Kamil Stokfiszewski

NEURONOWE REALIZACJE SZYBKICH ZUNIFIKOWANYCH ALGORYTMÓW OBLICZANIA PRZEKSZTAŁCEŃ DYSKRETNYCH W KOMPRESJI OBRAZÓW

Monografie Politechniki Łódzkiej
Łódź 2021

Recenzenci:
prof. dr hab. inż. Marek Kurzyński
prof. dr. hab. Michał Jacymirski, prof. nadzw. UŁ

ISBN 978-83-66741-31-7
DOI: 10.34658/9788366741317

Wydawnictwo Politechniki Łódzkiej
93-005 Łódź, ul. Wólczańska 223
Tel. 42-631-20-87, 42-631-29-52
E-mail: zamowienia@info.p.lodz.pl
www.wydawnictwo.p.lodz.pl

Monografie Politechniki Łódzkiej, Nr 2389

Książkę dedykuję Tacie

SPIS TREŚCI

WYKAZ OZNACZEŃ

$\forall$ – kwantyfikator ogólny „dla każdego”
$\exists$ – kwantyfikator szczegółowy „istnieje”
$\wedge$ – operator logiczny koniunkcji
$\vee$ – operator logiczny alternatywy
$\Rightarrow$ – operator logiczny implikacji
$\Leftrightarrow$ – operator logiczny równoważności
$\cup$ – operator mnogościowy sumy zbiorów
$\cap$ – operator mnogościowy iloczynu zbiorów
$\setminus$ – operator mnogościowy różnicy zbiorów
$\mathbb{N}$ – zbiór liczb naturalnych
$\mathbb{R}$ – zbiór liczb rzeczywistych
$\mathbb{R}_+$ – zbiór liczb rzeczywistych dodatnich
$\mathbb{Z}$ – zbiór liczb całkowitych
$\mathbb{C}$ – zbiór liczb zespolonych
$\mathbb{R}^N$ – zbiór N–elementowych wektorów rzeczywistych
$\mathbb{C}^N$ – zbiór N–elementowych wektorów zespolonych
$\mathbb{R}^{M\times N}$ – zbiór $M \times N$–elementowych macierzy rzeczywistych
$\mathbb{C}^{M\times N}$ – zbiór $M \times N$–elementowych macierzy zespolonych
X, Y – zmienne losowe
$X_{m,n}$, $\xi_{m,n}$, ξ_n – zmienne losowe pól lub procesów stochastycznych
$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ – wektory rzeczywiste lub wektory rzeczywistych zmiennych losowych
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ – macierze rzeczywiste lub macierze rzeczywistych zmiennych losowych
$E\{X\}$ – wartość oczekiwana zmiennej losowej X
$\rho\{X, Y\}$ – unormowany współczynnik korelacji pary zmiennych losowych X i Y
σ_X^2 – wariancja zmiennej losowej lub pola losowego X
$E\{\mathbf{x}\}$ – wartość oczekiwana wektora rzeczywistych zmiennych losowych $\mathbf{x}$
$E\{\mathbf{x}\mathbf{x}^T\}$ – macierz autokowariancji wektora rzeczywistych zmiennych losowych $\mathbf{x}$
$\rho\{\mathbf{x}\mathbf{x}^T\}$ – macierz autokorelacji wektora rzeczywistych zmiennych losowych $\mathbf{x}$
σ^2 – wariancja zmiennej stacjonarnego pola losowego
ρ_r – współczynnik autokorelacji wierszowej separowalnego pola losowego
ρ_c – współczynnik autokorelacji kolumnowej separowalnego pola losowego
$\mathcal{N}(\mu, \sigma^2)$ – rozkład normalny o wartości oczekiwanej μ i wariancji σ^2
$X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ – zmienna losowa X o rozkładzie normalnym z wartością oczekiwaną μ_X i wariancją σ_X^2
$\delta_{m,n}$ – delta Kroneckera dla zmiennych m i n

$\mathbf{a}^T$, $\mathbf{A}^T$ – operator transpozycji wektora i/lub macierzy

$tr\,[\,\mathbf{A}\,]$ – operator śladu rzeczywistej macierzy $\mathbf{A}$

$vec(\mathbf{A})$ – operator wektoryzacji kolumnowej macierzy $\mathbf{A}$

$vec^{-1}_{(m)}(\mathbf{v})$ – operator dewektoryzacji rzeczywistego lub losowego wektora $\mathbf{v}$

$\mathbf{A} \otimes \mathbf{B}$ – produkt Kroneckera macierzy rzeczywistych/zespolonych lub losowych

$diag\,(a_1, \ldots, a_N)$ – kwadratowa macierz diagonalna o zadanych elementach na głównej przekątnej

$\mathbf{K}$ – macierz autokowariancji sygnału wejściowego procesu kompresji

$\mathbf{U}$ – macierz przekształcenia kodującego procesu kompresji obrazu

$\mathbf{V}$ – macierz przekształcenia dekodującego procesu kompresji obrazu

$\mathbf{I}$ – macierz jednostkowa

$\mathbf{I}_m$ – macierz obcinająca procesu kompresji obrazu

$\bar{e}_m\,(\,\mathbf{U}, \mathbf{V}\,)$ – błąd średniokwadratowy (skalowany) dla procesu kompresji obrazu

$\lambda_1, \ldots, \lambda_N$ – uporządkowane malejąco wartości własne macierzy autokowariancji $\mathbf{K}$ sygnału wejściowego procesu kompresji

$\mathbf{W}$ – macierz jednowymiarowego dyskretnego przekształcenia kosinusowego

$\mathbb{W}$ – macierz dwuwymiarowego dyskretnego przekształcenia kosinusowego

$\mathcal{O}(N)$ – ogólna złożoność liniowa

$\mathcal{O}(N^2)$ – ogólna złożoność kwadratowa

$\mathcal{O}(Nlog_2N)$ – ogólna złożoność liniowo - logarytmiczna

$E\,(\,\mathbf{w}\,)$ – funkcja błędu dla procesu treningowego perceptronu wielowarstwowego

$z_{i\mu}$ – pożądane i-te wyjście μ-tego wzorca treningowego perceptronu

$w_{ij}^{(k)}$ – j-ta waga i-tego neuronu k-tej warstwy sieci

$v_{i\mu}^{(k)}$ – wyjście i-tego neuronu k-tej warstwy sieci dla μ-tego wzorca treningowego

$\delta_{i\mu}^{(k)}$ – sygnał zwrotny i-tego neuronu k-tej warstwy sieci dla μ-tego wzorca uczącego

$\overrightarrow{\mathcal{P}}^{+}_{\mathcal{W}}, \overrightarrow{\mathcal{P}}^{*}_{\mathcal{W}}$ – liczba dodawań/mnożeń dla propagacji prostej w sieci neuronowej

$\overleftarrow{\mathcal{P}}^{+}_{\mathcal{W}}, \overleftarrow{\mathcal{P}}^{*}_{\mathcal{W}}$ – liczba dodawań/mnożeń jednego kroku wstecznej propagacji błędu

C_N – N-punktowa jednowymiarowa dyskretna transformata kosinusowa

$\overline{C}_N$ – N-punktowa skalowana jednowymiarowa dyskretna transformata kosinusowa

$\widetilde{C}_N$ – N-punktowa zunifikowana jednowymiarowa dyskretna transformata kosinusowa

$\overline{C}_{N\times N}$ – $N \times N$-punktowa skalowana dwuwymiarowa dyskretna transformata kosinusowa

$\widetilde{\mathbf{C}}_{N\times N}$ – $N\times N$-punktowa zunifikowana dwuwymiarowa dyskretna transformata kosinusowa

$\mathcal{C}^{+}_{N}, \mathcal{C}^{*}_{N}$ – liczba dodawań/mnożeń dla szybkiego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia kosinusowego

$\widetilde{\mathcal{C}}^{+}_{N}, \widetilde{\mathcal{C}}^{*}_{N}$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia kosinusowego

$\widetilde{\mathcal{C}}^{+}_{N\times N}, \widetilde{\mathcal{C}}^{*}_{N\times N}$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania dwuwymiarowego dyskretnego przekształcenia kosinusowego

S_N – N-punktowa jednowymiarowa dyskretna transformata sinusowa

$\overline{S}_N$ – N-punktowa skalowana jednowymiarowa dyskretna transformata sinusowa

$\widetilde{S}_N$ – N-punktowa zunifikowana jednowymiarowa dyskretna transformata sinusowa

$\overline{S}_{N\times N}$ – $N \times N$-punktowa skalowana dwuwymiarowa dyskretna transformata sinusowa

$\widetilde{\mathbf{S}}_{N\times N}$ – $N\times N$-punktowa zunifikowana dwuwymiarowa dyskretna transformata sinusowa

$\mathcal{S}_N^+, \mathcal{S}_N^*$ – liczba dodawań/mnożeń dla szybkiego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia sinusowego

$\widetilde{\mathcal{S}}_N^+, \widetilde{\mathcal{S}}_N^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia sinusowego

$\widetilde{\mathcal{S}}_{N\times N}^+, \widetilde{\mathcal{S}}_{N\times N}^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania dwuwymiarowego dyskretnego przekształcenia sinusowego

H_N – N-punktowa jednowymiarowa dyskretna transformata Hartleya

$\overline{H}_N$ – N-punktowa skalowana jednowymiarowa dyskretna transformata Hartleya

$\widetilde{H}_N$ – N-punktowa zunifikowana jednowymiarowa dyskretna transformata Hartleya

$\overline{H}_{N\times N}$ – $N \times N$-punktowa skalowana dwuwymiarowa dyskretna transformata Hartleya

$\widetilde{\mathbf{H}}_{N\times N}$ – $N\times N$-punktowa zunifikowana dwuwymiarowa dyskretna transformata Hartleya

$\mathcal{H}_N^+, \mathcal{H}_N^*$ – liczba dodawań/mnożeń dla szybkiego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Hartleya

$\widetilde{\mathcal{H}}_N^+, \widetilde{\mathcal{H}}_N^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Hartleya

$\widetilde{\mathcal{H}}_{N\times N}^+, \widetilde{\mathcal{H}}_{N\times N}^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania dwuwymiarowego dyskretnego przekształcenia Hartleya

F_N – N-punktowa jednowymiarowa dyskretna transformata Fouriera

$\overline{F}_N$ – N-punktowa skalowana jednowymiarowa dyskretna transformata Fouriera

$\widetilde{F}_N$ – N-punktowa zunifikowana jednowymiarowa dyskretna transformata Fouriera

$\overline{F}_{N\times N}$ – $N \times N$-punktowa skalowana dwuwymiarowa dyskretna transformata Fouriera

$\widetilde{\mathbf{F}}_{N\times N}$ – $N\times N$-punktowa zunifikowana dwuwymiarowa dyskretna transformata Fouriera

$\mathcal{F}_N^+, \mathcal{F}_N^*$ – liczba dodawań/mnożeń dla szybkiego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Fouriera

$\widetilde{\mathcal{F}}_N^+, \widetilde{\mathcal{F}}_N^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Fouriera

$\widetilde{\mathcal{F}}_{N\times N}^+, \widetilde{\mathcal{F}}_{N\times N}^*$ – liczba dodawań/mnożeń dla zunifikowanego algorytmu wyznaczania dwuwymiarowego dyskretnego przekształcenia Fouriera

$\mathcal{W}$ – całkowita liczba wag sieci wielowarstwowej z pominięciem wag warstwy kopiującej

$\mathcal{K}$ – całkowita liczba neuronów sieci wielowarstwowej z pominięciem neuronów warstwy kopiującej

$\mathcal{W}_f$ – liczność wag sieci w warstwie o indeksie równym jeden

$\mathcal{K}_l$ – liczność neuronów ostatniej warstwy sieci

$e_\mu\ (\mathcal{X})$ – błąd estymacji wektora oczekiwanego zbioru próbek $\mathcal{X}$ dla symulatora pól losowych Gaussa-Markowa

$e_{\mathbf{K}}(\mathcal{X})$ – błąd estymacji macierzy autokowariancji zbioru próbek $\mathcal{X}$ dla symulatora pól losowych Gaussa-Markowa

$e_{\mathcal{N}}(\mathcal{X})$ – błąd estymacji rozkładu normalnego dla zbioru próbek $\mathcal{X}$ dla symulatora pól losowych Gaussa-Markowa

$\prod_{n=1}^{N} \mathbf{X}_n$ – iloczyn lewostronny macierzy $\mathbf{X}_n$

$\mathbf{X}_n \prod_{n=1}^{N}$ – iloczyn prawostronny macierzy $\mathbf{X}_n$

$\overline{\mathbf{I}},\quad \widetilde{\mathbf{I}}$ – pomocnicze macierze jednostkowe dwuwymiarowego algorytmu zunifikowanego

$\overline{\mathbf{Q}},\ \widetilde{\mathbf{Q}}$ – pomocnicze macierze diagonalne dwuwymiarowego algorytmu zunifikowanego

$\mathbf{R},\ \mathbf{Z}$ – macierze pomocnicze definiujące algorytm zunifikowany wyznaczania dwuwymiarowego dyskretnego przekształcenia kosinusowego

$\mathbf{P}_n$ – macierze wyjściowych permutacji przekształceń jednowymiarowych dla kolejnych etapów algorytmu `JFCT2D`

$\mathbf{C}_n$ – macierze kolejnych etapów zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia kosinusowego

$\mathbf{S}_n$ – macierze kolejnych etapów zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia sinusowego

$\mathbf{H}_n$ – macierze kolejnych etapów zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Hartleya

$\mathbf{F}_n$ – macierze kolejnych etapów zunifikowanego algorytmu wyznaczania jednowymiarowego dyskretnego przekształcenia Fouriera

$\mathbf{P}_S$ – macierz permutacji wyjściowej dla zunifikowanego algorytmu wyznaczania dwuwymiar. dyskretnego przekształcenia sinusowego

$\mathbf{P}_H$ – macierz permutacji wejściowej dla zunifikowanego algorytmu wyznaczania dwuwymiar. dyskretnego przekształcenia Hartleya

$\mathbf{P}_F$ – macierz permutacji wejściowej dla zunifikowanego algorytmu wyznaczania dwuwymiar. dyskretnego przekształcenia Fouriera

R – współczynnik kompresji

PSNR – szczytowy stosunek sygnału do szumu dla odtwarzanego obrazu

KLT – dyskretne przekształcenie Karhunena-Loève'go

DFT/FFT – dyskretna/szybka transformata Fouriera

DCT/FCT – dyskretna/szybka transformata kosinusowa

DST/FST – dyskretna/szybka transformata sinusowa

DHT/FHT – dyskretna/szybka transformata Hartleya

ZIG-ZAG – skrót odnoszący się do kolejności kodowania współczynników transformaty kosinusowej w procesie kwantyzacji standardu JPEG

`BP`	– pojedynczy krok algorytmu wstecznej propagacji błędu
`FP`	– pojedynczy krok propagacji prostej sygnału w liniowej sieci neuronowej
`FCT/FCT1D`	– szybki algorytm jednowymiarowej transformaty kosinusowej
`FST/FST1D`	– szybki algorytm jednowymiarowej transformaty sinusowej
`FHT/FHT1D`	– szybki algorytm jednowymiarowej transformaty Hartleya
`FFT/FFT1D`	– szybki algorytm jednowymiarowej transformaty Fouriera
`JFCT1D/JFCT2D`	– zunifikowany algorytm jedno/dwuwymiarowej transformaty kosinusowej
`JFST1D/JFST2D`	– zunifikowany algorytm jedno i dwuwymiarowej transformaty sinusowej
`JFHT1D/JFHT2D`	– zunifikowany algorytm jedno/dwuwymiarowej transformaty Hartleya
`JFFT1D/JFFT2D`	– zunifikowany algorytm jedno/dwuwymiarowej transformaty Fouriera
`KLT2D`	– algorytm wyznaczania dwuwymiarowej transformaty Karhunena-Loève'go
`FCT2D`	– szybki algorytm dwuwymiarowej transformaty kosinusowej
`MLP2D`	– standardowy autokoder neuronowy do kompresji obrazów
`NFJT2D`	– zunifikowany, szybki autokoder neuronowy do kompresji obrazów

1. WSTĘP

1.1. Wprowadzenie

Algorytmy i metody kompresji danych stanowią obecnie bardzo istotną składową systemów informatycznych wspomagających działalność człowieka w wielu obszarach jego aktywności. Poczynając od zastosowań w dziedzinie teleinformatyki i telekomunikacji [1–3], projektach informatycznych opierających się o wykorzystanie ogólnoświatowej sieci komputerowej Internet [4], skończywszy zaś na systemach archiwizacji i katalogowania dużych zbiorów danych stosowanych często w przemyśle np. dla celów wizualizacji diagnostycznej [5, 6] oraz w medycynie i kryminalistyce [7–9]. Już od momentu powstania szczególną uwagę badaczy zajmujących się metodami kompresji zwróciły systemy multimedialne, w których ilość przetwarzanych informacji jest bardzo duża w porównaniu z innymi dziedzinami zastosowań systemów informatycznych. Wynika to zarówno z samej natury problemu dostatecznie dobrego (z jakościowego punktu widzenia) kodowania sygnałów reprezentujących dane multimedialne, jak i stale rosnącej mocy obliczeniowej ówczesnych i dzisiejszych systemów komputerowych.

Wychodząc naprzeciw wyzwaniu zmniejszenia ogromnej ilości informacji transmitowanych i gromadzonych na potrzeby systemów multimedialnych, bez jednoczesnej utraty jakości przekazu i przy utrzymaniu krytycznych wartości prędkości przetwarzania dla omawianych systemów, począwszy od lat siedemdziesiątych ubiegłego wieku zaczęto projektować dedykowane techniki kompresji uwzględniające specyfikę tak postawionych zadań. Bardzo intensywnie rozwijaną gałęzią w ramach wspomnianych technik stały się metody kompresji obrazów, stanowiące jedną z najistotniejszych form przekazu multimedialnego. Bazując na dobrze już wówczas znanej teorii liniowej filtracji procesów stochastycznych [10–13] poprzez rozszerzenie przydatnych pojęć na przypadek dwuwymiarowy, dokonano pierwszych prób syntezy modeli teoretycznych obrazu naturalnego na potrzeby konstrukcji algorytmów kompresji za pomocą narzędzi teorii rachunku probabilistycznego [14]. Próby te rozwijano intensywnie w dalszych latach [15, 16].

Wspomniane modele, jakkolwiek nie pozbawione szeregu wad i często dyskusyjnych uproszczeń [17], okazały się niezwykle pomocne w zrozumieniu własności statystycznych obrazów naturalnych oraz przyczyniły się w późniejszym okresie do powstania efektywnych jakościowo i obliczeniowo algorytmów ich stratnej kompresji. Równolegle toczyły się prace nad nowymi typami dyskretnych przekształceń ortogonalnych, dedykowanych między innymi problemowi kompresji stratnej jednowymiarowych sygnałów stochastycznych. Zdefiniowano wiele nowych rodzin przekształceń [18] i podano efektywne metody ich obliczania, opierając się na pomyśle

szybkiego algorytmu wyznaczania transformaty Fouriera (FFT[1]) [19]. Geneza taka dotyczy również przekształcenia stosowanego obecnie najczęściej w praktyce kompresji obrazów, określanego mianem dyskretnej transformaty kosinusowej (w skrócie DCT[2]). Ze względu na fakt, że transformata ta została zaprojektowana przez jej autorów [20] jako graniczny przypadek [21] teoretycznie optymalnego[3] przekształcenia Karhunena-Loève'go [11, 12] dla silnie skorelowanego stacjonarnego procesu Markowa pierwszego rzędu[4], zaczęła się ona cieszyć dużym zainteresowaniem.

Zainteresowanie to stało się również udziałem badaczy i praktyków zagadnienia kompresji stratnej sygnałów rzeczywistych, w tym także obrazów. Z punktu widzenia tych ostatnich najistotniejszą przesłanką przemawiającą na rzecz szerokiego, praktycznego wykorzystania DCT, był fakt, że w oryginalnej pracy [20] autorzy nie ograniczyli się jedynie do wyprowadzenia omawianego przekształcenia, podali także szybki algorytm jego wyznaczania (oparty o FFT) i zaprezentowali wyniki symulacji filtracji świadczące pod wieloma względami o przewadze DCT nad innymi, znanymi wcześniej, jednowymiarowymi dyskretnymi przekształceniami ortogonalnymi. W rezultacie transformata kosinusowa uznana została przez środowisko praktyków dziedziny kompresji stratnej za potencjalnie obiecujący kompromis pomiędzy jakością kompresji a jej obliczeniową efektywnością. Zaowocowało to – po rozszerzeniu podstawowej metody kodowania transformacyjnego danych obrazowych na przypadek dwuwymiarowy z blokową kwantyzacją skalarną [3, 22][5] i bezstratnym, entropijnym algorytmem kodowania współczynników transformaty [23] – powstaniem współczesnych standardów kompresji stratnej obrazów, w tym bardzo dziś popularnego algorytmu JPEG [4].

Choć wspomniany standard oferuje zadowalającą w większości zastosowań jakość przetworzonych obrazów przy dobrych poziomach kompresji istnieją dziedziny, w których możliwości algorytmu JPEG są niewystarczające. Za przykład mogą tu posłużyć problemy kompresji w obrazowaniu przemysłowym, medycznym, czy kryminalistycznym. Stąd też w ostatnich latach podejmowano liczne próby ulepszenia podstawowego schematu kodowania podanego omawianym standardem. Można wymienić tu chociażby prace dotyczące zastosowania przekształceń falkowych[6] w kompresji obrazów medycznych [24], transformat zakładkowych[7] [25] redukujących niepożądany efekt blokowości skompresowanego obrazu, czy wreszcie ostatnią standaryzowaną odsłonę standardu JPEG – metodę JPEG2000 [26], umożliwiającą zróżnicowanie stopnia kompresji obszarów obrazu o różnym znaczeniu merytorycznym. Równolegle, prężnie rozwijającą się w ostatnich latach gałęzią badań nad poprawą efektywności metod kompresji obrazów są adaptacyjne algorytmy kompresji oparte o wykorzystanie sztucznych sieci neuronowych, których interesujące przykłady zastosowań można znaleźć między innymi w pracach [5, 27–34].

[1] ang. *fast Fourier transform*
[2] ang. *discrete cosine transform*
[3] przekształcenia optymalnego, między innymi, w sensie aproksymacji sygnału na podstawie niepełnego zestawu próbek
[4] popularnym, teoretycznym modelu stochastycznym licznej grupy procesów rzeczywistych
[5] oryginalna wersja artykułu S.P. Lloyda została przedstawiona już w roku 1957
[6] ang. *wavelet transforms*
[7] ang. *lapped transforms*

Autor w swej dotychczasowej pracy badawczej zajmował się wymienioną tematyką, w szczególności konstrukcją efektywnych obliczeniowo szybkich zunifikowanych algorytmów obliczania dyskretnych przekształceń liniowych, zarówno jedno-, jak i dwuwymiarowych oraz ich neuronowych realizacji w zastosowaniu do kompresji obrazów [35–43]. Praca ta była realizowana w Zespole badawczym adaptacyjnych metod przetwarzania sygnałów i obrazów Instytutu Informatyki Politechniki Łódzkiej, kierowanym przez prof. dr. hab. Michała Jacymirskiego, a następnie kontynuowana w Zespole badawczym masowo-równoległego, adaptacyjnego przetwarzania danych i obliczeń kwantowych Instytutu Informatyki Politechniki Łódzkiej pod kierownictwem dr. hab. inż. Dariusza Puchały. Treść niniejszej monografii stanowi podsumowanie tych prac w aspekcie konstrukcji efektywnych obliczeniowo szybkich zunifikowanych algorytmów obliczania dyskretnych przekształceń liniowych, zarówno jedno jak i dwuwymiarowych, i ich neuronowych realizacji w zastosowaniu do kompresji obrazów.

Opracowanie przedstawia propozycje efektywnych obliczeniowo metod automatyzacji procesu doboru dyskretnych przekształceń liniowych dostosowanych do zadania wydajnej jakościowo i obliczeniowo stratnej kompresji wybranych typów obrazów rzeczywistych. Umożliwia to zwolnienie projektanta schematu kompresji z konieczności analitycznego wyznaczania postaci transformat roboczych, przeznaczonych na potrzeby wspomnianego zadania. Rola projektanta ogranicza się w tym przypadku jedynie do przygotowania zestawu obrazów przykładowych, charakterystycznych dla wybranej dziedziny zastosowań, które służą następnie jako dane wzorcowe w procesie optymalizacji przekształceń kompresującego i dekodującego.

