M. Barski, W. Jędruch, wykłady na temat „Sztuczne sieci neuronowe", Politechnika Gdańska 1995.
25. Yann Le Cun, Ido Kanter, Sara A.Solla, „Eigenvalues of Covariance Matrices: Application to Neural-Network Learning", vol.66, number 18 of *Physical Review Letters,* The American Physical Society 6 May 1991.
26. H.M. Abbas, M.M. Fahmy, PhD, PEng, "Neural model for Karhunen-Loeve transform with application to adaptive image compression", IEE PROCEEDINGS-I, Vol. 140, No. 2, April 1993.
27. Franklin G. Horowitz, Don Bone, Paul Veldkamp, "Karhunen-Loeve Based Iterated Function Systems Encodings", International Picture Coding Symposium, Melbourne, Australia 13-15 March 1996.

28. "Kompresja bezstratna – Materiały pomocnicze do wykładu *Kompresja danych*", Politechnika Wrocławska 2003.
29. Roman Rykaczewski, „Kompresja obrazu wykorzystująca transformację unitarną – transformację Karhunena-Loeve", Prezentacja w programie MathCAD, Politechnika Gdańska 19xx.
30. Roman Rykaczewski, „Nowoczesne Metody Kompresji Obrazów", Katedra systemów Informacyjnych Politechniki Gdańskiej, 19xx.
31. Stanisław Osowski, Piotr Bojarczyk, Maciej Stodolski, „Fast second order learning algorithm for feedforward multilayer neural networks and its applications", Warsaw University of Technology, 28 February 1996.
32. Yann LeCun, Patrice Y.Simard, Barak Pearlmutter, "Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors", AT&T Bell Laboratories and CS&E Dept. Oregon Grad. Inst.
33. Andrea Basso, Murat Kunt, "Autoassociative Neural Networks for Image Compression", Ecole Polytechnique Federale de Lausanne, vol.3, No.6, Switzerland Nov.-Dec. 1992.
34. Robert D. Dony, Simon Haykin, "Neural Network Approaches to Image Compression", Proceedings of the IEEE, Vol. 83, No. 2, February 1995.
35. J.Jiang, "Image compression with neural networks – A survey", School of Computing, University of Glamorgan, UK 24 April 1996.
36. B.M. Wilamowski, "Neural Network Design", Prezentacja w programie PowerPoint. 15.10.2001
37. Martin Riedmiller, „RPROP – Description and Implementation Details", Institute für Logik, University of Karlsruhe, January 1994.
38. Valérie Frayssé, Luc Giraud, "A Set of Conjugate Gradient Routines for Real and Complex Arithmetics", CERFACS Technical Report TR/PA/00/47, France, July 2000.
39. G. Martinelli, L. Prina Ricotti, G. Marcone, "Neural Clustering for Optimal KLT Image Compression", IEEE Transactions on Signal Processing, Vol.41, No.4 April 1993.