Prezentowana monografia powstała na podstawie rozprawy doktorskiej [44] autora pt. "Kompresja obrazów za pomocą sieci neuronowych realizujących szybkie przekształcenia dyskretne" napisanej pod opieką naukową prof. dr. hab. inż. Piotra S. Szczepaniaka z Wydziału Fizyki Technicznej Informatyki i Matematyki Stosowanej Politechniki Łódzkiej, obronionej w 2010 na tymże wydziale i zrecenzowanej przez prof. dr. hab. inż. Marka Kurzyńskiego z Wydziału Elektroniki Politechniki Wrocławskiej oraz prof. dr. hab. Michała Jacymirskiego z Wydziału Fizyki Technicznej Informatyki i Matematyki Stosowanej Politechniki Łódzkiej.

1.2. Aktualny stan wiedzy i przebieg badań

Oprócz bardzo ważnych zastosowań metod kompresji obrazów, takich jak projekty multimedialne w ramach sieci Internet, systemy medyczne, telemedyczne kryminalistyczne i przemysłowe, w ostatnich latach można zaobserwować bardzo dynamiczny rozwój wielu nowych form komunikacji multimedialnej, szczególnie multimedialnych serwisów internetowych, telewizji cyfrowej czy multimedialnej komunikacji mobilnej. Wymusza to ciągły postęp w dziedzinie metod kompresji obrazów i konieczność projektowania nowych, efektywnych z jakościowego i obliczeniowego punktu widzenia, algorytmów kompresji obrazów co powoduje, że dziedzina ta cieszy się niezmiennie dużym zainteresowaniem środowiska naukowego i przemysłu. Fakty te pozwalają bezpiecznie prognozować dalsze zapotrzebowanie na intensywny rozwój technik kompresji multimediów, w tym również obrazów.

Wymienione tu przesłanki stały się głównym motorem motywującym autora niniejszej pracy do podjęcia badań w rozważanej dziedzinie. Innym, równie istotnym bodźcem, który skłonił autora do zajęcia się problematyką konstrukcji efektywnych obliczeniowo adaptacyjnych metod kompresji obrazów i metod jest wniosek o możliwości ulepszenia istniejących algorytmów, oparty o analizę dotychczasowych prac, między innymi tych, podanych we wstępie. Wreszcie jako trzeci z głównych powodów realizacji badań w omawianym obszarze, można podać pojawienie się w ostatniej dekadzie nowych metod optymalizacyjnych opartych o wykorzystanie sieci neuronowych, patrz np. [33, 35, 36]. Metody te otwierają zupełnie nową perspektywę podejścia do konstrukcji skutecznych algorytmów kompresji obrazów i są atrakcyjną alternatywą dla klasycznych sposobów ich projektowania.

Treść poprzedniego podrozdziału pokazuje dotychczasowe kierunki rozwoju algorytmów kompresji obrazów i zgrubnie nakreśla potencjalne możliwości ulepszenia istniejących metod. Skłania to do wniosku, że dotychczasowe wysiłki badawcze były skierowane na konstrukcję metod kompresji, które z jednej strony starają się przybliżać znane rozwiązania teoretycznie optymalne, z drugiej zaś pozostają efektywne obliczeniowo. W ramach najpopularniejszego obecnie schematu kompresji obrazów z użyciem kodowania transformacyjnego JPEG, zdaniem autora, ten swoisty kompromis przesunięty jest znacznie w stronę efektywności obliczeniowej kosztem jakości metod kompresji. Z powodów wymienionych na początku niniejszego podrozdziału, cały czas prowadzone są prace zmierzające do poprawy istniejących metod kompresji obrazów. Są one prowadzone równolegle w wielu obszarach. I tak, stosunkowo niedawne osiągnięcia w zakresie modelowania statystycznego obrazów naturalnych są przedstawione w pracach [45, 46], a także w [47–49].

Z nowymi rodzajami szybkich, liniowych przekształceń dyskretnych stworzonych, między innymi z myślą o metodach kompresji obrazów można zapoznać się dzięki lekturze opracowań [50–52] jak również [53, 54]. Istotną częścią prac związanych z poprawą efektywności czasowej obliczania przekształceń dyskretnych stały się ostatnio propozycje ich masowo-równoległych realizacji z wykorzystaniem kart graficznych (GPU - ang. *Graphics Processing Units*), patrz np. [37, 39–43]. Probabilistyczne modele procesu kwantyzacji zostały zaprezentowane w pracach [55, 56]. Dodatkowo, analizę statystyczną wpływu modelowanej w ten sposób kwantyzacji równomiernej[1] na postaci dyskretnych przekształceń kodujących/dekodujących, optymalnych w zagadnieniu kompresji stratnej sygnałów jednowymiarowych dla wysokich współczynników kompresji można znaleźć w pracach [57, 58], zaś w opracowaniach [59, 60] została ona rozszerzona na przypadek łącznej kompresji z szyfrowaniem obrazów wraz z propozycją realizujących tę metodę algorytmów obliczeniowych. Wartościową analizę porównawczą technik stratnej kompresji obrazów za pomocą znanych rodzajów przekształceń liniowych można znaleźć chociażby w pracy [61]. Wreszcie, ciekawe zestawienie używanych obecnie miar jakości metod kompresji stratnej znajduje się między innymi w książce [62], choć temat ten jest zdecydowanie najsłabiej rozwiniętą gałęzią badań omawianej dziedziny.

Stosunkowo niedawno pojawiły się propozycje wykorzystania sztucznych sieci neuronowych, opisanych obszernie między innymi w książkach [63], [28] czy [29]

[1] schematu kwantyzacji używanego między innymi w standardzie JPEG

w problemie stratnej kompresji obrazów. Ich zastosowanie we wspomnianym zadaniu posiada dwie zasadnicze zalety, które wspólnie czynią z podejścia neuronowego bardzo atrakcyjną alternatywę dla innych sposobów poprawy skuteczności metod kompresji obrazów. Są to adaptacyjność i współbieżność obliczeniowa algorytmów neuronowych. Obydwie z wymienionych cech umożliwiają wydajną czasowo realizację procesów adaptacyjnej konstrukcji i późniejszego obliczania dyskretnych przekształceń liniowych, dostosowanych do zadania kompresji stratnej założonej klasy obrazów rzeczywistych o wspólnych cechach statystycznych. Co istotne, za sprawą zrównoleglenia obliczeń specyficznego dla metod neuronowych, wspomniana wydajność czasowa obydwu rozważanych procesów dotyczy także przekształceń o dużej złożoności obliczeniowej (rozumianej w sensie klasycznym), a przez to pozwala na uzyskanie efektywnych implementacji transformat o bardzo dobrej charakterystyce jakościowej.

Sprawność czasowa procesów adaptacji i obliczania dyskretnych przekształceń liniowych o dużej złożoności obliczeniowej manifestuje się jednak jedynie w przypadku sprzętowej realizacji metod neuronowych. Przykłady sieci neuronowych wraz z dedykowanymi algorytmami nauczania, zakładające taki właśnie sprzętowy scenariusz ich implementacji zostały zaprezentowane w opracowaniach [64] oraz [29]. W przypadku symulacji komputerowej metod neuronowych tradycyjna złożoność rachunkowa omawianych procesów ma już jednak bardzo istotny wpływ zarówno na czas jak i skuteczność ich realizacji. Ponadto symulacja komputerowa sieci neuronowych w kompresji jest najmniej kosztownym sposobem konstrukcji i testowania uzyskanych przekształceń. Powyższe przyczyny doprowadziły do narodzin pomysłu tzw. „szybkich sieci neuronowych" o efektywnej obliczeniowo architekturze umożliwiającej jednocześnie skuteczną jakościowo adaptację sieci do zadania kompresji wybranej klasy obrazów rzeczywistych. Zarysy teoretyczne pomysłu szybkich sieci neuronowych zostały podane w publikacjach [6, 27, 28, 32, 65–74], następnie pomysł ten sukcesywnie rozwijany był m. in. w pracach [31, 75–79]. Część nich stała się bezpośrednią inspiracją autora niniejszej monografii do prowadzenia badań w dziedzinie efektywnej obliczeniowo kompresji obrazów.

Podsumowując, niesłabnące zainteresowanie ciągłym usprawnianiem istniejących technik kompresji, postęp teorii modelowania procesu kompresji obrazów oraz rozwój metod optymalizacji neuronowej (opis szerokiego spektrum praktycznego zastosowania technik neuronowych w szczególności w obrazowaniu medycznym i innych gałęziach medycyny można znaleźć między innymi w [80–84]) tworzą obiecującą perspektywę poprawy zarówno jakościowej jak i efektywnościowej metod kompresji obrazów w oparciu o syntezę wymienionych obszarów wiedzy.

2. SZYBKIE ZUNIFIKOWANE PRZEKSZTAŁCENIA DYSKRETNE

2.1. Wprowadzenie

W niniejszym rozdziale przedstawione i wyprowadzone zostaną szybkie zunifikowane algorytmy wyznaczania wybranych przekształceń dyskretnych, które stanowią bazę zaprezentowanych w pracy konstrukcji sieci neuronowych do kompresji obrazów. Algorytmy te poddane zostaną teoretycznej analizie efektywnościowo pojemnościowej pozwalającej na późniejszą weryfikację skuteczności metod neuronowych opartych o ich wykorzystanie. Najpierw podane zostaną zasady konstrukcji szybkich algorytmów obliczeniowych wybranych dyskretnych transformat ortonormalnych. Wyprowadzono i dokonano analizy najistotniejszych cech jednego z wybranych algorytmów szybkich realizacji przekształceń dyskretnych posługując się w tym przypadku przykładem dyskretnej transformaty Fouriera.

Wybór tego przekształcenia przykładowego wynika z faktu, iż zdaniem autora, szybka metoda jego wyznaczania nacechowana jest wyjątkową wśród innych, analizowanych w dalszej części rozdziału przekształceń, prostotą struktury obliczeniowej, a jednocześnie sposób jej konstrukcji w pełni odpowiada ogólnej zasadzie konstrukcyjnej innych szybkich algorytmów obliczania przekształceń dyskretnych. Warto dodać, że część rozdziału odnosząca się do przykładowej transformaty Fouriera obejmuje także ogólne zasady budowy i problem efektywności szybkich algorytmów obliczeniowych dla przekształceń dwuwymiarowych i odwrotnych, zarówno jedno jak i dwuwymiarowych. Kolejna część materiału dotyczy analizy efektywnościowej szybkich algorytmów wyznaczania przekształcenia Hartleya i transformat kosinusowej oraz sinusowej. Wybór ten podyktowany jest faktem, iż nowe metody adaptacyjne prezentowane w niniejszej pracy zostały skonstruowane w oparciu o szybkie algorytmy wyznaczania tychże czterech wybranych przekształceń i są z nimi dalej bezpośrednio porównywane. Następnie autor przejdzie do głównego tematu tej części opracowania, a więc wyprowadzenia szybkich zunifikowanych algorytmów wyznaczania wybranych przekształceń dyskretnych.

Pomysł szybkich algorytmów zunifikowanych wyznaczania ortogonalnych przekształceń dyskretnych został oryginalnie zaproponowany przez prof. dr. hab. Michała Jacymirskiego, który, między innymi, w pracach [85] oraz [86] podał ich postaci dla sygnałów jednowymiarowych oraz dokonał odpowiednich wyprowadzeń. Podstawowymi wyróżnikami tych algorytmów wśród innych szybkich metod obliczeniowych dla szeregu przekształceń dyskretnych, są efektywność obliczeniowa równa rzędem znanym metodom szybkim wyznaczania transformat ortogonalnych oraz, przede

wszystkim, ich zunifikowana struktura grafowa. Ta ostatnia szczególnie istotna cecha polega na tym, że metody te definiują sposób obliczania wielu różnych przekształceń dyskretnych przy pomocy jednolitej, regularnej i szybkiej zarazem struktury grafu przepływu danych. Obliczanie współczynników wybranych transformat ortogonalnych wymaga w tym przypadku jedynie ewentualnej zmiany permutacji elementów wektorów wejściowych, względnie wyjściowych, dla danego przekształcenia dyskretnego oraz odpowiedniej modyfikacji współczynników obrotów wewnątrz odpowiadającego mu grafu przepływowego, bez zmiany struktury połączeń w obrębie samego grafu. Możliwość organizacji obliczeń współczynników przekształceń dyskretnych w opisany przed chwilą sposób sugeruje z kolei uniwersalny, choć w dalszym ciągu efektywny obliczeniowo, charakter omawianych algorytmów. Ostatnie spostrzeżenie doprowadziło do powstania pomysłu wykorzystania struktur grafów przepływowych algorytmów zunifikowanych do syntezy architektury sieci neuronowych, realizujących rozmaite zadania adaptacyjnej filtracji liniowej w tym również adaptacyjną kompresję obrazów. Tak zwane *szybkie sieci neuronowe* o architekturze odpowiadającej szybkim zunifikowanym algorytmom wyznaczania przekształceń dyskretnych zostały oryginalnie zaproponowane i przedstawione w publikacji [27].

W niniejszej pracy przedstawione zostaną kompleksowe wyniki analiz i badań tychże szybkich sieci neuronowych w zastosowaniu do problemu kompresji obrazów. Częściowe opracowania dotyczące zarówno zagadnień teoretycznych jak i wyników badań praktycznych nad kompresją obrazów za pomocą rozważanych tu szybkich sieci neuronowych zostały zaprezentowane już wcześniej między innymi również przez autora niniejszej pracy w publikacjach [32, 65, 67, 87]. Jak wspomniano na początku niniejszego podrozdziału podstawą konstrukcji szybkich neuronowych algorytmów adaptacyjnej kompresji obrazów są zunifikowane metody wyznaczania dyskretnych przekształceń ortogonalnych. Opracowania dotyczące ostatniego z wymienionych tu zagadnień można znaleźć w pracach [28, 85, 86, 88]. Aby jednak zapewnić kompleksowość opisu zagadnień z zakresu tematyki niniejszej pracy, mając na względzie fundamentalną rolę algorytmów zunifikowanych w konstrukcji architektury sieci neuronowych do adaptacyjnej kompresji obrazów autor zdecydował się poddać wspomniane algorytmy szczegółowej analizie uwzględniając też ich wyprowadzenia, w tym także stanowiący oryginalny wkład autora niniejszej pracy w omawiane tu zagadnienie opis zunifikowanych algorytmów szybkich dla przypadku przekształceń dwuwymiarowych, niezbędnych dla neuronowej implementacji technik adaptacyjnej kompresji obrazów z ich użyciem.

2.2. Szybkie algorytmy przekształceń dyskretnych

Treść niniejszego podpunktu obejmuje dość szczegółową analizę efektywnościową wybranych algorytmów szybkich wyznaczania rodziny transformat Fouriera. Jak zapowiedziano wcześniej wynika to z faktów, iż dyskretna transformata Fouriera pełni istotną rolę w teorii przetwarzania sygnałów, a ponadto szybkie algorytmy jej realizacji są chronologicznie najwcześniejszą [19] i, zdaniem autora, najbardziej przejrzystą koncepcyjnie grupą metod szybkich wyznaczania przekształceń dyskretnych rozważanych dalej w niniejszym rozdziale, a co za tym idzie stanowią dobry przykład

ogólnych zasad konstrukcji algorytmów szybkich dla wspomnianych tu pozostałych transformat dyskretnych. Podajmy definicję jednowymiarowego dyskretnego przekształcenia Fouriera.

Definicja 2.1.

Dla dowolnej liczby $N \in \mathbb{N}$ parę funkcji $F_N, F_N^{-1} \colon \mathbb{C}^N \to \mathbb{C}^N$ zdefiniowanych w następujący sposób:

$$\forall\, \mathbf{x} \in \mathbb{C}^N \quad \forall\, k \in \{0, \ldots, N-1\} \qquad F_N(\mathbf{x})_k \stackrel{\Delta}{=} \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \cdot e^{-j2\pi nk/N} \tag{2.1a}$$

$$\forall\, \mathbf{y} \in \mathbb{C}^N \quad \forall\, n \in \{0, \ldots, N-1\} \qquad F_N^{-1}(\mathbf{y})_n \stackrel{\Delta}{=} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \cdot e^{\,j2\pi kn/N} \tag{2.1b}$$

zwiemy *N–punktowym prostym i odwrotnym jednowymiarowym dyskretnym przekształceniem Fouriera*.

Łatwo pokazać,[1] że obydwie przedstawione wyżej funkcje (2.1a) oraz (2.1b) są unitarne i w istocie tworzą parę przekształceń wzajemnie odwrotnych dla dowolnego $\mathbf{z} \in \mathbb{C}^N$. Zajmijmy się teraz zagadnieniem efektywności obliczeniowej i pamięciowej bezpośrednich metod wyznaczania współczynników przedstawionych przekształceń bazujących na definicji 2.1. Algorytmy obliczania przekształceń jednowymiarowych procedurami bezpośrednimi za pomocą wzorów (2.1a) i (2.1b) nazywa się często w skrócie odpowiednio przekształceniami `DFT1D` oraz `IDFT1D`.[2] Przekształcenie Fouriera operuje na zmiennych zespolonych i jak powszechnie wiadomo uzyskanie wyniku operacji prostej (tzn. operacji dodawania lub odejmowania) dla dwóch argumentów zespolonych wymaga wykonania dwóch operacji prostych na zmiennych rzeczywistych. Analogicznie mnożenie dwóch liczb zespolonych wymaga przeprowadzenia czterech mnożeń rzeczywistych i dwóch operacji prostych. Wobec tego nie trudno dojść do wniosku, że obliczenie pełnej prostej N–punktowej dyskretnej transformaty Fouriera wzorem (2.1a) wymaga wykonania $4N^2 - 2N$ prostych operacji rzeczywistych i $4N^2$ rzeczywistych mnożeń. Złożoność pamięciowa metody bezpośredniej `DFT1D` jest liniowa, choć niestety wymagana jest w tym przypadku rezerwacja w pamięci systemu komputerowego dodatkowej tablicy złożonej z N zmiennych zespolonych, ponieważ każda składowa wektora wynikowego wyznaczana jest na podstawie wszystkich elementów zespolonych wektora wejściowego.

2.2.1. Szybka jednowymiarowa transformata Fouriera

W niniejszej części monografii zaprezentowany zostanie jeden z algorytmów pozwalających zredukować o rząd wielkości złożoność obliczeniową zadania wyznaczania współczynników jednowymiarowej dyskretnej transformaty Fouriera w stosunku do metody bezpośredniej. Po raz pierwszy algorytm ten został podany w pracy [19]. Autor zdecydował się na przedstawienie wyprowadzenia i szczegółową analizę

[1] patrz chociażby [89]

[2] ang. *one dimensional discrete Fourier transform* oraz *one dimensional inverse discrete Fourier transform*

efektywności rozważanego algorytmu, ze względu na fakt, iż jest to jeden z najważniejszych algorytmów w historii badań metod szybkich transformat dyskretnych i dobrze obrazuje ogólną zasadę konstruowania wspomnianych procedur dla innych rodzajów przekształceń. Podajmy własności:

$$\forall\, \beta \in \mathbb{R} \qquad e^{\,j\beta} \;=\; cos\,(\,\beta\,) + j\; sin\,(\,\beta\,), \tag{2.2a}$$

$$\forall k \in \mathbb{Z}\; e^{-j\pi k} \overset{(2.2a)}{=} cos(-\pi k) + j\, sin(-\pi k) = cos(\pi k) - j\, sin(-\pi k) = cos(\pi k) = (-1)^{\,k} \tag{2.2b}$$

gdzie własność (2.2a) jest dobrze znanym *wzorem Eulera* dla postaci wykładniczej liczby zespolonej. Niech $N \in \mathbb{N}$ będzie pewną naturalną potęgą liczby dwa. Utwórzmy funkcję pomocniczą $\overline{F}_N \colon \mathbb{C}^{\,N} \to \mathbb{C}^{\,N}$

$$\forall\, \mathbf{x} \in \mathbb{C}^{\,N} \quad \forall\, k \in \{\,0, \ldots, N-1\,\} \qquad \overline{F}_N(\,\mathbf{x}\,)_k \;\triangleq\; \sum_{n=0}^{N-1} x_n \;\cdot\; e^{-j2\pi nk/N} \tag{2.3}$$

Jak widać funkcja $\overline{F}_N$ jest po prostu przeskalowaną wersją N–punktowej jednowymiarowej dyskretnej transformaty Fouriera danej zależnością (2.1a), dlatego nazywajmy ją dalej skalowanym przekształceniem Fouriera. Niech $\mathbf{x} \in \mathbb{C}^{\,N}$, zauważmy, że dla dowolnego $k \in \{\,0, \ldots, N-1\,\}$ współczynnik k–ty N–punktowej skalowanej dyskretnej transformaty Fouriera można, korzystając z własności funkcji wykładniczej, wyrazić w następujący sposób:

$$\begin{aligned}
\overline{F}_N(\,\mathbf{x}\,)_k &\overset{(2.3)}{=} \sum_{n=0}^{N-1} x_n \;\cdot\; e^{-j2\pi nk/N} \;= \\
&= \sum_{n=0}^{N/2-1} x_n \;\cdot\; e^{-j2\pi nk/N} \;+\; \sum_{n=N/2-1}^{N-1} x_n \;\cdot\; e^{-j2\pi nk/N} \;= \\
&= \sum_{n=0}^{N/2-1} \left[\; x_n \;\cdot\; e^{-j2\pi nk/N} \;+\; x_{N/2+n} \cdot\; e^{-j2\pi\,(N/2+n)\,k/N} \;\right] = \\
&= \sum_{n=0}^{N/2-1} \left[\; x_n \;\cdot\; e^{-j2\pi nk/N} \;+\; x_{N/2+n} \;\cdot\; e^{-j2\pi nk/N} \;\cdot\; e^{-j\pi k} \;\right] = \\
&= \sum_{n=0}^{N/2-1} \left[\; x_n \;+\; x_{N/2+n} \;\cdot\; e^{-j\pi k} \;\right] \cdot\; e^{-j2\pi nk/N} \;= \\
&\overset{(2.2b)}{=} \sum_{n=0}^{N/2-1} \left[\; x_n \;+\; (-1)^{\,k}\, x_{N/2+n} \;\right] \cdot\; e^{-j2\pi nk/N}
\end{aligned} \tag{2.4}$$

Utwórzmy zmienną pomocniczą $W_N^n \in \mathbb{C}$ oraz zespolone wektory $\overline{\mathbf{x}},\, \widetilde{\mathbf{x}} \in \mathbb{C}^{\,N/2}$ w następujący sposób:

$$\forall n \in \{0, \ldots, N/2-1\}\; W_N^n \triangleq e^{-j2\pi n/N};\; \overline{x}_n \triangleq x_n + x_{N/2+n};\; \widetilde{x}_n \triangleq [\,x_n - x_{N/2+n}\,]\, W_N^n \;. \tag{2.5}$$

Niech $l \in \{\, 0, \ldots, N/\,2-1\}$, dla parzystych współczynników N–punktowej skalowanej dyskretnej transformaty Fouriera, podanej zależnością (2.3) otrzymujemy następujący ciąg równości:

$$\begin{aligned}
\overline{F}_N(\,\mathbf{x}\,)_{\,2l} &\overset{(2.4)}{=} \sum_{n=0}^{N/2-1} \left[\; x_n \;+\; (-1)^{\,2l}\, x_{N/2+n} \;\right] \cdot e^{-j2\pi n 2l/N} \;= \\
&= \sum_{n=0}^{N/2-1} \left[\; x_n \;+\; x_{N/2+n} \;\right] \cdot e^{-j2\pi nl/(N/2)} \;= \\
&\overset{(2.5)}{=} \sum_{n=0}^{N/2-1} \overline{x}_n \cdot e^{-j2\pi nl/(N/2)} \overset{(2.3)}{=} \overline{F}_{N/2}(\,\overline{\mathbf{x}}\,)_l\,.
\end{aligned} \tag{2.6a}$$

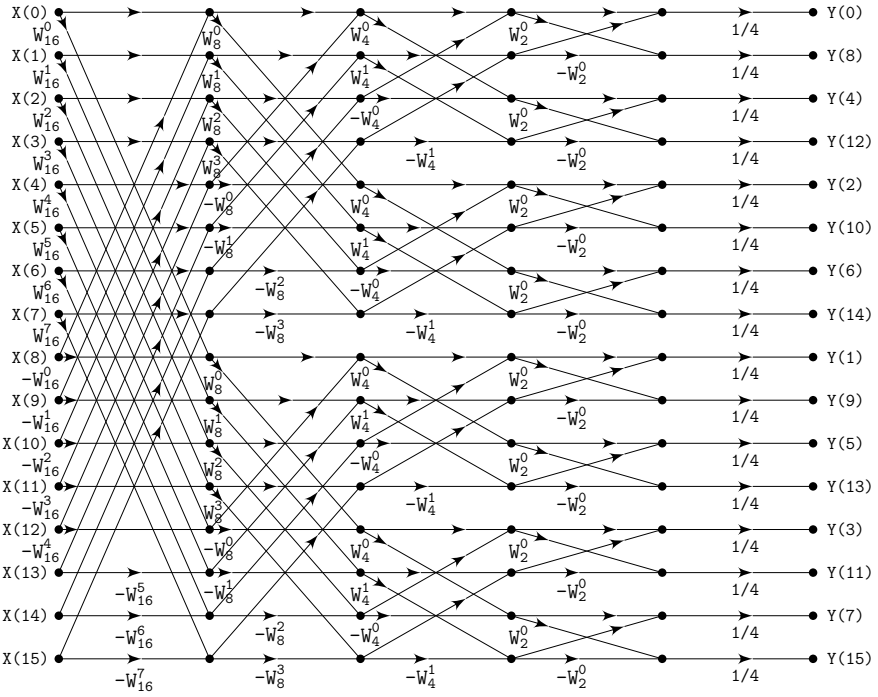
Podobnie dla współczynników nieparzystych dostajemy:

$$\begin{aligned}
\overline{F}_N(\,\mathbf{x}\,)_{\,2l+1} &\overset{(2.4)}{=} \sum_{n=0}^{N/2-1} \left[\; x_n \;+\; (-1)^{\,2l+1}\, x_{N/2+n} \;\right] \cdot e^{-j2\pi n 2(l+1)/N} \;= \\
&= \sum_{n=0}^{N/2-1} \left[\; x_n \;-\; x_{N/2+n} \;\right] e^{-j2\pi n/N} \cdot e^{-j2\pi nl/(N/2)} \;= \\
&\overset{(2.5)}{=} \sum_{n=0}^{N/2-1} \left[\; x_n \;-\; x_{N/2+n} \;\right] W_N^n \cdot e^{-j2\pi nl/(N/2)} \;= \\
&\overset{(2.5)}{=} \sum_{n=0}^{N/2-1} \widetilde{x}_n \cdot e^{-j2\pi nl/(N/2)} \overset{(2.3)}{=} \overline{F}_{N/2}(\,\widetilde{\mathbf{x}}\,)_l\,.
\end{aligned} \tag{2.6b}$$

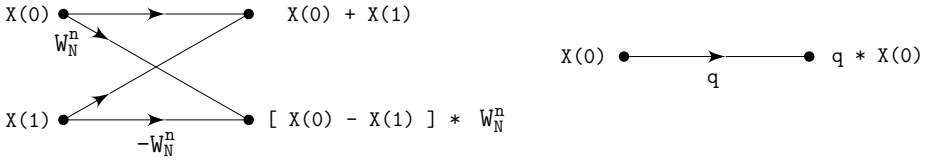
Równania (2.6a) i (2.6b) określają zależność pomiędzy parzystymi i nieparzystymi współczynnikami szukanej N–punktowej skalowanej dyskretnej transformaty Fouriera, a współczynnikami analogicznych przekształceń $N/\,2$–punktowych wektorów pomocniczych $\overline{\mathbf{x}}$ oraz $\widetilde{\mathbf{x}}$. Rekursywne zastosowanie zależności (2.6a) i (2.6b) względem wszystkich dyskretnych transformat wyjściowych, stanowiących pośrednie rezultaty skonstruowanego w ten sposób procesu obliczeniowego, powoduje w końcu uzyskanie sekwencji N jednopunktowych skalowanych dyskretnych transformat Fouriera, które z definicji (2.3) są równe co do wartości swoim (jednopunktowym) wektorom wejściowym, a co za tym idzie, prowadzi ostatecznie do wyznaczenia szukanego, wejściowego N–punktowego dyskretnego przekształcenia Fouriera F_N. Opisana przed momentem rekursywna procedura obliczeniowa określana jest mianem algorytmu *szybkiej transformaty Fouriera* `FFT1D`[1] i wobec założonej wcześniej dowolności wyboru N może być zastosowana do wyznaczania współczynników przekształceń o dowolnym wymiarze będącym naturalną potęgą liczby dwa. Bezpośrednia, rekurencyjna implementacja rozpatrywanej metody, choć oczywiście możliwa do realizacji w praktyce, jest w większości systemów komputerowych mało efektywna. Dlatego też najczęściej dokonuje się implementacji algorytmu `FFT1D` metodą iteracyjną z pojedynczą tablicą współczynników, przechowującą w kolejnych etapach przetwarzania wszystkie

[1] ang. (*decimation in frequency*) *one dimensional fast Fourier transform*

pośrednie wyniki obliczeń.[1] Rysunek 2.1a przedstawia graficzną interpretację omawianej metody iteracyjnej obliczania 16-punktowego przekształcenia Fouriera. Graf przepływu danych uwzględnia też skutkującą iteracyjną organizacją omawianej procedury permutację współczynników wyjściowych tej wersji algorytmu `FFT1D` zwaną *odwróconym porządkiem bitowym*,[2]



Rys. 2.1a: Szybki algorytm obliczania 16-punktowego dyskretnego przekształcenia Fouriera `FFT1D`



Rys. 2.1b: Operacje podstawowe algorytmu `FFT1D`; $X(0)$, $X(1)$, $q \in \mathbb{C}$; $W_N^n \overset{(2.5)}{=} e^{-j2\pi n/N}$

Po analizie grafu przepływu danych dla N–punktowej dyskretnej transformaty Fouriera można dojść do wniosku, że złożoność obliczeniowa rozważanego algorytmu wynosi [90]:

$$\mathcal{F}_N^+ = 3\,N(\,log_2\,N - 1\,) + N + 2\;, \quad \mathcal{F}_N^* = 2\,N(\,log_2\,N - 2\,) + 4\,. \tag{2.7}$$

Ponadto algorytm *in place* nie wymaga rezerwacji pamięci systemu komputerowego dla dodatkowych potrzeb poza jedną zmienną zespoloną służącą do bieżących obliczeń, co czyni rząd jego złożoności pojemnościowej liniowym. Tabela 2.1

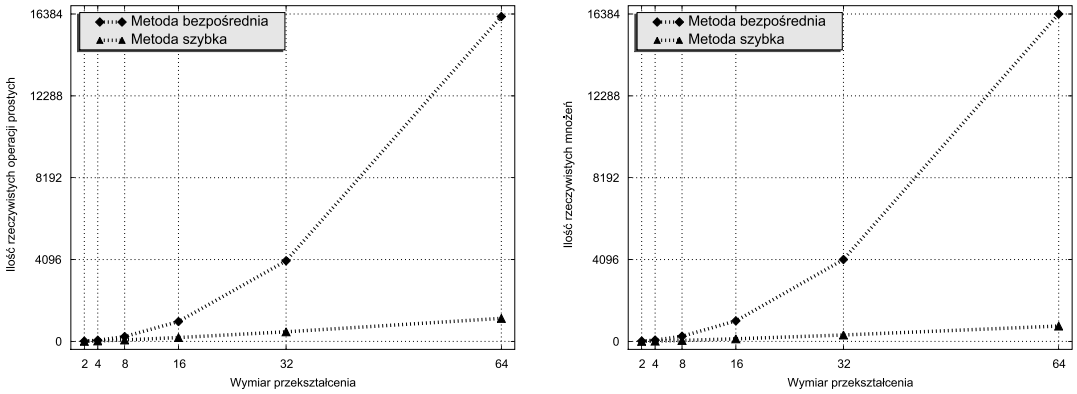
[1] ang. *in-place* `FFT1D`
[2] ang. *bit reversal ordering*

zawiera porównanie ilości rzeczywistych operacji prostych oraz mnożeń dla metody bezpośredniej (2.1) i szybkiej (2.6) wyznaczania N–punktowego przekształcenia Fouriera.

Tab. 2.1: Porównanie liczby rzeczywistych operacji arytmetycznych dla algorytmów `DFT1D` i `FFT1D`

Algorytm	*Ilość rzeczywistych operacji arytmetycznych*	
	operacje proste	*mnożenia*
bezpośredni	$4N^2 - 2N$	$4N^2$
szybki	$3\,N\,(\,log_2\,N - 1\,) + N + 2$	$2\,N\,(\,log_2\,N - 2\,) + 4$

Rysunek 2.2 obrazuje przewagę efektywnościową algorytmu szybkiego (2.6) nad metodą bezpośrednią (2.1) obliczania dyskretnej transformaty Fouriera zarówno względem ilości rzeczywistych operacji prostych jak i mnożeń na przykładzie kilku wybranych przekształceń o zadanych wymiarach.



Rys. 2.2: Porównanie ilości operacji rzeczywistych dla metod bezpośredniej `DFT1D` i szybkiej `FFT1D`

Jak widać z powyższego rysunku już przy niewielkich rozmiarach sygnałów wejściowych przewaga efektywnościowa metody szybkiej wyznaczania współczynników rozważanej transformaty nad bezpośrednią procedurą obliczeniową jest bardzo duża. Wszystkie prezentowane w niniejszym rozdziale algorytmy wyznaczania przekształceń dyskretnych, jak również te nowo zaproponowane, oparte są właśnie na pomyśle [19] rekursywnego podziału zadania wejściowego na dwa zadania o dwa razy mniejszym rozmiarze zwanym strategią *dziel i rządź*.[1] Przejdźmy do omówienia szybkiej metody dwuwymiarowej.

[1] ang. – *divide and conquer*

2.2.2. Szybki algorytm dla transformaty dwuwymiarowej

W niniejszym podpunkcie zostanie wyprowadzony i poddany analizie efektywnościowej szybki algorytm wyznaczania (prostego) dwuwymiarowego dyskretnego przekształcenia Fouriera. Jak wspomniano we wstępie do bieżącego rozdziału przedstawiona tu zasada konstrukcji metody szybkiej, choć dotyczy przypadku konkretnego rodzaju przekształcenia, ma charakter uniwersalny, umożliwiający jej bezpośrednie zastosowanie do wszystkich rozważanych dalej przekształceń dwuwymiarowych. Można wykazać [17], że dwuwymiarowa dyskretna transformata Fouriera jest przekształceniem separowalnym. Innymi słowy jądro $M \times N$–punktowej transformaty Fouriera daje się wyrazić poprzez macierze przekształceń jednowymiarowych za pomocą produktu Kroneckera tych ostatnich w podany niżej sposób:

$$\mathbf{F}_{M\times N} \;=\; \mathbf{F}_N \otimes \mathbf{F}_M \tag{2.8}$$

Posługując się dobrze znaną z rachunku macierzowego produktów Kroneckera własnością[1] orzekającą, że dla dowolnych macierzy zespolonych (rzeczywistych) o zgodnych wymiarach prawdziwa jest zależność:

$$(\,\mathbf{A} \otimes \mathbf{B}\,)\, vec\,(\,\mathbf{C}\,) \;=\; vec\;(\;\mathbf{B}\,\mathbf{C}\,\mathbf{A}^T) \tag{2.9}$$

można wysnuć wniosek, że dla każdej pary $M \times N$–elementowych macierzy zespolonych $\mathbf{X}$ i $\mathbf{Y}$, z których ostatnia jest wynikiem przyłożenia $M \times N$–punktowego dwuwymiarowego dyskretnego przekształcenia Fouriera do pierwszej z podanych macierzy, prawdziwa jest następująca równość:

$$\begin{aligned}\mathbf{Y} \;=\; F_{M\times N}(\,\mathbf{X}\,) \;\Longrightarrow\; \mathbf{Y} \;=\; vec^{-1}_{(M)}\,\{\,\mathbf{F}_{M\times N}\, vec\,(\,\mathbf{X}\,)\,\} \;= \\ \overset{(2.8)}{=}\; vec^{-1}_{(M)}\,\{\,(\,\mathbf{F}_N \otimes \mathbf{F}_M\,)\, vec\,(\,\mathbf{X}\,)\,\} \overset{(2.9)}{=} vec^{-1}_{(M)}\,\{\,vec\,(\,\mathbf{F}_M\,\mathbf{X}\;\mathbf{F}_N^T)\,\} \;=\; \mathbf{F}_M\,\mathbf{X}\;\mathbf{F}_N^T\end{aligned} \tag{2.10}$$

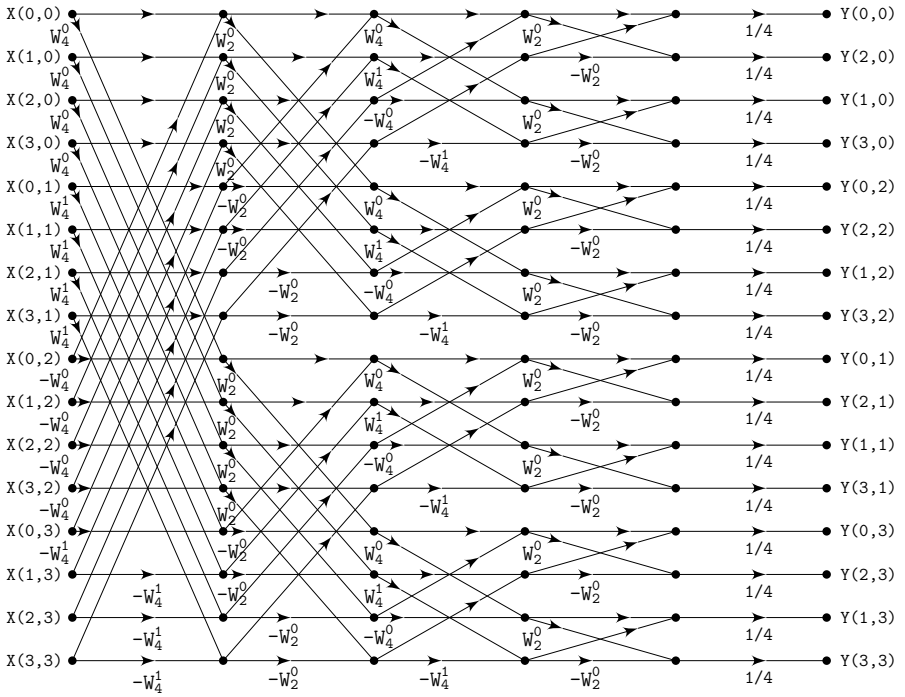
gdzie $vec^{-1}_{(\,\cdot\,)}\{\,\cdot\,\}$ jest operatorem dewektoryzacji macierzy. Na mocy dowolności wyboru wymiarów rozważanego wyżej przekształcenia oraz macierzy $\mathbf{X}$ równość (2.10) stwierdza, bardzo użyteczny z implementacyjnego punktu widzenia fakt, iż każdą $M \times N$–punktową dwuwymiarową dyskretną transformatę Fouriera dowolnego argumentu można wyznaczyć poprzez wstępne obliczenie (N–elementowych) przekształceń jednowymiarowych dla każdego z wierszy macierzy wejściowej, a następnie przyłożenie (M–elementowych) transformat jednowymiarowych do każdej z kolumn powstałej w ten sposób macierzy pośredniej.[2]

Opisana przed chwilą procedura kalkulacyjna stosowana dla dwuwymiarowych transformat separowalnych określana jest potocznie mianem techniki *wiersze – kolumny* obliczania współczynników wspomnianych przekształceń i pozwala natychmiast zredukować złożoność rachunkową oryginalnego problemu ich wyznaczania

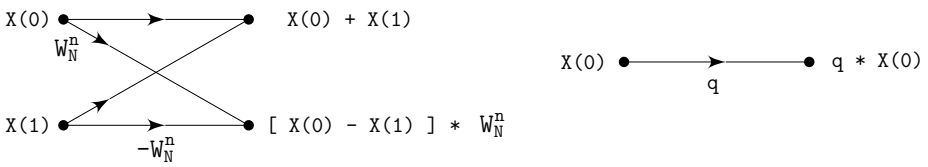
[1] patrz także twierdzenie T2.13 na str. 773 w pracy [91]

[2] własność łączności mnożenia macierzy pozwala również w tym przypadku na wykonanie wskazanych operacji w kolejności odwrotnej

względem odpowiadających im algorytmów bezpośrednich, bez odwoływania się nawet do jednowymiarowych metod szybkich. Dla $N \in \mathbb{N}$ będącego dowolną, naturalną potęgą liczby dwa, algorytm wyznaczania $N \times N$–punktowego dwuwymiarowego dyskretnego przekształcenia Fouriera ustalonej macierzy wejściowej $\mathbf{X} \in \mathbb{C}^{N \times N}$, polegający na wykorzystaniu przedstawionej wyżej techniki (2.10) wiersze – kolumny przy jednoczesnym użyciu metody szybkiej (2.4) obliczania jednowymiarowych N–punktowych transformat Fouriera względem każdego z wierszy macierzy wejściowej i, sukcesywnie, każdej z kolumn otrzymanej w ten sposób macierzy pośredniej, określany jest mianem szybkiej dwuwymiarowej transformaty Fouriera i oznaczy skrótowo zapisem `FFT2D`. Jak wynika natychmiast z konstrukcji takiego sposobu wyznaczania dwuwymiarowego dyskretnego przekształcenia Fouriera złożoność rachunkowa tej techniki obliczeniowej jest rónież rzędu $\mathcal{O}(M \log_2 M)$, gdzie $M = N^2$ jest całkowitą ilością elementów macierzy roboczych podlegających przetwarzaniu przez rozważane dyskretne przekształcenie dwuwymiarowe.



Rys. 2.3a: Szybki algorytm obliczania 4×4–punktowego dyskretnego przekształcenia Fouriera `FFT2D`



Rys. 2.3b: Operacje podstawowe algorytmu `FFT2D`; $X(0)$, $X(1)$, $q \in \mathbb{C}$; $W_N^n \overset{(2.5)}{=} e^{-j2\pi n/N}$

Jedną z ostatnich kwestii, którą autor chciał poruszyć w bieżącym podpunkcie

w ramach przykładu przekształcenia Fouriera, jest zagadnienie aranżacji schematów obliczeniowych dwuwymiarowych procedur szybkich, tak aby w sensie strukturalnym odpowiadały one dokładnie grafom przepływu danych stosownych algorytmów jednowymiarowych. W przypadku większości znanych, szybkich metod obliczeniowych dla dwuwymiarowych, separowalnych przekształceń dyskretnych analogiczna aranżacja jest możliwa i niewątpliwie bardzo korzystna z technicznego punktu widzenia, gdyż pozwala na implementację obydwu wariantów (jedno i dwuwymiarowego) wybranego przekształcenia dyskretnego za pomocą szybkich schematów kalkulacyjnych o identycznych strukturach grafowych, zmniejszając tym samym stopień komplikacji i kosztów implementacyjnych obu wersji rozpatrywanych procedur. Jedyną, i główną zarazem, trudnością takiego podejścia do realizacji dwuwymiarowych metod szybkich, pojawiającą się aczkolwiek tylko we wstępnym etapie ich projektowania, jest znalezienie odpowiedniej wejściowej/wyjściowej permutacji współczynników ich macierzy operacyjnych i związanego z nią sposobu mapowania wierszowo kolumnowego, dwuwymiarowego, szybkiego schematu obliczeniowego na strukturę grafu przepływu danych odpowiadającą pojedynczemu algorytmowi jednowymiarowemu. Poniższy rysunek ilustruje istotę rozważanego tu zagadnienia, ukazując na przykładzie 4×4–punktowej dwuwymiarowej dyskretnej transformaty Fouriera implementację (wierszowo – kolumnowej) metody (2.10) `FFT2D` o identycznej strukturze grafu przepływu danych, co szybki algorytm jednowymiarowy `FFT1D`, określony zależnościami (2.6) i przedstawiony w formie grafu w poprzednim podpunkcie na rysunku 2.1a.

Na koniec łatwo stwierdzić, uwzględniając pokazane wyżej postaci grafów przepływu danych przekształceń dwuwymiarowych Fouriera i biorąc pod uwagę identyczną argumentację co w przypadku transformat jednowymiarowych, że zapotrzebowanie pamięciowe omawianych tu dwuwymiarowych algorytmów szybkich jest także liniowe względem całkowitego wymiaru wybranego przekształcenia. Kolejny podpunkt dotyczyć będzie zagadnienia konstrukcji jedno i dwuwymiarowych szybkich algorytmów dla przekształceń odwrotnych Fouriera.

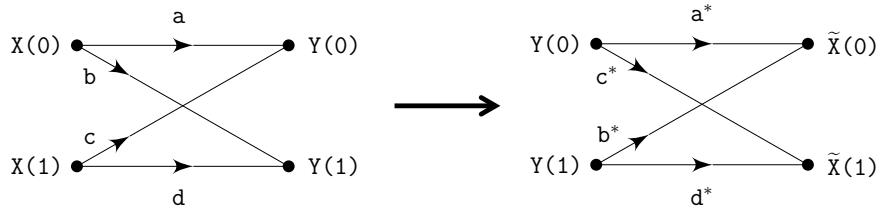
2.2.3. Szybkie algorytmy dla przekształceń odwrotnych

Zagadnienie konstrukcji szybkich algorytmów wyznaczania przekształceń odwrotnych stanowi równie istotny problem, co projektowanie metod dla ich prostych odpowiedników. W niniejszym podpunkcie, na przykładzie przekształcenia Fouriera, podane zostaną ogólne zasady budowy wspomnianych metod, nadające się do bezpośredniego zastosowania w przypadku pozostałych transformat rozważanych dalej w niniejszej pracy. Zapisując macierzowo postaci przekształceń odpowiednio Fouriera (2.1a) oraz jego skalowanego odpowiednika (2.3) w kontekście iteracyjnego schematu obliczeniowego – rys. 2.1a, algorytmu (2.6) szybkiego `FFT1D` widać, że jądro N – punktowej prostej transformaty Fouriera można przedstawić w formie iloczynu trzech $N \times N$ – elementowych macierzy składowych w następujący sposób: $\mathbf{F}_N = \mathbf{P}_N \mathbf{S}_N \widetilde{\mathbf{F}}_N$, gdzie $\mathbf{P}_N$ jest (symetryczną) macierzą permutacji odwróconego porządku bitowego, $\mathbf{S}_N \stackrel{\Delta}{=} diag\,(\,1/\sqrt{N},\,\ldots\,,\,1/\sqrt{N}\,)$ jest skalującą macierzą diagonalną, zaś $\widetilde{\mathbf{F}}_N$ reprezentuje „część motylkową” grafu przepływu danych z rysunku 2.1a implementacji iteracyjnej algorytmu `FFT1D`, bez uwzględnienia ostatniego etapu skalującego.

Zapisując formalnie powyższe wnioski otrzymujemy:

$$\mathbf{F}_N^{-1} \underset{(2.1b)}{\overset{(2.1a)}{=}} \mathbf{F}_N^H \;\wedge\; \mathbf{F}_N = \mathbf{P}_N \mathbf{S}_N \widetilde{\mathbf{F}}_N \;\Rightarrow\; \mathbf{F}_N^{-1} = (\,\mathbf{P}_N \mathbf{S}_N \widetilde{\mathbf{F}}_N\,)^H = \widetilde{\mathbf{F}}_N^{\,H} \mathbf{S}_N \mathbf{P}_N \quad (2.11)$$

Z rysunku 2.1a wynika, że macierz „obszaru motylkowego" $\widetilde{\mathbf{F}}_N$ iteracyjnego schematu obliczeniowego algorytmu szybkiego `FFT1D` można zapisać jako iloczyn log_2N zespolonych macierzy $N \times N$–elementowych $\mathbf{A}_k$, $k = 1, \ldots, log_2N$ o strukturze motylkowej każda, reprezentujących poszczególne etapy rozważanej procedury iteracyjnej, innymi słowy zachodzi następująca równość: $\widetilde{\mathbf{F}}_N = \prod_{k=1}^{log_2N} \mathbf{A}_k$. Wobec tego, sprzężenie hermitowskie macierzy $\widetilde{\mathbf{F}}_N$ ma postać: $\widetilde{\mathbf{F}}_N^H = \mathbf{A}_{log_2N}^H \cdot \mathbf{A}_{log_2N-1}^H \cdot \ldots \cdot \mathbf{A}_1^H$. Łatwo dociec, że każda z cząstkowych macierzy sprzężonych $\mathbf{A}_k^H$, $k = 1, \ldots, log_2N$ przekształcenia $\widetilde{\mathbf{F}}_N^H$ ma identyczną strukturę motylkową, co jej prosty odpowiednik $\mathbf{A}_k$ będący składnikiem macierzy $\widetilde{\mathbf{F}}_N$. A zatem nietrudno też stwierdzić, że zachowując strukturę motylkową ustalonej macierzy składowej $\mathbf{A}_n$, $n \in \{1, \ldots, log_2N\}$ i stosując dla każdej czwórki jej elementów, odpowiadających pojedynczej operacji motylkowej, zamianę współczynników zgodną ze schematem z rys. 2.4 uzyskujemy hermitowsko sprzężony odpowiednik $\mathbf{A}_n^H$ rozważanej macierzy $\mathbf{A}_n$.



Rys. 2.4: Sposób zamiany współczynników pojedynczej operacji motylkowej

Wobec tego postać macierzy $\widetilde{\mathbf{F}}_N^{\,H}$ odpowiada po prostu odwróconemu schematowi motylkowemu macierzy $\widetilde{\mathbf{F}}_N$ z odpowiednio zamienionymi współczynnikami operacji podstawowych. Biorąc pod uwagę powyższy wniosek można orzec, że aby uzyskać graf przepływu danych odwrotnego jednowymiarowego przekształcenia Fouriera wystarczy odwrócić strukturę schematu obliczeniowego (rys. 2.1a) odpowiedniego przekształcenia prostego i zamienić współczynniki wszystkich operacji bazowych powstałego w ten sposób grafu przepływowego zgodnie z regułą podaną na rysunku 2.4. W podany wyżej sposób, dysponując grafem przepływu danych algorytmu prostego przekształcenia jedno lub dwuwymiarowego uzyskuje się graf przepływowy algorytmu odpowiedniego przekształcenia odwrotnego o identycznych charakterystykach wydajnościowo-pojemnościowych, co jego prosty odpowiednik. Na koniec warto jeszcze raz wspomnieć, iż metoda ta może być zastosowana do dowolnej pary zespolonych lub rzeczywistych przekształceń dyskretnych, o ile tylko dysponujemy odpowiednim grafem przepływu danych transformaty prostej.

Rozważania przedstawione w bieżącym podrozdziale dotyczyły aspektów konstrukcji i efektywności obliczeniowo – pamięciowej szybkich algorytmów kalkulacyjnych dla rodziny dyskretnych transformat Fouriera. Na przykładzie tejże rodziny przekształceń podano typową kolejność działań i metodykę procesu projektowania szybkich algorytmów obliczeniowych przekształceń dyskretnych. W tym miejscu należy jeszcze raz podkreślić fakt, iż choć przedstawiony tu materiał dotyczył aspektów

konstrukcji i ogólnej efektywności algorytmów szybkich konkretnej grupy przekształceń, to analogiczne postępowanie może być z powodzeniem przeniesione w całości na procesy projektowe wydajnych metod obliczeniowych innych rodzin transformat dyskretnych, włączając w to wszystkie metody zaprezentowane w kolejnej części monografii, także te nowo zaproponowane.

2.3. Szybkie zunifikowane transformaty dyskretne

W niniejszym podrozdziale przedstawione i wyprowadzone zostaną szybkie zunifikowane algorytmy wyznaczania wybranych przekształceń dyskretnych, które stanowią bazę konstrukcji sieci neuronowych do kompresji obrazów zaprezentowanych w monografii. Następnie algorytmy te poddane zostaną teoretycznej analizie efektywnościowo pojemnościowej pozwalającej na późniejszą weryfikację, względem wymienionych wyżej kryteriów, metod neuronowych opartych o ich wykorzystanie.

2.3.1. Szybkie algorytmy zunifikowane przekształceń jednowymiarowych

Oryginalny pomysł [85], [86] zunifikowanych algorytmów szybkich wyznaczania przekształceń dyskretnych opiera się na fakcie bliskich relacji matematycznych zachodzących pomiędzy wieloma znanymi rodzinami dyskretnych przekształceń liniowych. W odróżnieniu jednak od tradycyjnego podejścia zespalającego różne metody obliczania wybranych przekształceń dyskretnych i bazującego na szybkich procedurach obliczeniowych dla dyskretnych transformat Fouriera [18] czy Hartleya [92] jako przekształceniach stosunkowo najbardziej ogólnych, algorytmy zunifikowane prezentowane w niniejszej pracy za bazę konstrukcji swoich metod obliczeniowych obierają dyskretne przekształcenie kosinusowe, które jest z kolei podstawową transformatą ortogonalną wykorzystywaną w procesie kompresji obrazów.

Wychodząc od szybkiego algorytmu dwuetapowego [28, 85, 88] dla bazowego jednowymiarowego przekształcenia kosinusowego można w dalszej perspektywie skonstruować algorytmy obliczeniowe dla innych znanych jednowymiarowych transformat ortogonalnych, takich jak transformata sinusowa czy wspomniane przekształcenia Hartleya lub Fouriera, o identycznej strukturze grafów przepływu danych dla każdej z analizowanych transformat. Na tej podstawie uwzględniając liniowość oraz separowalność dwuwymiarowych przekształceń dyskretnych, z których ostatnia cecha jest immanentną własnością większości znanych rodzin dwuwymiarowych przekształceń dyskretnych, można rozszerzyć opisane wyżej podejście konstrukcyjne dla rozważanych tu algorytmów na przypadek dwuwymiarowy, i w analogiczny sposób dokonać syntezy zunifikowanych metod szybkich wyznaczania dyskretnych przekształceń dwuwymiarowych bazujących na szybkiej dwuetapowej procedurze obliczania dwuwymiarowego dyskretnego przekształcenia kosinusowego.

Opisany wyżej proces konstrukcji zunifikowanych algorytmów szybkich wyznaczania jedno i dwuwymiarowych przekształceń dyskretnych zostanie szczegółowo prześledzony w niniejszej części pracy, dowodząc uniwersalnego i efektywnego zarazem charakteru prezentowanych tu procedur obliczeniowych, umożliwiającego

w konsekwencji syntezę architektury szybkich sieci neuronowych do kompresji obrazów. Zgodnie z opisanym wyżej przebiegiem kolejnych etapów konstrukcji jednolitych strukturalnie, szybkich algorytmów wyznaczania przekształceń dyskretnych należy najpierw zająć się zagadnieniem budowy procedur obliczeniowych dla przekształceń jednowymiarowych stanowiących bazowy komponent prezentowanej tu rodziny algorytmów. Na wstępie analizie poddany zostanie podstawowy algorytm zunifikowany wyznaczania jednowymiarowej transformaty kosinusowej.

2.3.1.1. Szybkie zunifikowane jednowymiarowe dyskretne przekształcenie kosinusowe

W niniejszym paragrafie zaprezentowany zostanie szybki dwuetapowy algorytm wyznaczania dyskretnej transformaty kosinusowej, który jak wspomniano wcześniej, stanowi podstawę konstrukcji rozważanych dalej zunifikowanych algorytmów szybkich obliczania pozostałych, przykładowych jedno i dwuwymiarowych, przekształceń dyskretnych, tzn. transformat sinusowej, Hartleya oraz Fouriera.

Zanim jednak uwaga zostanie skoncentrowana na głównym temacie tej części pracy warto dokonać wstępnego przeglądu podstawowych własności wybranych wyrażeń trygonometrycznych przydatnych w dalszej części wywodu. Podajmy dobrze znane własności dla funkcji trygonometrycznych sumy i różnicy kątów:

$$\forall\, \alpha, \beta \in \mathbb{R} \qquad sin\,[\,\alpha \pm \beta\,] = sin\,\alpha\, cos\,\beta \pm cos\,\alpha\, sin\,\beta \tag{2.12a}$$

$$\forall\, \alpha, \beta \in \mathbb{R} \qquad cos\,[\,\alpha \pm \beta\,] = cos\,\alpha\, cos\,\beta \mp sin\,\alpha\, sin\,\beta \tag{2.12b}$$

Ponadto dla dowolnego $n \in \mathbb{N}$ z podstawowych własności funkcji *sinus* i *kosinus* mamy:

$$sin\,[\,(\,2n+1\,)\,\pi\,] \,=\, 0\,,\;\; cos\,[\,(\,2n+1\,)\,\pi\,] \,=\, -1 \tag{2.13a}$$

$$sin\left[\,(\,2n+1\,)\,\frac{\pi}{2}\,\right] \,=\, cos\,[\,n\,\pi\,] \,=\, (-1\,)^{n}\,,\; cos\left[\,(\,2n+1\,)\,\frac{\pi}{2}\,\right] \,=\, sin\,[\,n\,\pi\,] \,=\, 0 \tag{2.13b}$$

Korzystając z wymienionych tu własności pokażmy następującą tożsamość, niech $N, k, n \in \mathbb{N}$, mamy:

$$\begin{aligned}
sin\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \,&=\, (-1\,)^{n}\cdot(-1\,)^{n}\,sin\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \,= \\
&\overset{(2.13b)}{=}\; (-1\,)^{n}\,sin\left[\,(\,2n+1\,)\,\frac{\pi}{2}\,\right]\,sin\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \,= \\
&\overset{(2.13b)}{=}\; (-1\,)^{n}\left\{\,cos\left[\,(\,2n+1\,)\,\frac{\pi}{2}\,\right]\,cos\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] + \right. \\
&\quad \left. +\qquad\qquad sin\left[\,(\,2n+1\,)\,\frac{\pi}{2}\,\right]\,sin\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right]\,\right\} \,= \\
&\overset{(2.12b)}{=} (-1\,)^{n}\,cos\left[\,(\,2n+1\,)\,\frac{\pi}{2} \,-\, \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \,= \\
&=\quad (-1\,)^{n}\,cos\left[\,\frac{(\,2\,n+1\,)\,(\,N\,/\,2 \,-\, k\,)\,\pi}{2\,(\,N\,/\,2\,)}\,\right]
\end{aligned} \tag{2.14}$$

Podajmy niżej jeszcze cztery równości wykorzystane podczas dalszych obliczeń, dostajemy:

$$\begin{aligned} cos\left[\frac{(2n+1)(N-k)\pi}{2(N/2)}\right] &= cos\left[(2n+1)\pi - \frac{(2n+1)k\pi}{2(N/2)}\right] = \\ &\overset{(2.12b)}{=} cos[(2n+1)\pi]\, cos\left[\frac{(2n+1)k\pi}{2(N/2)}\right] + \\ + sin[(2n+1)\pi]\, sin\left[\frac{(2n+1)k\pi}{2(N/2)}\right] &\overset{(2.12a)}{=} -cos\left[\frac{(2n+1)k\pi}{2(N/2)}\right] \end{aligned} \tag{2.15a}$$

Kolejna tożsamość ma postać:

$$\begin{aligned} cos\left[\frac{(2n+1)(N/2-(N-k))\pi}{2(N/2)}\right] &= cos\left[\frac{(2n+1)k\pi}{2(N/2)} - (2n+1)\frac{\pi}{2}\right] = \\ \overset{(2.12b)}{=} cos\left[\frac{(2n+1)k\pi}{2(N/2)}\right] cos\left[(2n+1)\frac{\pi}{2}\right] &+ sin\left[\frac{(2n+1)k\pi}{2(N/2)}\right] sin\left[(2n+1)\frac{\pi}{2}\right] = \\ \overset{(2.13b)}{=} (-1)^n sin\left[\frac{(2n+1)k\pi}{2(N/2)}\right] &\overset{(2.14)}{=} cos\left[\frac{(2n+1)(N/2-k)\pi}{2(N/2)}\right] \end{aligned} \tag{2.15b}$$

Na koniec dostajemy:

$$sin\left[\frac{(N-k)\pi}{2N}\right] \overset{(2.12a)}{=} sin\left[\frac{\pi}{2}\right] cos\left[\frac{k\pi}{2N}\right] - cos\left[\frac{\pi}{2}\right] sin\left[\frac{k\pi}{2N}\right] \overset{(2.13b)}{=} cos\left[\frac{k\pi}{2N}\right] \tag{2.15c}$$

$$cos\left[\frac{(N-k)\pi}{2N}\right] \overset{(2.12b)}{=} cos\left[\frac{\pi}{2}\right] cos\left[\frac{k\pi}{2N}\right] + sin\left[\frac{\pi}{2}\right] sin\left[\frac{k\pi}{2N}\right] \overset{(2.13b)}{=} sin\left[\frac{k\pi}{2N}\right] \tag{2.15d}$$

Dla uniknięcia nieścisłości przedstawmy formalną definicję rozważanego przekształcenia dyskretnego.

Definicja 2.2.

Dla dowolnej liczby $N = 2^m, m \in \mathbb{N}$ funkcję $\overline{C}_N : \mathbb{R}^N \to \mathbb{R}^N$ zdefiniowaną w następujący sposób:

$$\forall\, \mathbf{x} \in \mathbb{R}^N \quad \forall\, k \in \{0, \ldots, N-1\} \qquad \overline{C}_N(\mathbf{x})_k \triangleq \sum_{n=0}^{N-1} x_n\, cos\left[\frac{(2n+1)k\pi}{2N}\right] \tag{2.16}$$

nazywać będziemy *N–punktową jednowymiarową skalowaną dyskretną transformatą kosinusową.*

Jak wynika bezpośrednio z równania (2.16) wymieniona transformata jest po prostu przeskalowaną wersją dyskretnego przekształcenia kosinusowego zdefiniowaną dla wektorów rzeczywistych będących dowolną naturalną potęgą liczby dwa. Przejdźmy do wyprowadzenia wzoru rozkładu skalowanej transformaty kosinusowej stanowiącego podstawę dla propozycji dwuetapowego szybkiego algorytmu zunifikowanego obliczania jednowymiarowego dyskretnego przekształcenia kosinusowego.

Niech $k \in \{\, 0, \ldots, N-1 \,\}$ otrzymujemy następujący ciąg równości:

$$\overline{C}_N(\,\mathbf{x}\,)_k \;=\; \sum_{n=0}^{N-1} x_n \; cos\left[\; \frac{(2\,n+1)\,k\,\pi}{2\,N} \;\right] \;=$$

$$= \sum_{n=0}^{N/2-1} x_{2n}\, cos\left[\; \frac{(\,2\,(2\,n)+1\,)\,k\,\pi}{2\,N} \;\right] + \sum_{n=0}^{N/2-1} x_{2n+1}\, cos\left[\; \frac{(\,2\,(2\,n+1\,)+1\,)\,k\,\pi}{2\,N} \;\right] =$$

$$= \sum_{n=0}^{N/2-1} x_{2n}\, cos\left[\; \frac{(\,2\,(2\,n+1\,)-1\,)\,k\,\pi}{2\,N} \;\right] + \sum_{n=0}^{N/2-1} x_{2n+1}\, cos\left[\; \frac{(\,2\,(2\,n+1\,)+1\,)\,k\,\pi}{2\,N} \;\right] =$$

$$= \sum_{n=0}^{N/2-1} x_{2n}\, cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;-\; \frac{k\,\pi}{2\,N} \;\right] + \sum_{n=0}^{N/2-1} x_{2n+1}\, cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;+\; \frac{k\,\pi}{2\,N} \;\right] =$$

$$\overset{(2.12b)}{=} \sum_{n=0}^{N/2-1} x_{2n} \left\{ cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] cos\left[\; \frac{k\,\pi}{2\,N} \;\right] + sin\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] sin\left[\; \frac{k\,\pi}{2\,N} \;\right] \right\} +$$

$$+ \sum_{n=0}^{N/2-1} x_{2n+1} \left\{ cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] cos\left[\; \frac{k\,\pi}{2\,N} \;\right] - sin\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] sin\left[\; \frac{k\,\pi}{2\,N} \;\right] \right\} =$$

$$= \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] \right\} cos\left[\; \frac{k\,\pi}{2\,N} \;\right] + \tag{2.17}$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,-\, x_{2n+1}\,)\, sin\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] \right\} sin\left[\; \frac{k\,\pi}{2\,N} \;\right] =$$

$$\overset{(2.14)}{=} \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\; \frac{(\,2\,n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)} \;\right] \right\} cos\left[\; \frac{k\,\pi}{2\,N} \;\right] +$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (-1\,)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\; \frac{(\,2\,n+1\,)\,(\,N\,/\,2 \,-\, k\,)\,\pi}{2\,(\,N\,/\,2\,)} \;\right] \right\} sin\left[\; \frac{k\,\pi}{2\,N} \;\right]$$

Utwórzmy wektory pomocnicze $\overline{\mathbf{x}}$, $\widetilde{\mathbf{x}} \in \mathbb{R}^{\,N/2}$ zdefiniowane w podany niżej sposób:

$$\forall\, n \in \{\, 0, \ldots, N/\,2-1\,\} \quad \overline{x}_n \;\triangleq\; (\,x_{2n} \,+\, x_{2n+1}\,)\,, \quad \widetilde{x}_n \;\triangleq\; (-1\,)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,) \tag{2.18}$$

Korzystając z zależności (2.17) i (2.18) zapiszmy zerową współrzędną przekształcenia (2.16), dostajemy:

$$\overline{C}_N(\,\mathbf{x}\,)_0 \overset{(2.17)}{=} \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\; \frac{(\,2\,n+1\,)\,0\,\pi}{2\,(\,N\,/\,2\,)} \;\right] \underset{(2.18)}{\overset{(2.16)}{=}} \overline{C}_{N/2}(\,\overline{\mathbf{x}}\,)_0 \tag{2.19a}$$

Podobnie zbadajmy wartość współrzędnej $N/2$ wspomnianego przekształcenia otrzymujemy podaną niżej zależność.

Dostajemy:

$$\overline{C}_N(\,\mathbf{x}\,)_{N/2} \overset{(2.17)}{=} \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\,(\,2n+1\,)\frac{\pi}{2}\,\right] \right\} cos\left[\,\frac{\pi}{4}\,\right] +$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (-1)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,0\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} sin\left[\,\frac{k\,\pi}{2\,N}\,\right] = \tag{2.19b}$$

$$\overset{(2.13b)}{=} \frac{\sqrt{2}}{2} \sum_{n=0}^{N/2-1} (-1)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,0\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \underset{(2.18)}{\overset{(2.16)}{=}} \frac{\sqrt{2}}{2}\, \overline{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_0$$

Dla k–tej współrzędnej przekształcenia (2.16), $k \in \{\,1,\ldots,N\,/\,2-1\,\}$ mamy:

$$\overline{C}_N(\,\mathbf{x}\,)_k \overset{(2.17)}{=} \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} cos\left[\,\frac{k\,\pi}{2\,N}\,\right] +$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (-1)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,(\,N\,/\,2 \,-\, k\,)\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} sin\left[\,\frac{k\,\pi}{2\,N}\,\right] =$$

$$\underset{(2.18)}{\overset{(2.16)}{=}} \overline{C}_{N/2}(\,\overline{\mathbf{x}}\,)_k\, cos\left[\,\frac{k\,\pi}{2\,N}\,\right] + \overline{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_{N/2-k}\, sin\left[\,\frac{k\,\pi}{2\,N}\,\right] \tag{2.19c}$$

Podobnie, dla dowolnego $k \in \{\,1,\ldots,N\,/\,2-1\,\}$, $N-k$–ta współrzędna (2.16) wynosi:

$$\overline{C}_N(\,\mathbf{x}\,)_{N-k} \overset{(2.17)}{=} \left\{ \sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,(\,N-k\,)\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} cos\left[\,\frac{(\,N-k\,)\,\pi}{2\,N}\,\right] +$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (-1)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,(\,N\,/\,2 \,-\, (\,N-k\,)\,)\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} sin\left[\,\frac{(\,N-k\,)\,\pi}{2\,N}\,\right] =$$

$$\overset{(2.15a-d)}{=} \left\{ -\sum_{n=0}^{N/2-1} (\,x_{2n} \,+\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,k\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} sin\left[\,\frac{k\,\pi}{2\,N}\,\right] +$$

$$+ \left\{ \sum_{n=0}^{N/2-1} (-1)^n \cdot (\,x_{2n} \,-\, x_{2n+1}\,)\, cos\left[\,\frac{(\,2n+1\,)\,(\,N\,/\,2 \,-\, k\,)\,\pi}{2\,(\,N\,/\,2\,)}\,\right] \right\} cos\left[\,\frac{k\,\pi}{2\,N}\,\right] = \tag{2.18d}$$

$$\underset{(2.18)}{\overset{(2.16)}{=}} -\overline{C}_{N/2}(\,\overline{\mathbf{x}}\,)_k\, sin\left[\,\frac{k\,\pi}{2\,N}\,\right] + \overline{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_{N/2-k}\, cos\left[\,\frac{k\,\pi}{2\,N}\,\right]$$

Przepisując zależności (2.18) – (2.18d) dla $k \in \{\,0,\ldots,N\,/\,2-1\,\}$ dostajemy:

$$\widetilde{C}_N(\,\mathbf{x}\,)_0 \overset{(2.19a)}{=} \widetilde{C}_{N/2}(\,\overline{\mathbf{x}}\,)_0 \tag{2.19a}$$

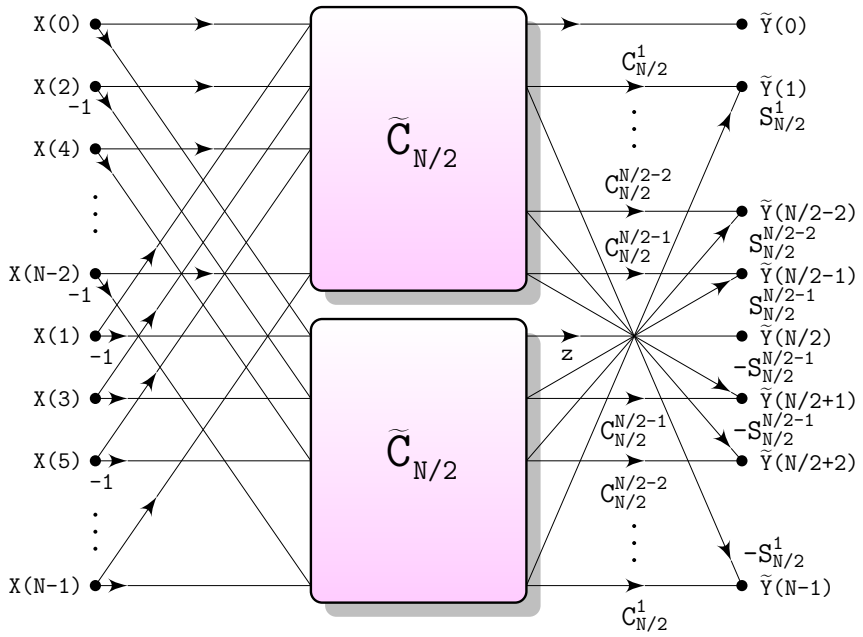
$$\widetilde{C}_N(\,\mathbf{x}\,)_{N/2} \overset{(2.19b)}{=} \frac{\sqrt{2}}{2} \cdot \widetilde{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_0 \tag{2.19b}$$

$$\widetilde{C}_N(\,\mathbf{x}\,)_k \overset{(2.19c)}{=} \widetilde{C}_{N/2}(\,\overline{\mathbf{x}}\,)_k\, cos\left[\,\frac{k\,\pi}{2\,N}\,\right] + \widetilde{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_{N/2-k}\, sin\left[\,\frac{k\,\pi}{2\,N}\,\right] \tag{2.19c}$$

$$\widetilde{C}_N(\,\mathbf{x}\,)_{N-k} \overset{(2.19d)}{=} -\widetilde{C}_{N/2}(\,\overline{\mathbf{x}}\,)_k\, sin\left[\,\frac{k\,\pi}{2\,N}\,\right] + \widetilde{C}_{N/2}(\,\widetilde{\mathbf{x}}\,)_{N/2-k}\, cos\left[\,\frac{k\,\pi}{2\,N}\,\right] \tag{2.19d}$$

$\overline{\mathbf{x}}, \widetilde{\mathbf{x}} \in \mathbb{R}^N,\ \forall n \in \{0,\ldots,N/2-1\}\ \overline{x}_n = (x_{2n} + x_{2n+1})$**,** $\widetilde{x}_n = (-1)^n (x_{2n} - x_{2n+1})$

Zależności (2.19) stanowią rekursywny wzór rozkładu dla konstrukcji szybkiego zunifikowanego algorytmu dwuetapowego obliczania N–punktowego jednowymiarowego dyskretnego przekształcenia kosinusowego. Algorytm ten określany będzie dalej, od nazwiska autora, skrótowo mianem algorytmu `JFCT1D`. Poniższy rysunek obrazuje interpretację graficzną pojedynczego kroku rekursji wzoru rozkładu (2.19) dla dowolnego N będącego wybraną naturalną potęgą dwójki.



Rys. 2.5: Interpretacja graficzna pojedynczego kroku rekursji algorytmu `JFCT1D`

Schemat 2.1 przedstawia pseudokod szybkiego algorytmu obliczania N–punktowego jednowymiarowego skalowanego dyskretnego przekształcenia kosinusowego metodą `FJCT1D`, wynikający ze wzoru rozkładu (2.19).

Rysunek 2.6a przedstawia graf przepływu danych dla iteracyjnej implementacji *in-place* algorytmu `JFCT1D`, przedstawionego pseudokodem 2.1 dla przykładowego 16–punktowego przekształcenia kosinusowego, z wynikającą ze wzoru rozkładu (2.19) permutacją elementów wektora wejściowego rozważanej wersji algorytmu zgodną
z odwróconym porządkiem bitowym.

Na rysunku 2.6b z kolei widnieją operacje bazowe dla grafu przepływowego. Po analizie pseudokodu rozważanego algorytmu oraz wynikających z niego postaci grafów przepływu danych N–punktowych dyskretnych transformat kosinusowych można dojść do wniosku, że złożoność obliczeniowa procedury `JFCT1D` dla operacji dodawania/odejmowania oraz mnożenia wynosi odpowiednio:

$$\widetilde{\mathcal{C}}_N^{+} \;=\; 2\,N(\,log_2\,N-2\,)+N+3\;\;, \quad \widetilde{\mathcal{C}}_N^{*} \;=\; 2\,N(\,log_2\,N-1\,)+2 \tag{2.20}$$

A zatem jest to procedura szybka, której rząd złożoności obliczeniowej jest równy $\mathcal{O}(Nlog_2N)$, gdzie N jest wymiarem rozważanego przekształcenia. Dodatkowo, jak wy-

nika natychmiast z analizy pseudokodu 2.1 i odpowiadającego mu przykładowego grafu przepływowego pokazanego odpowiednio na rysunkach 2.6a i 2.6b, złożoność

Alg. 2.1: Algorytm `JFCT1D` dla N–punktowej transformaty kosinusowej

$N \in \mathbb{N}$: wymiar transformaty będący naturalną potęgą liczby 2 ;
$\mathbf{x} \in \mathbb{R}^N$: wektor wejściowy ;
$\mathbf{y} \in \mathbb{R}^N$: wektor wyjściowy ;

$\mathbf{x}_N^{(0)} = \mathbf{x}$;
`/* ------------------------ Etap pierwszy -------------------------- */`
dla $M = N$ **do** $M = 2$ **powtarzaj**
  dla $k = 0$ **do** $k = N \,/\, M \,-\, 1$ **powtarzaj**

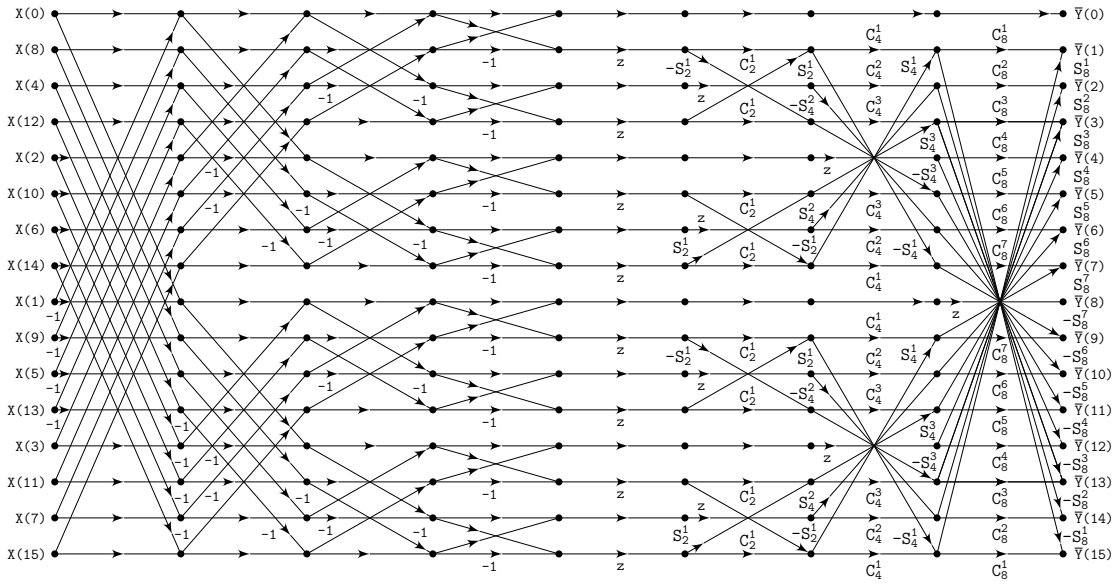
$$\begin{aligned} &\mathbf{x}_{M/2}^{(2k)}(n) = \mathbf{x}_M^{(k)}(2n) + \mathbf{x}_M^{(k)}(2n+1)\,, \\ &\mathbf{x}_{M/2}^{(2k+1)}(n) = (-1)^n\,[\,\mathbf{x}_M^{(k)}(2n) - \mathbf{x}_M^{(k)}(2n+1)\,]\,, \\ &n = 0, 1, \ldots, M \,/\, 2 \,-\, 1\,; \end{aligned} \tag{2.1.1}$$

    $M = M \,/\, 2$;
  koniec pętli
koniec pętli
`/* ------------------- Koniec etapu pierwszego ---------------------- */`
$\mathbf{y}_1^{(k)} = \mathbf{x}_1^{(k)},\ k = 0, 1, \ldots, N \,-\, 1$;
`/* -------------------------- Etap drugi ---------------------------- */`
dla $M = 1$ **do** $M = N \,/\, 2$ **powtarzaj**
  dla $k = 0$ **do** $k = N \,/\, (\,2\,M\,) \,-\, 1$ **powtarzaj**

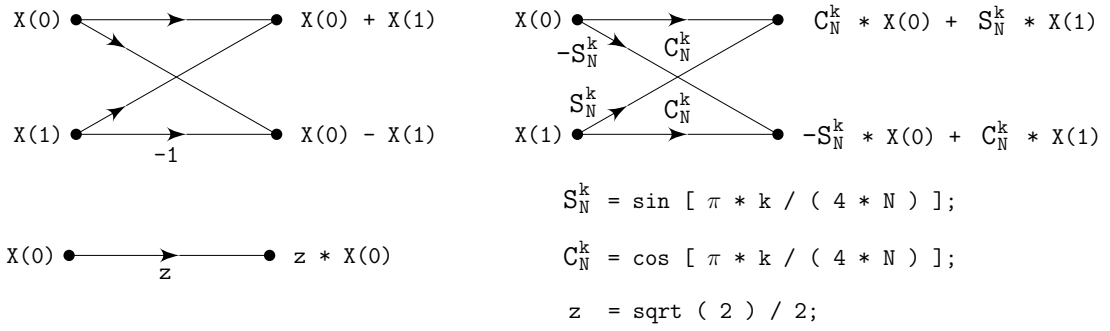
$$\begin{aligned} &\mathbf{y}_{2M}^{(k)}(0) = \mathbf{y}_M^{(2k)}(0)\,,\ \mathbf{y}_{2M}^{(k)}(M) = \sqrt{2}\,/\,2 \cdot \mathbf{y}_M^{(2k+1)}(0)\,, \\ &\mathbf{y}_{2M}^{(k)}(n) = C_M^n \cdot \mathbf{y}_M^{(2k)}(n) \,+\, S_M^n \cdot \mathbf{y}_M^{(2k+1)}(M-n)\,, \\ &\mathbf{y}_{2M}^{(k)}(M-n) = -S_M^n \cdot \mathbf{y}_M^{(2k)}(n) \,+\, C_M^n \cdot \mathbf{y}_M^{(2k+1)}(M-n)\,, \\ &n = 1, 2, \ldots, M \,-\, 1\,; \quad S_N^k = sin\,(\,\pi\,k\,/\,(\,4\,N\,)\,)\,,\ C_N^k = cos\,(\,\pi\,k\,/\,(\,4\,N\,)\,)\,; \end{aligned} \tag{2.1.2}$$

    $M = M \,*\, 2$;
  koniec pętli
koniec pętli
`/* ---------------------- Koniec etapu drugiego ----------------------- */`
$\mathbf{y} = \mathbf{y}_N^{(0)}$;

pamięciowa procedury *in-place* algorytmu (2.19) jest liniowa, a zatem wynosi $\mathcal{O}(N)$. W tym miejscu zakończmy rozważania dotyczące bazowego dla wszystkich pozostałych szybkich procedur zunifikowanych algorytmu `JFCT1D`.



Rys. 2.6a: Szybki algorytm `JFCT1D` obliczania 16–punktowej skalowanej transformaty kosinusowej



Rys. 2.6b: Operacje bazowe algorytmu `JFCT1D` dla dyskretnego przekształcenia kosinusowego

2.3.1.2. Szybkie zunifikowane jednowymiarowe dyskretne przekształcenie sinusowe

W niniejszej części monografii zaprezentowany zostanie szybki zunifikowany algorytm dwuetapowy wyznaczania dyskretnej transformaty sinusowej bazujący na bliskim związku przekształceń sinusowego oraz omówionej w poprzednim paragrafie transformaty kosinusowej. Podobnie jak poprzednio w celu uniknięcia nieścisłości przedstawmy formalną definicję 2.3 rozważanego dalej przekształcenia dyskretnego.

Tak jak w przypadku skalowanej transformaty kosinusowej przedstawione w definicji 2.3 przekształcenie jest po prostu przeskalowaną wersją dyskretnego przekształcenia sinusowego dla wektorów wejściowych o wymiarze będącym potęgą liczby dwa.

Definicja 2.3.

Dla dowolnej liczby $N = 2^m$, $m \in \mathbb{N}$ funkcję $\overline{S}_N : \mathbb{R}^N \to \mathbb{R}^N$ zdefiniowaną w następujący sposób:

$$\forall\, \mathbf{x} \in \mathbb{R}^N \quad \forall\, k \in \{0, \ldots, N-1\} \qquad \overline{S}_N(\mathbf{x})_k \stackrel{\Delta}{=} \sum_{n=0}^{N-1} x_n \; sin\left[\frac{(2n+1)(k+1)\pi}{2N}\right] \tag{2.21}$$

nazywać będziemy *N–punktową jednowymiarową skalowaną dyskretną transformatą sinusową*.

Zgodnie z zapowiedzią poniżej pokazany zostanie fakt nieskomplikowanego związku pomiędzy parą jednowymiarowych przekształceń skalowanych sinusowego i kosinusowego pozwalający na bezpośrednie wykorzystanie opisanego w poprzednim podpunkcie szybkiego algorytmu zunifikowanego `JFCT1D` do wyznaczania współczynników rozważanego tu przekształcenia. Niech $N = 2^m$ oraz $\mathbf{x} \in \mathbb{R}^N$, zdefiniujmy N–elementowy wektor $\widetilde{\mathbf{x}} \in \mathbb{R}^N$ następująco:

$$\forall\, n \in \{0, \ldots, N-1\} \qquad \widetilde{x}_n = (-1)^n x_n \tag{2.22}$$

Wybierzmy teraz dowolne $k \in \{0, \ldots, N-1\}$, otrzymujemy następujący ciąg równości:

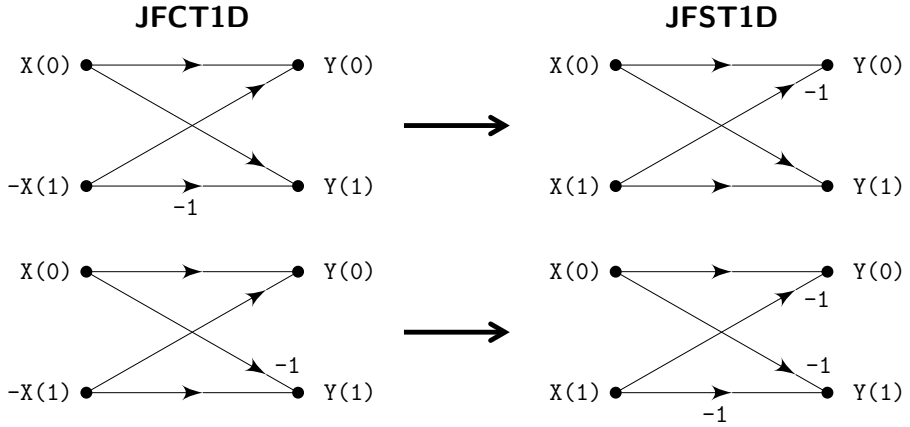
$$\begin{aligned}
\overline{S}_N(\mathbf{x})_k &\stackrel{(2.21)}{=} \sum_{n=0}^{N-1} x_n \; sin\left[\frac{(2n+1)(k+1)\pi}{2N}\right] = \\
&= \sum_{n=0}^{N-1} (-1)^n x_n \left\{(-1)^n sin\left[\frac{(2n+1)(k+1)\pi}{2N}\right]\right\} = \\
&\stackrel{(2.13b)}{=} \sum_{n=0}^{N-1} (-1)^n x_n \left\{sin\left[(2n+1)\frac{\pi}{2}\right] sin\left[\frac{(2n+1)(k+1)\pi}{2N}\right]\right\} = \\
&\stackrel{(2.13b)}{=} \sum_{n=0}^{N-1} (-1)^n x_n \left\{cos\left[(2n+1)\frac{\pi}{2}\right] cos\left[\frac{(2n+1)(k+1)\pi}{2N}\right] + \right. \\
&\qquad\qquad \left. + sin\left[(2n+1)\frac{\pi}{2}\right] sin\left[\frac{(2n+1)(k+1)\pi}{2N}\right]\right\} = \\
&\stackrel{(2.12b)}{=} \sum_{n=0}^{N-1} (-1)^n x_n \, cos\left[(2n+1)\frac{\pi}{2} - \frac{(2n+1)(k+1)\pi}{2N}\right] = \\
&= \sum_{n=0}^{N-1} (-1)^n x_n \, cos\left[\frac{(2n+1)(N-k-1)\pi}{2N}\right] = \\
&\stackrel{(2.16)}{\underset{(2.22)}{=}} \overline{C}_N(\widetilde{\mathbf{x}})_{N-k-1}
\end{aligned} \tag{2.23}$$

A zatem na mocy dowolności wyboru $N = 2^m$, $\mathbf{x} \in \mathbb{R}^N$ oraz $k \in \{0, \ldots, N-1\}$ otrzymujemy wniosek:

$$\forall\, N = 2^m \quad \forall\, \mathbf{x} \in \mathbb{R}^N \quad \forall\, k \in \{0, \ldots, N-1\} \qquad \widetilde{S}_N(\mathbf{x})_k \stackrel{(2.19)}{\underset{(2.23)}{=}} \widetilde{C}_N(\widetilde{\mathbf{x}})_{N-k-1} \tag{2.24}$$

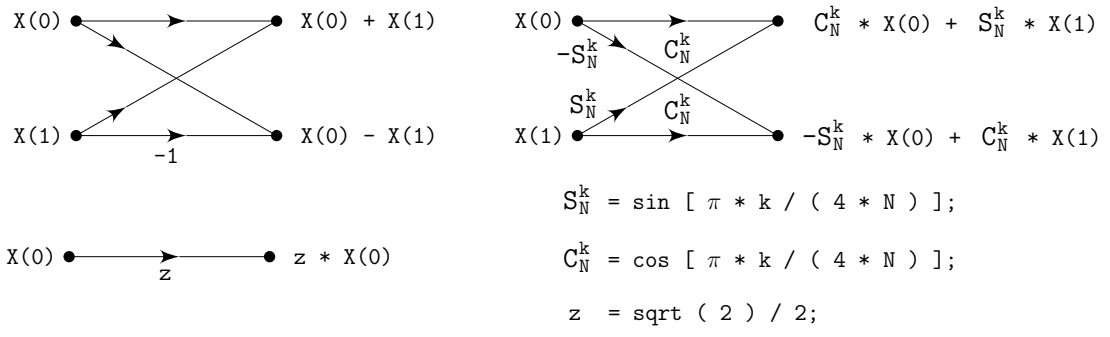
$$\widetilde{\mathbf{x}} \in \mathbb{R}^N, \;\; \forall\, n \in \{0, \ldots, N-1\} \qquad \widetilde{x}_n = (-1)^n x_n$$

Wniosek (2.24) oznacza wprost, że w celu obliczenia współczynników wynikowego N–punktowego dyskretnego skalowanego przekształcenia sinusowego dla dowolnej N–punktowej sekwencji rzeczywistej wystarczy zaburzyć współrzędne wektora wejściowego transformaty sinusowej zgodnie z zależnością (2.22), wyznaczyć N–punktowe



Rys. 2.7: Reguła zamiany współczynników dla pierwszej iteracji algorytmu `JFST1D`

skalowane przekształcenie kosinusowe zmienionej w ten sposób sekwencji wejściowej i zinterpretować (dokonać permutacji) współczynniki tak uzyskanego ciągu wyjściowego w odwrotnej kolejności. Chcąc zatem dokonać efektywnej kalkulacji dyskretnego przekształcenia sinusowego można skorzystać w pełni z gotowej metody szybkiej (2.19) wyznaczania współczynników transformaty kosinusowej odpowiednio



Rys. 2.8: Operacje bazowe algorytmu `JFST1D` dla dyskretnego przekształcenia sinusowego

zaburzając jedynie współrzędne wektora wejściowego. Co więcej, ze względu na prostą formę zależności (2.22) oraz postać grafu przepływu danych (rys. 2.6a) iteracyjnego wariantu metody szybkiej wyznaczania współczynników transformaty kosinusowej, można włączyć zaburzenie (2.22) do pierwszej iteracji wspomnianej metody, tak aby stanowiło ono jej integralną część. Taki sposób obliczania współczynników

skalowanego przekształcenia sinusowego nazywać będziemy dalej metodą `JFST1D`. Regułę zamiany współczynników dla pierwszej iteracji tej metody pokazuje Rys. 2.7.

Alg. 2.2: Algorytm `JFST1D` dla N–punktowej transformaty sinusowej

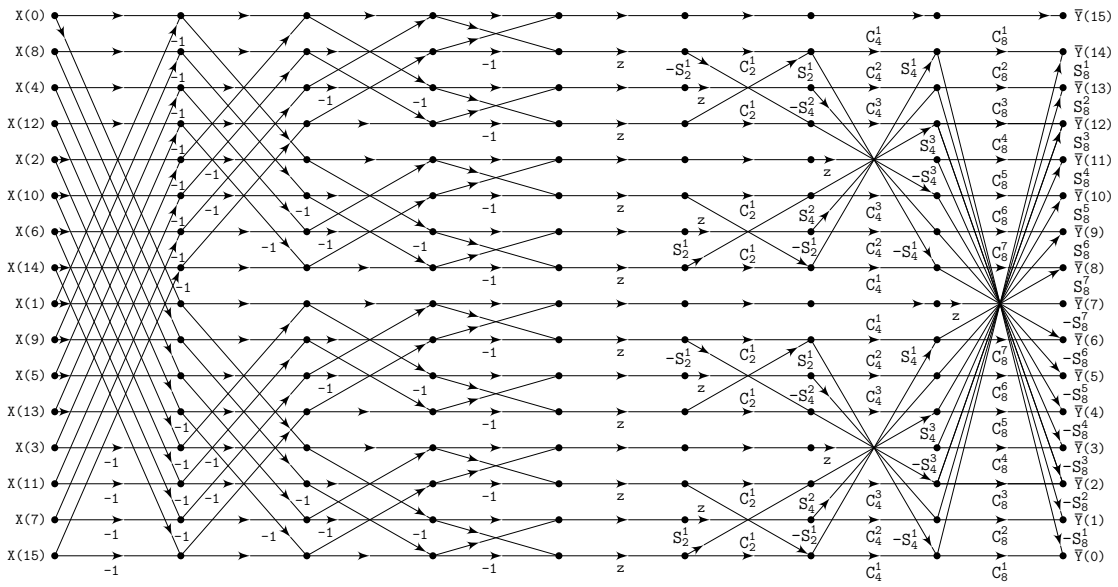
---

$N \in \mathbb{N}$: wymiar transformaty będący naturalną potęgą liczby 2 ;
$\mathbf{x} \in \mathbb{R}^N$: wektor wejściowy ;
$\mathbf{y} \in \mathbb{R}^N$: wektor wyjściowy ;

Utwórz wektor pomocniczy $\widetilde{\mathbf{x}} \in \mathbb{R}^N$ w następujący sposób:
$\forall\, n \in \{0, \ldots, N-1\}\ \ \widetilde{x}_n = (-1)^n x_n$;

Używając szybkiego algorytmu 2.1 obliczania N - punktowego jednowymiarowego dyskretnego przekształcenia kosinusowego za pomocą szybkiego dwuetapowego algorytmu `JFCT1D` wyznacz skalowaną transformatę kosinusową dla wejściowego wektora pomocniczego $\widetilde{\mathbf{x}}$ i oznacz ją przez $\widetilde{\mathbf{y}}$;

Wektor wyjściowy $\mathbf{y}$ określony następująco: $\forall\, n \in \{0, \ldots, N-1\}\ \ y_n = \widetilde{y}_{N-n-1}$ jest skalowanym dyskretnym przekształceniem sinusowym (2.21) wektora wejściowego $\mathbf{x}$;



Rys. 2.9: Szybki algorytm `JFST1D` obliczania 16–punktowej skalowanej transformaty sinusowej

Schemat 2.2 przedstawia pseudokod szybkiego algorytmu obliczania N–punktowego jednowymiarowego skalowanego dyskretnego przekształcenia sinusowego metodą `JFST1D`, wynikający z wniosku (2.24).

Analiza współzależności (2.24) pomiędzy skalowanymi przekształceniami sinusowym i kosinusowym, psedudokod 2.2 oraz postać reguły 2.7 zamiany współczynników pierwszej iteracji algorytmu `JFST1D` i wynikające z niej postaci grafów przepływu danych algorytmów wyznaczania N–punktowych transformat sinusowych skłaniają do natychmiastowego wniosku o fakcie identycznych złożoności obliczeniowych i pojemnościowych obydwu rozważanych algorytmów - `JFCT1D` oraz `JFST1D`. A zatem `JFST1D` jest procedurą szybką, której złożoność obliczeniowa jest równa rzędem $\mathcal{O}(N\,log_2\,N)$, zaś rząd jej złożoności pamięciowej wynosi $\mathcal{O}(N)$. Dla ścisłości podajmy dokładne liczby operacji arytmetycznych prostych i mnożeń dla rozważanego w niniejszym paragrafie algorytmu 2.2, wynoszą one odpowiednio:

$$\widetilde{\mathcal{S}}_N^{+} \;=\; 2\,N(\,log_2\,N-2\,)+N+3\;\;\mathbf{,}\quad \widetilde{\mathcal{S}}_N^{*} \;=\; 2\,N(\,log_2\,N-1\,)+2 \tag{2.25}$$

W tym miejscu zakończmy rozważania dotyczące szybkiego zunifikowanego przekształcenia sinusowego. Następny paragraf dotyczyć będzie szybkiej procedury wyznaczania transformaty Hartleya.

2.3.1.3. Szybkie zunifikowane jednowymiarowe dyskretne przekształcenie Hartleya

W niniejszym paragrafie przedstawiony będzie kolejny szybki zunifikowany algorytm dwuetapowy przeznaczony dla wyznaczania współczynników dyskretnego przekształcenia Hartleya bazujący na bliskim związku tego przekształcenia oraz transformaty kosinusowej. Zanim jednak zajmiemy się wyprowadzeniem głównego związku pomiędzy wspomnianymi przekształceniami warto podać kilka przydatnych własności wyrażeń trygonometrycznych, niech $N=2\,M$, n, $k\in\mathbb{N}$, mamy:

$$\begin{aligned} cos\,[\,n\,\pi\,] \;&=\; cos\,[\,N\,\pi\,]\;cos\,[\,n\,\pi\,]\;+\;sin\,[\,N\,\pi\,]\;sin\,[\,n\,\pi\,]\;\overset{(2.12b)}{=}\;cos\,[\,(\,N-n\,)\,\pi\,]\;= \\ &=\;-\,(\,sin\,[\,(\,N-n\,)\,\pi\,]\;cos\left[\,\frac{\pi}{2}\,\right]\;-\;cos\,[\,(\,N-n\,)\,\pi\,]\;sin\left[\,\frac{\pi}{2}\,\right]\,)\;= \\ &\overset{(2.12b)}{=}\;-\,sin\left[\,(\,N-n\,)\,\pi\;-\;\frac{\pi}{2}\,\right]\;=\;-\,sin\left[\,(\,N-n-1\,)\,\pi\;+\;\frac{\pi}{2}\,\right] \end{aligned} \tag{2.26}$$

Dodatkowo, podstawiając bezpośrednio za N do wzoru (2.14) wartość $2\,N\in\mathbb{N}$ otrzymujemy natychmiast:

$$(-1\,)^n\,sin\left[\,\frac{(\,2\,n+1\,)\,k\,\pi}{2\,N}\,\right]\;\overset{(2.14)}{=}\;cos\left[\,\frac{(\,2\,n+1\,)\,(\,N-\,k\,)\,\pi}{2\,N}\,\right] \tag{2.27}$$

Wreszcie, ostatnie dwie z przydatnych w dalszym ciągu rozważań równości przybierają postaci:

$$\begin{aligned} cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\;&\overset{(2.13b)}{=}\;cos\,[\,2\,k\,\pi\,]\;cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\;+\;sin\,[\,2\,k\,\pi\,]\;sin\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\;= \\ &\overset{(2.12b)}{=}\;cos\left[\,2\,k\,\pi\;-\;\frac{2\,n\,k\,\pi}{N}\,\right]\;=\;cos\left[\,\frac{2\,(\,N-n-1\,)\,k\,\pi}{N}\;+\;\frac{2\,k\,\pi}{N}\,\right] \end{aligned} \tag{2.28a}$$

$$sin\left[\frac{2\,n\,k\,\pi}{N}\right] \overset{(2.13b)}{=} -\left(sin\,[\,2\,k\,\pi\,]\;cos\left[\frac{2\,n\,k\,\pi}{N}\right] - cos\,[\,2\,k\,\pi\,]\;sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) =$$
$$\overset{(2.12a)}{=} -sin\left[2\,k\,\pi - \frac{2\,n\,k\,\pi}{N}\right] = -sin\left[\frac{2\,(\,N-n-1\,)\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right] \tag{2.28b}$$

Dla uniknięcia nieścisłości przedstawmy formalną definicję rozważanego przekształcenia dyskretnego.

Definicja 2.4.

Dla dowolnej liczby $N = 2^{\,m}$, $m \in \mathbb{N}$ funkcję $\overline{H}_N : \mathbb{R}^{\,N} \to \mathbb{R}^{\,N}$ zdefiniowaną w następujący sposób:

$$\forall\,\mathbf{x} \in \mathbb{R}^{\,N}\;\; \forall\,k \in \{\,0,\ldots,N-1\}\;\; \overline{H}_N(\,\mathbf{x}\,)_k \triangleq \sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] + sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) \tag{2.29}$$

nazywać będziemy *N–punktową jednowymiarową skalowaną dyskretną transformatą Hartleya.*

Podobnie jak w przypadku dwóch poprzednich przekształceń transformata $\overline{H}_N$ jest odpowiednio zeskalowaną wersją przekształcenia Hartleya dla wektorów wejściowych o wymiarze będącym potęgą liczby dwa. Poniżej przedstawiony zostanie związek pomiędzy skalowanymi transformatami Hartleya i kosinusową stanowiący bazę dla propozycji szybkiego dwuetapowego algorytmu zunifikowanego obliczania dyskretnego przekształcenia Hartleya. Niech $N = 2^{\,m}$ oraz $\mathbf{x} \in \mathbb{R}^{\,N}$, utwórzmy na podstawie wektora wejściowego $\mathbf{x}$ wektor pomocniczy $\widetilde{\mathbf{x}} \in \mathbb{R}^{\,N}$ zdefiniowany następująco:

$$\forall\,n \in \{\,0,\ldots,N/\,2-1\}\quad \widetilde{x}_{2n} \triangleq x_n\,,\;\; \widetilde{x}_{2n+1} \triangleq x_{N-n-1} \tag{2.30}$$

Korzystając z definicji (2.30) zapiszmy zerową współrzędną przekształcenia (2.29), dostajemy natychmiast:

$$\begin{aligned}\overline{H}_N(\,\mathbf{x}\,)_0 &\overset{(2.29)}{=} \sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,0\,\pi}{N}\right] + sin\left[\frac{2\,n\,0\,\pi}{N}\right]\right) = \sum_{n=0}^{N-1} x_n = \\ &\overset{(2.30)}{=} \sum_{n=0}^{N-1} \widetilde{x}_n = \sum_{n=0}^{N-1} x_n\;cos\left[\frac{(2\,n+1)\,0\,\pi}{2\,N}\right] \overset{(2.16)}{=} \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_0\end{aligned} \tag{2.31a}$$

Podobnie, dla współrzędnej o indeksie $N\,/\,2$ wspomnianego wyżej przekształcenia otrzymujemy:

$$\begin{aligned}\overline{H}_N(\,\mathbf{x}\,)_{N/2} &\overset{(2.29)}{=} \sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\right] + sin\left[\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\right]\right) = \\ &= \sum_{n=0}^{N-1} x_n\;(\,cos\,[\,n\,\pi\,] + sin\,[\,n\,\pi\,]\,) = \sum_{n=0}^{N-1} x_n\,cos\,[\,n\,\pi\,] = \\ &= \sum_{n=0}^{N/2-1} x_n\,cos\,[\,n\,\pi\,] + \sum_{n=N/2}^{N-1} x_n\,cos\,[\,n\,\pi\,] =\end{aligned} \tag{2.31b}$$

$$
\begin{aligned}
&\overset{(2.26)}{=} \sum_{n=0}^{N/2-1} x_n \, cos\left[\, n\,\pi \,\right] \, - \sum_{n=N/2}^{N-1} x_n \, sin\left[\, (\, N-n-1\,)\,\pi \, + \, \frac{\pi}{2} \,\right] \, = \\
&= \sum_{n=0}^{N/2-1} x_n \, cos\left[\, n\,\pi \,\right] \, - \sum_{n=0}^{N/2-1} x_{N-n-1} \, sin\left[\, n\,\pi \, + \, \frac{\pi}{2} \,\right] \, = \\
&\overset{(2.30)}{=} \sum_{n=0}^{N/2-1} \widetilde{x}_{2n} \, cos\left[\, n\,\pi \,\right] \, - \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1} \, sin\left[\, n\,\pi \, + \, \frac{\pi}{2} \,\right] \, = \\
&\overset{(2.13b)}{=} \sum_{n=0}^{N/2-1} \widetilde{x}_{2n} \, \left(cos\left[\, (\, 2\,n\,)\,\frac{\pi}{2} \,\right] \, - \, sin\left[\, (\, 2\,n\,)\,\frac{\pi}{2} \,\right] \right) \, + \\
&+ \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1} \, \left(cos\left[\, (\, 2\,n+1\,)\,\frac{\pi}{2} \,\right] \, - \, sin\left[\, (\, 2\,n+1\,)\,\frac{\pi}{2} \,\right] \right) \, = \\
&= \sum_{n=0}^{N-1} \widetilde{x}_n \, \left(cos\left[\, n\,\frac{\pi}{2} \,\right] \, - \, sin\left[\, n\,\frac{\pi}{2} \,\right] \right) \, = \\
&= \sqrt{2} \sum_{n=0}^{N-1} \widetilde{x}_n \, \left(cos\left[\, n\,\frac{\pi}{2} \,\right] \, cos\left[\, \frac{\pi}{4} \,\right] \, - \, sin\left[\, n\,\frac{\pi}{2} \,\right] \, sin\left[\, \frac{\pi}{4} \,\right] \right) \, = \\
&\overset{(2.12b)}{=} \sqrt{2} \sum_{n=0}^{N-1} \widetilde{x}_n \, cos\left[\, n\,\frac{\pi}{2} \, + \, \frac{\pi}{4} \,\right] \, = \, \sqrt{2} \, \sum_{n=0}^{N-1} \widetilde{x}_n \, cos\left[\, \frac{(2\,n+1)\,(\,N\,/\,2\,)\,\pi}{2\,N} \,\right] \, = \\
&\underset{(2.30)}{\overset{(2.16)}{=}} \sqrt{2} \, \cdot \, \overline{C}_N\,(\,\widetilde{\mathbf{x}}\,)_{\,N/2}
\end{aligned}
\tag{2.31b}
$$

Wybierzmy teraz dowolne $k \in \{\, 0, \ldots, N\,/\,2-1 \,\}$, aby rozpatrzyć łącznie postaci k–tej oraz $N-k$–tej współrzędnej przekształcenia (2.29) pokażmy najpierw poniższą własność, otrzymujemy:

$$
\begin{aligned}
\overline{H}_N\,(\,\mathbf{x}\,)_{\,N-k} &\overset{(2.29)}{=} \sum_{n=0}^{N-1} x_n \, \left(cos\left[\, \frac{2\,n\,(\,N\,-\,k\,)\,\pi}{N} \,\right] \, + \, sin\left[\, \frac{2\,n\,(\,N\,-\,k\,)\,\pi}{N} \,\right] \right) \, = \\
&= \sum_{n=0}^{N-1} x_n \, \left(cos\left[\, 2\,n\,\pi \, - \, \frac{2\,n\,k\,\pi}{N} \,\right] \, + \, sin\left[\, 2\,n\,\pi \, - \, \frac{2\,n\,k\,\pi}{N} \,\right] \, \right) \, = \\
&\overset{(2.12)}{=} \sum_{n=0}^{N-1} x_n \, \left(cos\left[\, 2\,n\,\pi \,\right] \, cos\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \, + \, sin\left[\, 2\,n\,\pi \,\right] \, sin\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \, + \right. \\
&\qquad \left. + \, sin\left[\, 2\,n\,\pi \,\right] \, cos\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \, - \, cos\left[\, 2\,n\,\pi \,\right] \, sin\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \, \right) \, = \\
&\overset{(2.13b)}{=} \sum_{n=0}^{N-1} x_n \, \left(cos\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \, - \, sin\left[\, \frac{2\,n\,k\,\pi}{N} \,\right] \right)
\end{aligned}
\tag{2.31c}
$$

Poniżej, korzystając z podanej przed chwilą zależności (2.31c), wyznaczone zostaną

łącznie postaci współrzędnych k–tej oraz $N-k$–tej przekształcenia Hartleya.

$$\overline{H}_N(\mathbf{x})^{k}_{N-k} \underset{(2.31c)}{\overset{(2.29)}{=}} \sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] \pm sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) =$$

$$\overset{(2.28)}{=} \sum_{n=0}^{N/2-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] \pm sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) +$$

$$+ \sum_{n=N/2}^{N-1} x_n \left(cos\left[\frac{2\,(N-n-1)\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right] \mp sin\left[\frac{2\,(N-n-1)\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right]\right) =$$

$$= \sum_{n=0}^{N/2-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] \pm sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) +$$

$$+ \sum_{n=0}^{N/2-1} x_{N-n-1} \left(cos\left[\frac{2\,n\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right] \mp sin\left[\frac{2\,n\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right]\right) =$$

$$\overset{(2.30)}{=} \sum_{n=0}^{N/2-1} \widetilde{x}_{2n} \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] \pm sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) +$$

$$+ \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1} \left(cos\left[\frac{2\,n\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right] \mp sin\left[\frac{2\,n\,k\,\pi}{N} + \frac{2\,k\,\pi}{N}\right]\right) =$$

$$= \sum_{n=0}^{N/2-1} \widetilde{x}_{2n} \left(cos\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N} - \frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N} - \frac{k\,\pi}{2\,N}\right]\right) +$$

$$+ \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1} \left(cos\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N} - \frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N} - \frac{k\,\pi}{2\,N}\right]\right) =$$

$$\overset{(2.12)}{=} \sum_{n=0}^{N/2-1} \widetilde{x}_{2n} \left\{ \left(cos\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] sin\left[\frac{k\,\pi}{2\,N}\right]\right) \pm \right.$$

$$\left. \pm \left(sin\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] cos\left[\frac{k\,\pi}{2\,N}\right] - cos\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] sin\left[\frac{k\,\pi}{2\,N}\right]\right)\right\} +$$

$$+ \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1} \left\{ \left(cos\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] sin\left[\frac{k\,\pi}{2\,N}\right]\right) \mp \right.$$

$$\left. \mp \left(sin\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] cos\left[\frac{k\,\pi}{2\,N}\right] + cos\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] sin\left[\frac{k\,\pi}{2\,N}\right]\right)\right\}$$

$$= \sum_{n=0}^{N/2-1} \widetilde{x}_{2n}\, cos\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{k\,\pi}{2\,N}\right]\right) \pm$$

$$\pm \sum_{n=0}^{N/2-1} \widetilde{x}_{2n}\, sin\left[\frac{(2\,(2\,n)+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{k\,\pi}{2\,N}\right]\right) +$$

$$+ \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1}\, cos\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{k\,\pi}{2\,N}\right]\right) \mp$$

$$\mp \sum_{n=0}^{N/2-1} \widetilde{x}_{2n+1}\, sin\left[\frac{(2\,(2\,n+1)+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{k\,\pi}{2\,N}\right]\right) = \qquad (2.31d)$$

$$
\begin{aligned}
&= \sum_{n=0}^{N-1} \widetilde{x}_n \, cos\left[\frac{(2\,n+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{k\,\pi}{2\,N}\right]\right) \pm \\
&\pm \sum_{n=0}^{N-1} (-1)^n \cdot \widetilde{x}_n \, sin\left[\frac{(2\,n+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{k\,\pi}{2\,N}\right]\right) = \\
&\overset{(2.27)}{=} \sum_{n=0}^{N-1} \widetilde{x}_n \, cos\left[\frac{(2\,n+1)\,k\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{k\,\pi}{2\,N}\right]\right) \pm \\
&\pm \sum_{n=0}^{N-1} \widetilde{x}_n \, cos\left[\frac{(2\,n+1)\,(\,N\,-\,k\,)\,\pi}{2\,N}\right] \left(cos\left[\frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{k\,\pi}{2\,N}\right]\right) = \qquad (2.31d) \\
&\underset{(2.30)}{\overset{(2.16)}{=}} \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] \mp sin\left[\frac{k\,\pi}{2\,N}\right]\right) \pm \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] \pm sin\left[\frac{k\,\pi}{2\,N}\right]\right)
\end{aligned}
$$

Przepisując zależności (2.30) – (2.31d) w krótszej formie dla $k \in \{\,0, \ldots, N\,/\,2-1\,\}$ otrzymujemy:

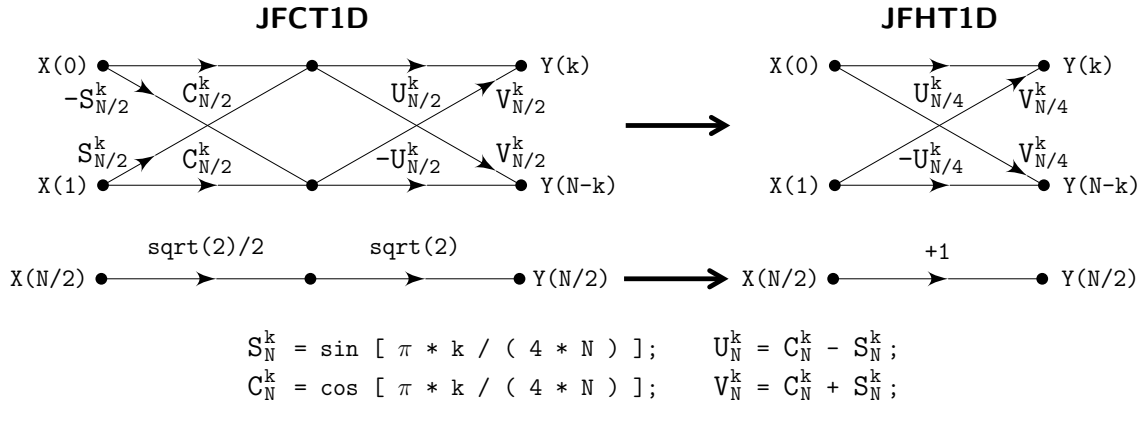
$$
\begin{aligned}
\widetilde{H}_N(\,\mathbf{x}\,)_0 &\overset{(2.31a)}{=} \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_0 \\
\widetilde{H}_N(\,\mathbf{x}\,)_{N/2} &\overset{(2.31b)}{=} \sqrt{2} \cdot \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_{N/2} \qquad (2.32)
\end{aligned}
$$

$$
\widetilde{H}_N(\,\mathbf{x}\,)_k \overset{(2.31d)}{=} \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right) + \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right)
$$

$$
\widetilde{H}_N(\mathbf{x})_{N-k} \overset{(2.31d)}{=} \widetilde{C}_N(\widetilde{\mathbf{x}})_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right) - \widetilde{C}_N(\widetilde{\mathbf{x}})_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right)
$$

$$
\mathbf{x},\, \widetilde{\mathbf{x}} \in \mathbb{R}^N, \;\; \forall\, n \in \{\,0, \ldots, N/\,2-1\} \;\; \widetilde{x}_{2n} \,=\, x_n \,, \;\; \widetilde{x}_{2n+1} \,=\, x_{N-n-1}
$$

Krótka refleksja nad postaciami zależności (2.32) oraz rekursywnego wzoru rozkładu (2.19) dla konstrukcji szybkiego zunifikowanego dwuetapowego algorytmu obliczania



Rys. 2.10: Reguła zamiany współczynników ostatniej iteracji algorytmu `JFHT1D`

N–punktowego jednowymiarowego dyskretnego przekształcenia kosinusowego skłania do wniosku, że algorytm wyznaczania N–punktowego przekształcenia Hartleya

Alg. 2.3: Algorytm `JFHT1D` dla N–punktowej transformaty Hartleya

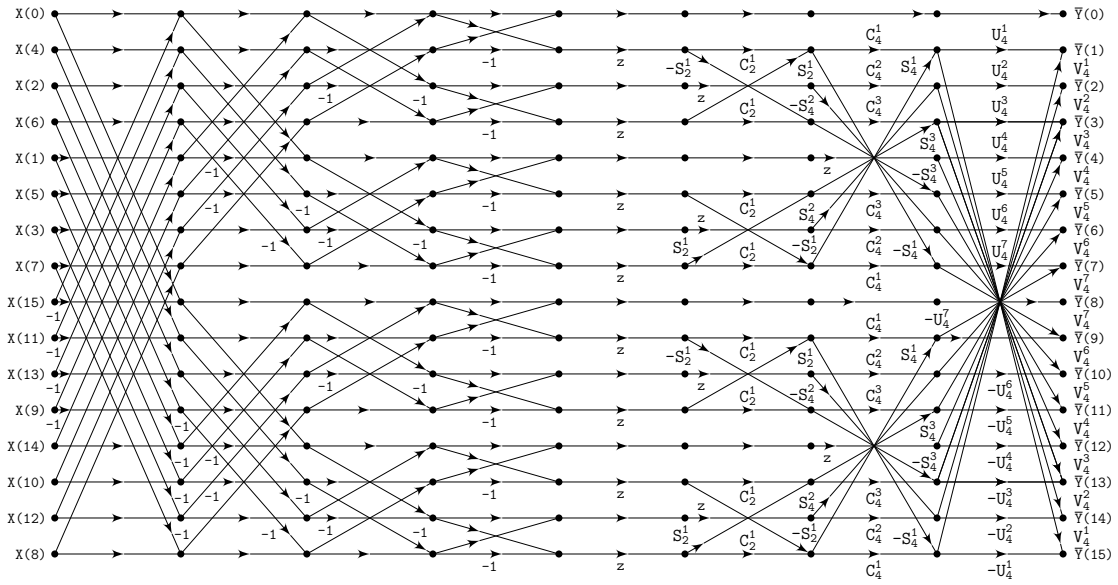
$N \in \mathbb{N}$: wymiar transformaty będący naturalną potęgą liczby 2 ;
$\mathbf{x} \in \mathbb{R}^{\,N}$: wektor wejściowy ;
$\mathbf{y} \in \mathbb{R}^{\,N}$: wektor wyjściowy ;

Utwórz wektor pomocniczy $\widetilde{\mathbf{x}} \in \mathbb{R}^{\,N}$ w następujący sposób:
$\forall\, n \in \{\, 0, \ldots, N/\,2\, -1\}\;\; \widetilde{x}_{2n} = x_n\,,\;\; \widetilde{x}_{2n+1} = x_{N-n-1}$;

Dla pomocniczego wektora wejściowego $\widetilde{\mathbf{x}}$ zastosuj szybki dwuetapowy algorytm 2.1 modyfikując jego działanie w ostatnim kroku etapu drugiego (dla $M = N\,/\,2$ oraz $k = N\,/\,(\,2\,M\,)\, -\, 1\,$) przez zastąpienie wzorów (2.1.2) następującymi zależnościami:

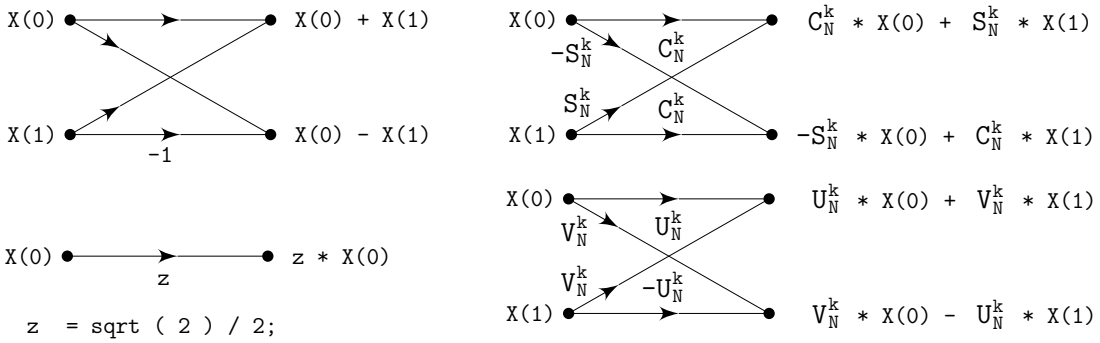
$$
\begin{aligned}
&\mathbf{y}^{(k)}_{2M}(0) = \mathbf{y}^{(2k)}_{M}(0)\,,\;\; \mathbf{y}^{(k)}_{2M}(M) = \mathbf{y}^{(2k+1)}_{M}(0)\,,\\
&\mathbf{y}^{(k)}_{2M}(n) = (\, C^{\,n}_{M/\,2} - S^{\,n}_{M/\,2}\,) \cdot \mathbf{y}^{(2k)}_{M}(n) \;+\; (\, C^{\,n}_{M/\,2} + S^{\,n}_{M/\,2}\,) \cdot \mathbf{y}^{(2k+1)}_{M}(M-n)\,, \qquad (2.3.1)\\
&\mathbf{y}^{(k)}_{2M}(M-n) = (\, C^{\,n}_{M/\,2} + S^{\,n}_{M/\,2}\,) \cdot \mathbf{y}^{(2k)}_{M}(n) \;-\; (\, C^{\,n}_{M/\,2} - S^{\,n}_{M/\,2}\,) \cdot \mathbf{y}^{(2k+1)}_{M}(M-n)\,,\\
&n = 1, 2, \ldots, M\, -\, 1\,; \quad S^{\,k}_{N} = sin\,(\,\pi\, k\,/\,(\,4\,N\,)\,)\,,\;\; C^{\,k}_{N} = cos\,(\,\pi\, k\,/\,(\,4\,N\,)\,)\,;
\end{aligned}
$$

Oznacz powstały w ten sposób wektor wynikowy przez $\mathbf{y}$. Wówczas otrzymany wektor $\mathbf{y}$ jest skalowanym dyskretnym przekształceniem Hartleya wektora wejściowego $\mathbf{x}$;



Rys. 2.11a: Szybki algorytm `JFHT1D` obliczania 16–punktowej skalowanej transformaty Hartleya

może być zrealizowany za pomocą wspomnianej szybkiej metody (2.19), jedynie poprzez zastosowanie odpowiedniej permutacji ciągu wejściowego $\mathbf{x}$ algorytmu (2.19) i modyfikację operacji bazowych na ostatnim etapie działania procedury `JFCT1D`. Taki sposób obliczania współczynników skalowanego przekształcenia Hartleya nazywać będziemy dalej metodą `JFHT1D`. Rysunek 2.10 obrazuje regułę zamiany współczynników dla ostatniej iteracji algorytmu `JFHT1D`. Schemat 2.3 przedstawia pseudokod szybkiego algorytmu obliczania N–punktowego jednowymiarowego skalowanego dyskretnego przekształcenia Hartleya metodą `JFHT1D`, wynikający z zależności (2.32). Rysunek 2.11a przedstawia graf przepływu danych dla algorytmu `JFHT1D` wynikjący z pseudokodu pokazanego na schemacie 2.3 dla przykładowego 16–punktowego przekształcenia Hartleya. Jak widać z rysunku grafu przepływu danych 2.11a różni się on od przykładowego grafu przepływowego 2.6a algorytmu `JFCT1D` jedynie przemieszaniem kolejności elementów wektora wejściowego oraz wartościami współczynników operacji bazowych ostatniego etapu procesu obliczeniowego. Rysunek 2.11b przedstawia operacje podstawowe dla grafu przepływowego 2.11a.




```
z  = sqrt ( 2 ) / 2;
```

C_N^k = cos [π * k / (4 * N)]; S_N^k = sin [π * k / (4 * N)];

U_N^k = cos [π * k / (4 * N)] - sin [π * k / (4 * N)];

V_N^k = cos [π * k / (4 * N)] + sin [π * k / (4 * N)];

Rys. 2.11b: Operacje bazowe algorytmu `JFHT1D` dla przekształcenia Hartleya

Zależności (2.32) między skalowanymi przekształceniami Hartleya i kosinusowym, a także postać reguły 2.10 zamiany współczynników w ostatnim etapie algorytmu `JFHT1D` pozwalają wnioskować, podobnie jak w przypadku transformaty sinusowej, o fakcie identycznych złożoności obliczeniowych i pamięciowych algorytmów `JFCT1D` i `JFHT1D`. Wobec tego `JFHT1D` jest również procedurą szybką o złożoności obliczeniowej równej rzędem $\mathcal{O}(N log_2 N)$, i złożoności pamięciowej rzędu $\mathcal{O}(N)$. Dla uniknięcia nieścisłości podajmy dokładne liczby prostych operacji arytmetycznych i mnożeń dla rozważanego w niniejszym paragrafie algorytmu 2.3, wynoszą one odpowiednio:

$$\widetilde{\mathcal{H}}_N^+ \;=\; 2\,N(\,log_2\,N-2\,)+N+3\;\;\mathbf{,}\quad \widetilde{\mathcal{H}}_N^* \;=\; 2\,N(\,log_2\,N-1\,)+2 \tag{2.33}$$

Ostatnią konkluzją zakończmy już rozważania nad szybkim zunifikowanym przekształceniem Hartleya przechodząc do algorytmu dla transformaty Fouriera.

2.3.1.4. Szybkie zunifikowane jednowymiarowe dyskretne przekształcenie Fouriera

W niniejszym paragrafie zaprezentowany zostanie szybki dwuetapowy algorytm wyznaczania dyskretnej transformaty Fouriera dla rzeczywistych ciągów wejściowych. Podobnie jak w przypadku rozważanych wcześniej przekształceń jednowymiarowych konstrukcja algorytmu bazuje na związku pomiędzy dwoma dyskretnymi przekształceniami skalowanymi, a mianowicie transformatą kosinusową i skalowanym przekształceniem Fouriera dla rzeczywistych wektorów wejściowych. Na wstępie warto podać kilka wybranych własności wyrażeń trygonometrycznych przydatnych dalej, niech $N, n, k \in \mathbb{N}$, mamy:

$$cos\left[\frac{2\,n\,k\,\pi}{N}\right] \overset{(2.13b)}{=} cos\,[\,2\,n\,\pi\,]\;cos\left[\frac{2\,n\,k\,\pi}{N}\right] + sin\,[\,2\,n\,\pi\,]\;sin\left[\frac{2\,n\,k\,\pi}{N}\right] = \overset{(2.12b)}{=} cos\left[2\,n\,\pi - \frac{2\,n\,k\,\pi}{N}\right] = cos\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right] \tag{2.34a}$$

$$sin\left[\frac{2\,n\,k\,\pi}{N}\right] \overset{(2.13b)}{=} -\left(sin\,[\,2\,n\,\pi\,]\;cos\left[\frac{2\,n\,k\,\pi}{N}\right] - cos\,[\,2\,n\,\pi\,]\;sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) = \overset{(2.12a)}{=} -sin\left[2\,n\,\pi - \frac{2\,n\,k\,\pi}{N}\right] = -sin\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right] \tag{2.34b}$$

Podajmy teraz definicję i podajmy podstawowe własności skalowanego dyskretnego przekształcenia Fouriera dla rzeczywistych ciągów wejściowych.

Definicja 2.5.

Dla dowolnej liczby $N = 2^{\,m}$, $m \in \mathbb{N}$ funkcję $\overline{F}_N : \mathbb{R}^{\,N} \to \mathbb{C}^{\,N}$ zdefiniowaną w następujący sposób:

$$\forall \mathbf{x} \in \mathbb{R}^{\,N}\;\forall k \in \{0,\ldots,N-1\}\quad \overline{F}_N\,(\mathbf{x})_k \triangleq \sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] - j\,sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) \tag{2.35}$$

nazywać będziemy *N–punktową jednowymiarową skalowaną dyskretną transformatą Fouriera*.

gdzie $j \in \mathbb{C}$ jest jednostką urojoną. Analogicznie jak w przypadku przekształceń rozważanych w poprzednich paragrafach transformata $\overline{F}_N$ jest odpowiednio przeskalowaną wersją przekształcenia (2.1a) dla rzeczywistych wektorów wejściowych o wymiarze będącym potęgą liczby dwa. Zgodnie z przyjętym założeniem dla dowolnego rzeczywistego ciągu wejściowego $\mathbf{x} \in \mathbb{R}^{\,N}$ z definicji 2.5 otrzymujemy:

$$Re\,\{\,\overline{F}_N(\,\mathbf{x}\,)_k\,\} \overset{(2.34a)}{=} \sum_{n=0}^{N-1} x_n\,cos\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right] \overset{(2.35)}{=} Re\,\{\,\overline{F}_N(\,\mathbf{x}\,)_{N-k}\,\} \tag{2.36a}$$

$$Im\,\{\,\overline{F}_N(\,\mathbf{x}\,)_k\,\} \overset{(2.34b)}{=} \sum_{n=0}^{N-1} x_n\,sin\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right] \overset{(2.35)}{=} -Im\,\{\,\overline{F}_N(\,\mathbf{x}\,)_{N-k}\,\} \tag{2.36b}$$

Jak zapowiedziano wcześniej przedstawiony zostanie teraz związek pomiędzy skalowanymi transformatami Fouriera i kosinusową stanowiący bazę dla propozycji szybkiego dwuetapowego algorytmu zunifikowanego obliczania współczynników dyskretnego przekształcenia Fouriera. Niech $N = 2^m$ oraz $\mathbf{x} \in \mathbb{R}^N$, na podstawie wektora wejściowego $\mathbf{x}$ utwórzmy wektor pomocniczy $\widetilde{\mathbf{x}} \in \mathbb{R}^N$ zdefiniowany w następujący sposób:

$$\forall\, n \in \{\,0, \ldots, N/\,2-1\}\quad \widetilde{x}_{2n} \stackrel{\Delta}{=} x_n\,,\quad \widetilde{x}_{2n+1} \stackrel{\Delta}{=} x_{N-n-1} \tag{2.37}$$

Korzystając z definicji (2.35) zapiszmy zerową współrzędną skalowanego przekształcenia Fouriera, mamy:

$$\begin{aligned} \overline{F}_N(\,\mathbf{x}\,)_0 &\stackrel{(2.35)}{=} \sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,0\,\pi}{N}\,\right] - j\, sin\left[\,\frac{2\,n\,0\,\pi}{N}\,\right]\right) = \\ = \sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,0\,\pi}{N}\,\right] + sin\left[\,\frac{2\,n\,0\,\pi}{N}\,\right]\right) &\stackrel{(2.29)}{=} \overline{H}_N(\,\mathbf{x}\,)_0 \underset{(2.37)}{\overset{(2.32)}{=}} \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_0 \end{aligned} \tag{2.38a}$$

Podobnie, dla współrzędnej o indeksie $N\,/\,2$ rozważanego przekształcenia mamy:

$$\begin{aligned} \overline{F}_N(\,\mathbf{x}\,)_{N/2} &\stackrel{(2.35)}{=} \sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\,\right] - j\, sin\left[\,\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\,\right]\right) = \\ &\stackrel{(2.13b)}{=} \sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\,\right] + sin\left[\,\frac{2\,n\,(\,N\,/\,2\,)\,\pi}{N}\,\right]\right) = \\ &\stackrel{(2.29)}{=} \overline{H}_N(\,\mathbf{x}\,)_{N/2} \underset{(2.37)}{\overset{(2.32)}{=}} \sqrt{2}\,\cdot\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N/2} \end{aligned} \tag{2.38b}$$

Wybierzmy teraz dowolne $k \in \{\,0, \ldots, N\,/\,2-1\}$ i zgodnie z własnością symetrii (2.36) dyskretnego przekształcenia Fouriera zbadajmy część rzeczywistą jedynie dla k–tej współrzędnej omawianej transformaty, otrzymujemy:

$$\begin{aligned} Re\,\{\,\overline{F}_N(\,\mathbf{x}\,)_k\,\} &\stackrel{(2.35)}{=} \sum_{n=0}^{N-1} x_n\, cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right] = \\ = \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right] + sin\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\right) &+ \\ + \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right] - sin\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\right) &= \\ \stackrel{(2.34)}{=} \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,k\,\pi}{N}\,\right] + sin\left[\,\frac{2\,n\,k\,\pi}{N}\,\right]\right) &+ \\ + \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\,\frac{2\,n\,(\,N-k\,)\,\pi}{N}\,\right] + sin\left[\,\frac{2\,n\,(\,N-k\,)\,\pi}{N}\,\right]\right) &= \end{aligned} \tag{2.38c}$$

$$\overset{(2.29)}{=} \frac{1}{2}\,\overline{H}_N(\,\mathbf{x}\,)_k \,+\, \frac{1}{2}\,\overline{H}_N(\,\mathbf{x}\,)_{N-k} \,=$$

$$\overset{(2.32)}{\underset{(2.37)}{=}} \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right) + \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right) +$$

$$+ \quad \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right) - \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right) =$$

$$= \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k\, cos\left[\frac{k\,\pi}{2\,N}\right] + \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k}\, sin\left[\frac{k\,\pi}{2\,N}\right] \tag{2.38c}$$

Podobnie, weźmy teraz pod uwagę dowolne $k \in \{0,\ldots,N/2-1\}$ i zgodnie z własnością symetrii (2.36) dyskretnego przekształcenia Fouriera zbadajmy część urojoną jedynie dla k–tej współrzędnej omawianej transformaty, dostajemy:

$$Im\,\left\{\,\overline{F}_N(\,\mathbf{x}\,)_k\,\right\} \overset{(2.35)}{=} \sum_{n=0}^{N-1} x_n\, sin\left[\frac{2\,n\,k\,\pi}{N}\right] =$$

$$= -\frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] + sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) +$$

$$+ \quad \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] - sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) =$$

$$\overset{(2.34)}{=} -\frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,k\,\pi}{N}\right] + sin\left[\frac{2\,n\,k\,\pi}{N}\right]\right) +$$

$$+ \frac{1}{2}\sum_{n=0}^{N-1} x_n \left(cos\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right] + sin\left[\frac{2\,n\,(\,N-k\,)\,\pi}{N}\right]\right) = \tag{2.38d}$$

$$\overset{(2.29)}{=} -\frac{1}{2}\,\overline{H}_N(\,\mathbf{x}\,)_k \,+\, \frac{1}{2}\,\overline{H}_N(\,\mathbf{x}\,)_{N-k} \,=$$

$$\overset{(2.32)}{\underset{(2.37)}{=}} -\frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right) - \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right) +$$

$$+ \quad \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k \left(cos\left[\frac{k\,\pi}{2\,N}\right] + sin\left[\frac{k\,\pi}{2\,N}\right]\right) - \frac{1}{2}\,\overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k} \left(cos\left[\frac{k\,\pi}{2\,N}\right] - sin\left[\frac{k\,\pi}{2\,N}\right]\right) =$$

$$= \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_k\, sin\left[\frac{k\,\pi}{2\,N}\right] - \overline{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k}\, cos\left[\frac{k\,\pi}{2\,N}\right]$$

Przepisując (2.36a) – (2.38d) w krótszej formie dla $k \in \{\,0,\ldots,N\,/\,2-1\,\}$ mamy:

$$\widetilde{F}_N(\,\mathbf{x}\,)_0 \overset{(2.38a)}{=} \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_0$$

$$\widetilde{F}_N(\,\mathbf{x}\,)_{N/2} \overset{(2.38b)}{=} \sqrt{2}\,\cdot\,\widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_{N/2} \tag{2.39}$$

$$Re\,\left\{\,\widetilde{F}_N(\,\mathbf{x}\,)_k\,\right\} \overset{(2.38c)}{=} \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_k\, cos\left[\frac{k\,\pi}{2\,N}\right] + \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k}\, sin\left[\frac{k\,\pi}{2\,N}\right] \overset{(2.36a)}{=} Re\,\left\{\,\widetilde{F}_N(\,\mathbf{x}\,)_{N-k}\,\right\}$$

$$Im\,\left\{\,\widetilde{F}_N(\,\mathbf{x}\,)_k\,\right\} \overset{(2.38d)}{=} \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_k\, sin\left[\frac{k\,\pi}{2\,N}\right] - \widetilde{C}_N(\,\widetilde{\mathbf{x}}\,)_{N-k}\, cos\left[\frac{k\,\pi}{2\,N}\right] \overset{(2.36a)}{=} -Im\,\left\{\,\widetilde{F}_N(\,\mathbf{x}\,)_{N-k}\,\right\}$$

$$\mathbf{x},\,\widetilde{\mathbf{x}} \in \mathbb{R}^N,\ \ \forall\, n \in \{\,0,\ldots,N/\,2-1\}\ \ \widetilde{x}_{2n} \,=\, x_n\,,\ \ \widetilde{x}_{2n+1} \,=\, x_{N-n-1}$$

Postaci zależności (2.39) oraz rekursywnego wzoru rozkładu (2.19) szybkiego algorytmu zunifikowanego obliczania N–punktowego jednowymiarowego dyskretnego przekształcenia kosinusowego skłaniają do wniosku, że wyznaczanie N–punktowego przekształcenia Fouriera może być wykonane za pomocą ostatniej z wymienionych metod, jedynie poprzez zastosowanie odpowiedniej permutacji ciągu wejściowego $\mathbf{x}$ algorytmu (2.19) i modyfikację operacji bazowych na ostatnim etapie działania procedury `JFCT1D`. Uzyskaną w ten sposób procedurę wyznaczania współczynników skalowanego przekształcenia Fouriera nazywać będziemy dalej metodą `JFFT1D`. Poniższy schemat przedstawia pseudokod szybkiego algorytmu obliczania N–punktowego jednowymiarowego skalowanego dyskretnego przekształcenia Fouriera metodą `JFFT1D`, wynikający z zależności (2.39) oraz reguły zamiany współczynników dla ostatniej iteracji rozważanej metody opisanej w dalszej części niniejszego podpunktu.

Alg. 2.4: Algorytm `JFFT1D` dla N–punktowej transformaty Fouriera

$N \in \mathbb{N}$: wymiar transformaty będący naturalną potęgą liczby 2 ;
$\mathbf{x} \in \mathbb{R}^N$: wektor wejściowy ;
$\mathbf{y} \in \mathbb{R}^N$: wektor wyjściowy ;

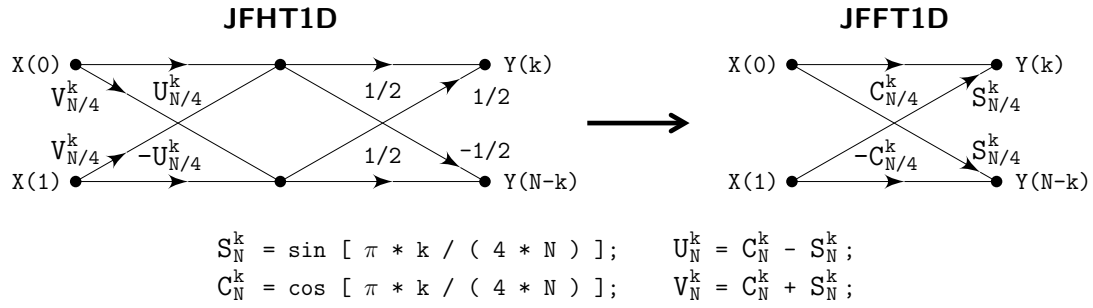
Utwórz wektor pomocniczy $\widetilde{\mathbf{x}} \in \mathbb{R}^N$ w następujący sposób:
$\forall\, n \in \{\, 0, \ldots, N/\,2\, -1\}\;\; \widetilde{x}_{2n} = x_n\,,\;\; \widetilde{x}_{2n+1} = x_{N-n-1}$;

Dla pomocniczego wektora wejściowego $\widetilde{\mathbf{x}}$ zastosuj szybki dwuetapowy algorytm 4.1 modyfikując jego działanie w ostatnim kroku etapu drugiego (dla $M = N\,/\,2$ oraz $k = N\,/\,(\,2\,M\,)\, -\, 1\,$) przez zastąpienie wzorów (4.1.2) następującymi zależnościami:

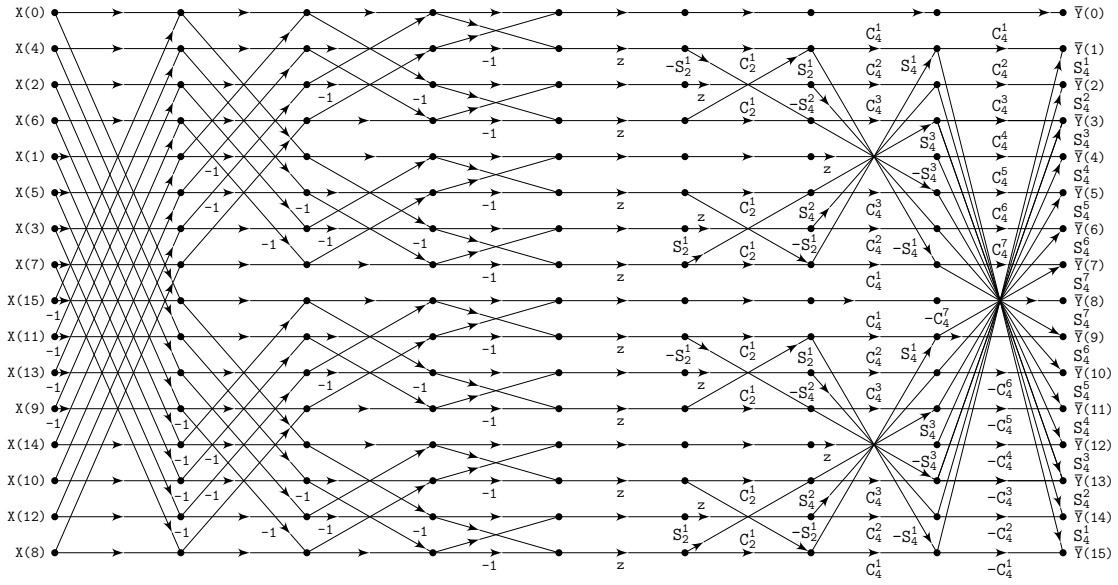
$$\begin{aligned}
&\mathbf{y}_{2M}^{(k)}(0) = \mathbf{y}_{M}^{(2k)}(0)\,,\;\; \mathbf{y}_{2M}^{(k)}(M) = \mathbf{y}_{M}^{(2k+1)}(0)\,,\\
&\mathbf{y}_{2M}^{(k)}(n) = C_{M/\,2}^{\,n} \cdot \mathbf{y}_{M}^{(2k)}(n) \,+\, S_{M/\,2}^{\,n} \cdot \mathbf{y}_{M}^{(2k+1)}(M-n)\,,\\
&\mathbf{y}_{2M}^{(k)}(M-n) = S_{M/\,2}^{\,n} \cdot \mathbf{y}_{M}^{(2k)}(n) \,-\, C_{M/\,2}^{\,n} \cdot \mathbf{y}_{M}^{(2k+1)}(M-n)\,,\\
&n = 1, 2, \ldots, M\, -\, 1\,;\quad S_N^k = sin\,(\,\pi\, k\,/\,(\,4\,N\,)\,)\,,\;\; C_N^k = cos\,(\,\pi\, k\,/\,(\,4\,N\,)\,)\,;
\end{aligned} \tag{2.4.1}$$

Oznacz powstały w ten sposób wektor wynikowy przez $\mathbf{y}$. Wówczas otrzymany wektor $\mathbf{y}$ jest skalowanym dyskretnym przekształceniem Fouriera `DFT1D` wektora wejściowego $\mathbf{x}$;

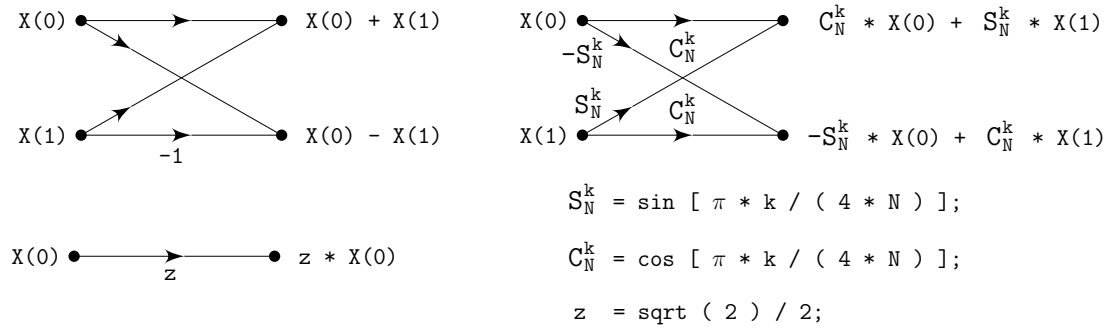
Rysunek 2.12 obrazuje regułę zamiany współczynników dla ostatniej iteracji algorytmu `JFFT1D` względem podanego wcześniej szybkiego algorytmu dwuetapowego wyznaczania dyskretnego przekształcenia Hartleya, zgodną z wnioskiem wynikającym z pary środkowych przejść w równościach (2.38a) i (2.38b). Rysunek 2.13a przedstawia graf przepływu danych dla algorytmu `JFFT1D` wynikający z pseudokodu uwidocznionego na schemacie 2.4 dla przykładowego 16–punktowego przekształcenia Fouriera. Rysunek 2.13b przedstawia operacje bazowe dla grafu przepływu danych 2.13a algorytmu `JFFT1D`.



Rys. 2.12: Reguła zamiany współczynników ostatniej iteracji algorytmu JFFT1D



Rys. 2.13a: Szybki algorytm JFFT1D obliczania 16–punktowej skalowanej transformaty Fouriera



Rys. 2.13b: Operacje bazowe algorytmu JFFT1D dla przekształcenia Fouriera

Warto wyjaśnić iż pomimo, że reguła zamiany współczynników przestawiona na rysunku 2.12 podana jest dla przejścia pomiędzy szybkimi procedurami obliczeniowymi przekształceń Hartleya i Fouriera (co spowodowane jest znacznie prostszą for-

mą zależności pomiędzy tymi przekształceniami niż odpowiednia zależność między parą transformat kosinusowej oraz Fouriera) to wynikowe współczynniki z rysunku 2.12 stanowią już, podobnie jak miało to miejsce dla reguł zamiany współczynników przedstawionych w poprzednich paragrafach, gotowe wartości dwupunktowych obrotów ortogonalnych jakimi należy zastąpić odpowiadające im wartości analogicznych współczynników w ostatniej iteracji szybkiego algorytmu bazowego `JFCT1D`, aby uzyskać rozważaną tu procedurę obliczeniową `JFFT1D`.

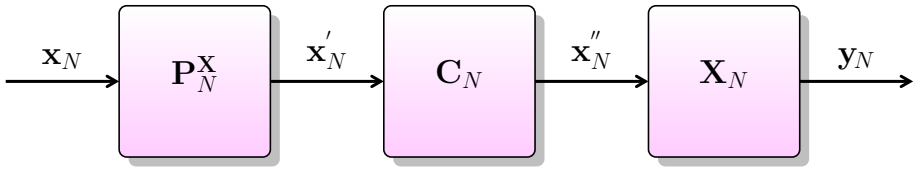
Zgodnie z powyższą uwagą, wobec związków (2.39) pomiędzy skalowanymi przekształceniami Fouriera i kosinusowym, a także z postaci reguły 2.12 zamiany współczynników w ostatnim etapie algorytmu `JFFT1D` można wnioskować o fakcie identycznych złożoności obliczeniowych i pamięciowych algorytmów `JFCT1D` i `JFFT1D`. A zatem algorytm `JFFT1D` jest procedurą szybką o złożoności obliczeniowej równej rzędem $\mathcal{O}\,(N log_2 N)$ i złożoności pamięciowej o rzędzie $\mathcal{O}\,(N)$. Podajmy na koniec dokładne liczby prostych operacji arytmetycznych i mnożeń dla rozważanego w niniejszym paragrafie algorytmu 2.4, wynoszą one:

$$\widetilde{\mathcal{F}}_N^{+} \;=\; 2\,N(\,log_2 N - 2\,) + N + 3\;\;, \quad \widetilde{\mathcal{F}}_N^{*} \;=\; 2\,N(\,log_2 N - 1\,) + 2 \tag{2.40}$$

Szybki dwuetapowy algorytm zunifikowany wyznacznia współczynników dyskretnego przekształcenia Fouriera jest już ostatnią z prezentowanych w niniejszym podrozdziale metod szybkich dla przykładowych przekształceń jednowymiarowych. Następny podpunkt zawiera już krótkie podsumowanie bieżącego podrozdziału i najważniejsze wnioski płynące z jego treści.

2.3.1.5. Uwagi końcowe

W niniejszym podrozdziale przedstawiono szybkie algorytmy zunifikowane wyznaczania wybranych jednowymiarowych przekształceń dyskretnych i poddano analizie ich charakterystyki efektywnościowo pamięciowe. Z rozważań tych wynika, w szczególności z pseudokodu 2.1 - 2.4 i grafów przepływu danych z rysunków 2.6a, 2.9, 2.11a i 2.13a, że rozpatrywane algorytmy dla przekształceń przykładowych posiadają wspólny, ogólny schemat obliczeniowy, którego graficzną interpretację przedstawia poniższy rysunek:



Rys. 2.14: Ogólny schemat obliczeniowy szybkich algorytmów zunifikowanych dla jednowymiarowych przekształceń dyskretnych

A zatem każdy z jednowymiarowych szybkich algorytmów zunifikowanych dla wybranego przekształcenia dyskretnego różni się od algorytmów dla pozostałych przekształceń jedynie sposobem przemieszania $\mathbf{P}_N^{\mathbf{X}}$ elementów wektorów wejściowych

i macierzą $\mathbf{X}_N$ ostatniej iteracji obliczeniowej realizującą przejście z przekształcenia pośredniego do wybranej transformaty docelowej, przy czym struktura motylkowa tejże macierzy jest identyczna dla wszystkich analizowanych przekształceń dyskretnych. Macierz $\mathbf{C}_N$ bazowego przekształcenia kosinusowego, stanowiąca główny trzon rozważanego tu ogólnego schematu obliczeniowego jest wspólna dla każdego z przedstawionych algorytmów. Ponadto wszystkie macierze są rzadkimi macierzami motylkowymi, co w sumie sprawia, iż cały schemat jest schematem szybkim o następujcej charakterystyce złożoności obliczeniowej, wynikającej z zależności (2.20), (2.25), (2.33) i (2.40):

$$\widetilde{\mathcal{J}}_N^{+} \;=\; 2\,N(\,log_2\,N-2\,)+N+3\;\;\mathbf{,}\qquad \widetilde{\mathcal{J}}_N^{*} \;=\; 2\,N(\,log_2\,N-1\,)+2 \tag{2.41}$$

gdzie $\widetilde{\mathcal{J}}_N^{+}$ oraz $\widetilde{\mathcal{J}}_N^{*}$ są odpowiednio liczbą dodawań/odejmowań rzeczywistych oraz rzeczywistych mnożeń potrzebnych do wyznaczenia współczynników wybranej jednowymiarowej transformaty docelowej. Dodatkowo, jak wynika chociażby z zaprezentowanych w niniejszym podrozdziale grafów przepływu danych, rozważany tu ogólny schemat obliczeniowy charakteryzuje się nie tylko liniowo-logarytmiczną złożonością obliczeniową rzędu $\mathcal{O}(Nlog_2N)$, ale także liniową złożonością pamięciową rzędu $\mathcal{O}(N)$ względem rozmiaru wybranego przekształcenia dyskretnego.

Podsumowując, w niniejszym podrozdziale przedstawiono algorytmy kalkulacji wybranych, znanych transformat ortogonalnych prowadzące do sformułowania jednolitego, ogólnego, szybkiego i efektywnego pamięciowo schematu ich obliczania. Na podstawie tych przesłanek można wnioskować, że schemat ten ma efektywny i uniwersalny zarazem charakter i jako taki mógłby stanowić bazę dla konstrukcji algorytmów obliczeniowych wyznaczania innych, nowych przekształceń dyskretnych, dedykowanych do wybranych zadań filtracji liniowej sygnałów, takich jak np. zadanie kompresji obrazów. Ze względu na korzystną postać grafową przedstawionych algorytmów zunifikowanych proces dostosowywania rozważanego schematu obliczeniowego do wybranego problemu filtracji mógłby być zrealizowany np. za pomocą neuronowych technik optymalizacyjnych. Zagadnienia przedstawione w dalszej części niniejszego opracowania dotyczyć będą właśnie konstrukcji architektury sieci neuronowej do kompresji obrazów opartej o przedstawiony tu uniwersalny, szybki schemat obliczeniowy wyznaczania dyskretnych przekształceń liniowych. Aby tego dokonać należy jednak rozważyć jeszcze przypadek szybkich algorytmów zunifikowanych dla przekształceń dwuwymiarowych, co jest treścią następnego podrozdziału.

2.3.2. Szybkie algorytmy zunifikowane przekształceń dwuwymiarowych

W tej częsci opracowania rozpatrzone zostaną szybkie algorytmy zunifikowane dla dwuwymiarowych przekształceń dyskretnych przedstawione, w sposób pośredni, między innymi także przez autora w publikacjach [27, 28, 32, 65, 67, 85]. Analogicznie jak w przypadku transformat jednowymiarowych algorytmy te cechują się wspólną, uniwersalną strukturą procedur obliczeniowych. Dodatkowo struktura ta jest identyczna nie tylko w obrębie rozpatrywanych przekształceń dwuwymiarowych,

ale także jest w pełni zgodna ze strukturą prezentowanych wcześniej metod obliczeniowych dla przekształceń jednowymiarowych, stanowiącą bazę konstrukcyjną dla rozpatrywanych tu algorytmów zunifikowanych. Obydwie te cechy są bardzo istotne i sprawiają, że przedstawione w niniejszym podrozdziale algorytmy definiują wspólny, jednolity i efektywny schemat obliczeniowy dla wyznaczania zarówno jedno jak i dwuwymiarowych przekształceń dyskretnych, będący już bezpośrednią podstawą konstrukcji architektury sieci neuronowej do kompresji obrazów. Podobnie jak w przypadku transformat jednowymiarowych tak i tu przekształceniem bazowym jest dyskretna transformata kosinusowa, której szybki algorytm zunifikowany jest przedmiotem rozważań następnego podpunktu rozdziału.

2.3.2.1. Szybkie zunifikowane dwuwymiarowe dyskretne przekształcenie kosinusowe

W niniejszym podpunkcie zaprezentowany zostanie szybki zunifikowany algorytm wyznaczania dwuwymiarowej dyskretnej transformaty kosinusowej, który jak wspomniano we wprowadzeniu, stanowi podstawę syntezy rozważanych dalej metod szybkich obliczania wybranych dwuwymiarowych przekształceń dyskretnych, takich jak przykładowe transformaty sinusowa, Hartleya czy Fouriera. W celu uniknięcia nieścisłości podajmy na wstępie formalną definicję rozważanego przekształcenia dwuwymiarowego.

Definicja 2.6.

Dla dowolnej liczby $N = 2^m$, $m \in \mathbb{N}$ funkcję $\overline{C}_{N\times N}\colon \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ określoną w następujący sposób:

$$\forall\, \mathbf{X} \in \mathbb{R}^{N\times N} \quad \forall\, k,\, l \in \{0, \ldots, N-1\}$$
$$\overline{C}_{N\times N}(\mathbf{X})_{k,l} \triangleq \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x_{m,n}\; cos\left[\frac{(2m+1)\,k\,\pi}{2M}\right] cos\left[\frac{(2n+1)\,l\,\pi}{2N}\right] \tag{2.42}$$

zwiemy $N \times N$*–punktowym skalowanym dwuwymiarowym dyskretnym przekształceniem kosinusowym.*

W celu zapisu rozważanego algorytmu autor skorzysta w tym przypadku z notacji macierzowej. Należy zatem podać definicje wszystkich składników macierzowych dla konstruowanej metody. Zgodnie z obowiązującą powszechnie konwencją iloczyny lewostronny i prawostronny macierzy oznaczane będą symbolami:

$$\prod_{n=1}^{N} \mathbf{X}_n = \mathbf{X}_N \cdot \mathbf{X}_{N-1} \cdot \ldots \cdot \mathbf{X}_2 \cdot \mathbf{X}_1\,; \qquad \mathbf{X}_n \prod_{n=1}^{N} = \mathbf{X}_1 \cdot \mathbf{X}_2 \cdot \ldots \cdot \mathbf{X}_{N-1} \cdot \mathbf{X}_N \tag{2.43}$$

Niech dla $N \in \mathbb{N}$ będącego dowolną naturalną potęgą dwójki macierze $\mathbf{C}_n \in \mathbb{R}^{N\times N}$, $n = 1, \ldots, 2\,log_2 N$ będą macierzami motylkowymi kolejnych etapów zdefiniowanego zależnościami (2.19) szybkiego algorytmu obliczania dyskretnego jednowymiarowego przekształcenia kosinusowego `JFCT1D`. Wtedy wspomniany szybki algorytm `JFCT1D`, reprezentowany macierzą $\widetilde{\mathbf{C}}_N$, może być przedstawiony w następujący sposób:

$$\widetilde{\mathbf{C}}_N = \prod_{n=1}^{2log_2N} \mathbf{C}_n \underset{(2.19)}{\overset{(2.16)}{=}} \overline{\mathbf{C}}_N \mathbf{B} \tag{2.44}$$

gdzie $\overline{\mathbf{C}}_N$ jest $N \times N$–elementowym jądrem jednowymiarowej skalowanej transformaty kosinusowej, danej zależnością (2.16), zaś $\mathbf{B} \in \mathbb{R}^N$ jest macierzą odwróconego porządku bitowego (*bit-reversal order*). Dla dowolnego $N = 2^m$ zdefiniujmy teraz pomocnicze macierze rzeczywiste w podany niżej sposób:

$$\mathbf{R}_N \triangleq \begin{pmatrix} 1 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \mathbf{G}_{N-1\times N-1} \end{pmatrix};$$

$$\overline{\mathbf{I}}_N \triangleq \begin{pmatrix} 0 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \mathbf{I}_{N-1\times N-1} \end{pmatrix}; \quad \widetilde{\mathbf{I}}_N \triangleq \begin{pmatrix} 1 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \mathbf{0}_{N-1\times N-1} \end{pmatrix}; \tag{2.45}$$

$$\overline{\mathbf{Q}}_N \triangleq diag\,(\,\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_N\,); \quad \overline{q}_n \triangleq \frac{1}{2}\cdot(-1)^n + \frac{1}{2}, \qquad n = 1, \ldots, N;$$

$$\widetilde{\mathbf{Q}}_N \triangleq diag\,(\,\widetilde{q}_1, \widetilde{q}_2, \ldots, \widetilde{q}_N\,); \quad \widetilde{q}_n \triangleq \frac{1}{2}\cdot(-1)^{n+1} + \frac{1}{2}, \quad n = 1, \ldots, N;$$

$$\mathbf{Z}_{N\times N} \triangleq \overline{\mathbf{Q}}_N \otimes \widetilde{\mathbf{I}}_N + \widetilde{\mathbf{Q}}_N \otimes (\,\mathbf{R}_N \cdot \widetilde{\mathbf{I}}_N\,);$$

$$\mathbb{R}^{N\times N} \ni \mathbf{P}_k \triangleq \overline{\mathbf{Q}}_{N/2^k} \otimes \widetilde{\mathbf{I}}_{2^k} + \widetilde{\mathbf{Q}}_{N/2^k} \otimes \mathbf{U}_{2^k}, \quad k = 1, \ldots, log_2N;$$

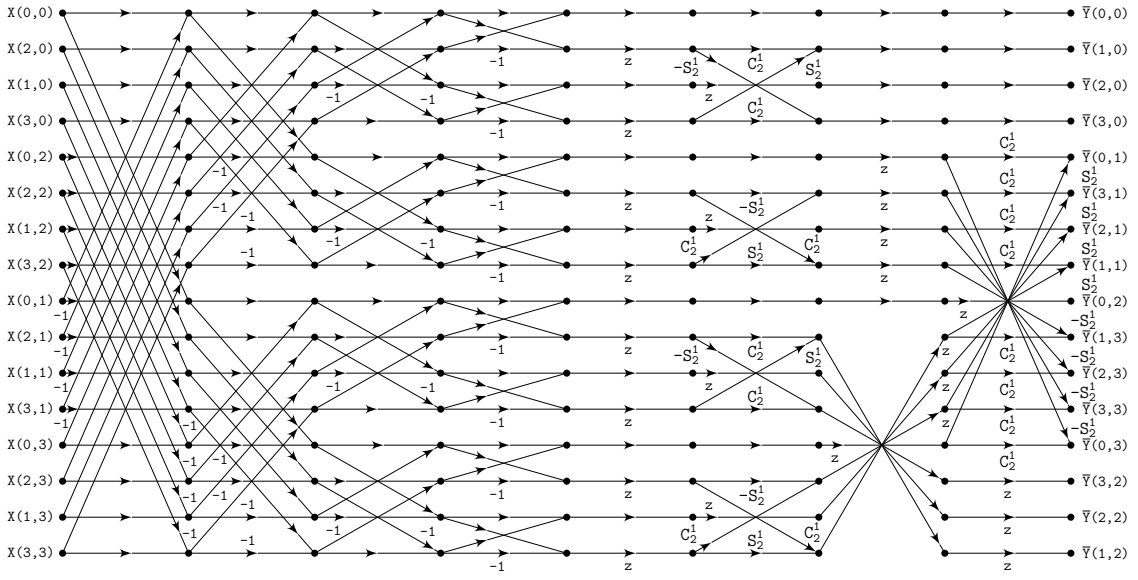
gdzie $\mathbf{G}$ jest macierzą rzeczywistą o elementach zerowych wszędzie poza jej główną przeciwdaigonalą złożoną z samych jedynek, zaś $\mathbf{U}$ jest macierzą powstałą z macierzy jednostkowej o odpowiednich wymiarach, w której zamieniono wiersze pierwszy z ostatnim. Oznaczmy symbolem $\widetilde{\mathbf{C}}_{N\times N}$ i zdefiniujmy poniżej rzeczywistą $N^2 \times N^2$ –elementową macierz kwadratową reprezentującą proponowany w niniejszym opracowaniu szybki algorytm wyznaczania dwuwymiarowego $N\times N$–punktowego dyskretnego przekształcenia kosinusowego:

$$\widetilde{\mathbf{C}}_{N\times N} \triangleq \prod_{n=1}^{log_2N} \left\{ \mathbf{C}_{log_2N+n} \otimes \overline{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{C}_{log_2N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} * \tag{2.46}$$

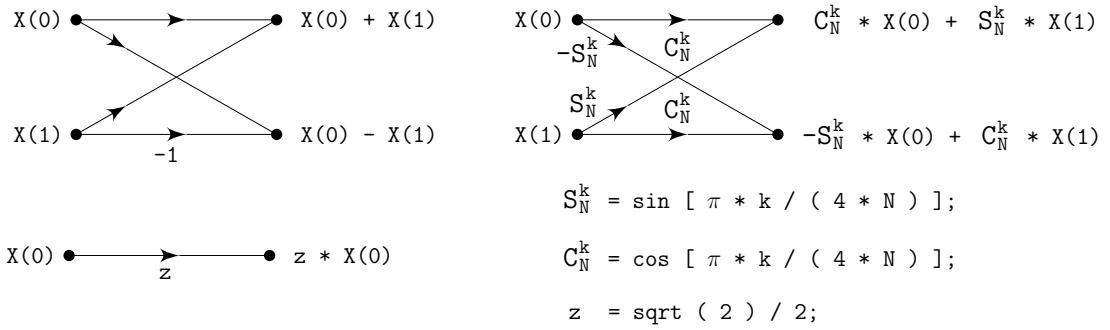
$$* \left\{ \left[\, \overline{\mathbf{Q}} \otimes \mathbf{C}_{2log_2N} + \widetilde{\mathbf{Q}} \otimes (\,\mathbf{R}\,\mathbf{C}_{2log_2N}\,) \right] \cdot \prod_{n=1}^{2log_2N-1} (\,\mathbf{I} \otimes \mathbf{C}_n\,) \right\} * \left\{ \prod_{n=1}^{log_2N} (\,\mathbf{C}_n \otimes \mathbf{I}\,) \right\}$$

gdzie wszystkie ze zdefiniowanych wzorami (2.46) macierzy opatrzone są domyślnym indeksem N lub, jak w przypadku macierzy $\mathbf{Z}$, odpowiednio indeksem $N \times N$. Bezpośrednio ze sposobu konstrukcji macierzy $\widetilde{\mathbf{C}}_{N\times N}$ wynika, że struktura jej poszczególnych macierzy składowych, odpowiadających kolejnym iteracjom zdeterminowanego w ten sposób algorytmu wyznaczania pewnego $N \times N$–punktowego dwuwymiarowego przekształcenia dyskretnego, jest w pełni zgodna ze strukturą kolejnych iteracji grafów przepływu danych przedstawionych wcześniej zunifikowanych algorytmów szybkich obliczania dyskretnych przekształceń jednowymiarowych dla przypadku transformat N^2–punktowych. Wyjaśnijmy teraz krótko rolę i postaci poszczególnych składników macierzy $\widetilde{\mathbf{C}}_{N\times N}$ becnych we wzorze (2.46). Oznaczmy dla

wygody macierze w nawiasach klamrowych przedstawione w tym wzorze odpowiednio od strony prawej do lewej symbolami $\widetilde{\mathbf{C}}_{(1)}$, $\widetilde{\mathbf{C}}_{(2)}$ oraz $\widetilde{\mathbf{C}}_{(3)}$. I tak, idąc od prawej strony wzoru (2.46), macierz $\widetilde{\mathbf{C}}_{(1)}$ stanowi iloczyn macierzy, którego każdy ze składników złożony jest z N „rozciągniętych" N–krotnie i umieszczonych na przemian obok siebie (co jeden wiersz każda) kopii macierzy składowych $\mathbf{C}_n$ (o indeksach od 1 do $log_2 N$) podanego zależnością (2.44) jądra $\widetilde{\mathbf{C}}_N$ N - punktowego jednowymiarowego szybkiego przekształcenia kosinusowego `JFCT1D`. Macierz blokowa $\widetilde{\mathbf{C}}_{(2)}$, zawarta w środkowym nawiasie klamrowym wzoru (2.46), jest macierzą, której każdy z $N \times N$–elementowych bloków tworzy wierną kopię grafu przepływu danych szybkiego algorytmu zunifikowanego (2.19) wyznaczania N–punktowego jednowymiarowego przekształcenia kosinusowego, przy czym nieparzyste bloki tej macierzy są macierzami szybkich przekształceń jednowymiarowych $\widetilde{\mathbf{C}}_N$ z odwróconą kolejnością poszczególnych wyjść (poza ich wyjściami o indeksach zero oraz $N/2$, które pozostają na swoich miejscach). Wreszcie ostatnia z macierzy obecnych we wzorze (2.46), tzn. macierz $\widetilde{\mathbf{C}}_{(3)}$ znajdująca się w jego lewym skrajnym nawiasie klamrowym, jest iloczynem macierzy, którego każdy ze składników stanowi N kopii odpowiednio zmapowanych na strukturę jednowymiarową macierzy składowych $\mathbf{C}_n$ (o indeksach od $log_2 N + 1$ do $2log_2 N$) jądra $\widetilde{\mathbf{C}}_N$ N–punktowego jednowymiarowego szybkiego przekształcenia kosinusowego `JFCT1D`. Wspomniane mapowanie zapewniane jest w tym przypadku wspólnie przez $N \times N$–elementowe macierze permutacji $\mathbf{P}_k$, $k = 1, \ldots, log_2 N$ oraz $N^2 \times N^2$–elementową macierz $\mathbf{Z}$, która dokonuje ostatecznego rozmieszczenia poszczególnych elementów macierzy składowych algorytmu szybkiego `JFCT1D` wyznaczania jednowymiarowego przekształcenia kosinusowego w macierzach rozważanego tu algorytmu dwuwymiarowego, w taki sposób, aby zagwarantować zgodność strukturalną grafów przepływu danych obydwu wspomnianych algorytmów.



Rys. 2.15a: Graf przepływu 4×4–punktowej transformaty dyskretnej, wynikający ze wzoru (2.46)



Rys. 2.15b: Operacje podstawowe grafu przepływu danych algorytmu (2.46)

Warto dodać, że macierz $\mathbf{Z}$ operuje w sensie permutacji na wszystkich wierszach poszczególnych macierzy wynikowych algorytmu dwuwymiarowego, poza wierszami o indeksach $N \cdot k$, $k = 1, \ldots, N-1$, które stanowią wierne, N–krotnie „rozciągnięte" kopie odpowiednich macierzy algorytmu jednowymiarowego (stąd operacja dodawania w skrajnym lewym nawiasie klamrowym wzoru (2.46)). Podsumowując, bezpośrednio z konstrukcji podanej wzorem (2.46) macierzy $\widetilde{\mathbf{C}}_{N\times N}$ wynika, że reprezentuje ona algorytm wyznaczania pewnego przekształcenia dyskretnego o grafie przepływu danych w pełni zgodnym z przedstawionymi w poprzednich podpunktach niniejszego rozdziału grafami przepływowymi szybkich algorytmów zunifikowanych wyznaczania wybranych dyskretnych przekształceń jednowymiarowych. Sytuację taką ilustruje przykładowa postać macierzy $\widetilde{\mathbf{C}}_{N\times N}$ dla $N = 4$, która przedstawiona jest w formie grafu przepływowego i jego operacji bazowych na rysunkach 2.15a oraz 2.15b. Pozostaje w takim razie pokazanie, iż dla N będącego dowolną potęgą liczby dwa macierz $\widetilde{\mathbf{C}}_{N\times N}$ reprezentuje w istocie zunifikowany algorytm szybki wyznaczania $N \times N$–punktowego dwuwymiarowego dyskretnego przekształcenia kosinusowego. Przeprowadźmy odpowiednie rozumowanie. Na wstępie warto podać bardzo użyteczny fakt dotyczący własności produktu Kroneckera dla macierzy rzeczywistych:[1]

$$(\,\mathbf{A}\otimes\mathbf{B}\,)\cdot(\,\mathbf{C}\otimes\mathbf{D}\,) = (\,\mathbf{A}\,\mathbf{C}\,)\otimes(\,\mathbf{B}\,\mathbf{D}\,) \tag{2.47}$$

Warto uprościć składnik $\widetilde{\mathbf{C}}_{(3)}$ we wzorze (2.46), dla N będącego potęgą dwójki oraz dowolnego $m \in \{\,1, \ldots, log_2 N - 1\,\}$ otrzymujemy:

$$\begin{aligned}
&\left\{ \mathbf{C}_{log_2N+m+1}\otimes\bar{\mathbf{I}} + \mathbf{Z}\left[\left(\prod_{k=1}^{m+1}\mathbf{P}_k\mathbf{C}_{log_2N+m+1}\mathbf{P}_k^T\prod_{k=1}^{m}\right)\otimes\widetilde{\mathbf{I}}\right]\mathbf{Z}^T\right\} * \\
&* \left\{\left(\prod_{n=log_2N+1}^{log_2N+m}\mathbf{C}_n\right)\otimes\bar{\mathbf{I}} + \mathbf{Z}\left[\left(\prod_{n=1}^{m}\mathbf{P}_n\cdot\prod_{n=log_2N+1}^{log_2N+m}\mathbf{C}_n\right)\otimes\widetilde{\mathbf{I}}\right]\mathbf{Z}^T\right\} = \\
&= \left(\mathbf{C}_{log_2N+m+1}\otimes\bar{\mathbf{I}}\right)\cdot\left[\left(\prod_{n=log_2N+1}^{log_2N+m}\mathbf{C}_n\right)\otimes\bar{\mathbf{I}}\right] + \\
&+ \mathbf{Z}\left[\left(\prod_{k=1}^{m+1}\mathbf{P}_k\mathbf{C}_{log_2N+m+1}\mathbf{P}_k^T\prod_{k=1}^{m}\right)\otimes\widetilde{\mathbf{I}}\right]\cdot\left[\left(\prod_{n=1}^{m}\mathbf{P}_n\cdot\prod_{n=log_2N+1}^{log_2N+m}\mathbf{C}_n\right)\otimes\widetilde{\mathbf{I}}\right]\mathbf{Z}^T
\end{aligned} \tag{2.48}$$

[1] patrz twierdzenie T2.4 na str. 773 w pracy [91]

Postawmy teraz hipotezę: dla dowolnego $m \in \{1, \ldots, log_2 N\}$ zachodzi równość:

$$\begin{aligned}
&\prod_{n=1}^{m} \left\{ \mathbf{C}_{log_2 N+n} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{C}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} = \\
&= \left(\prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{m} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T
\end{aligned} \tag{2.49}$$

Hipoteza (2.49) zostanie poniżej wykazana za pomocą zasady indukcji matematycznej. Sprawdźmy zatem czy hipoteza ta jest spełniona dla $m = 1$, przy tak dobranym m lewa strona zależności (2.49) jest równa:

$$\begin{aligned}
&\mathbf{C}_{log_2 N+1} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\mathbf{P}_1 \mathbf{C}_{log_2 N+1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T = \\
&= \left(\prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{m} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T
\end{aligned} \tag{2.50}$$

A zatem na mocy powyższej równości dla parametru $m = 1$ rozpatrywana hipoteza (2.49) jest spełniona. Wybierzmy teraz dowolne $m \in \{1, \ldots, log_2 N - 1\}$ i załóżmy, że dla tak dobranej wartości parametru m zachodzi zależność (2.49), wówczas otrzymujemy następujący ciąg równości:

$$\begin{aligned}
&\prod_{n=1}^{m+1} \left\{ \mathbf{C}_{log_2 N+n} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{C}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} = \\
&= \left\{ \mathbf{C}_{log_2 N+m+1} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{m+1} \mathbf{P}_k \mathbf{C}_{log_2 N+m+1} \mathbf{P}_k^T \prod_{k=1}^{m} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} * \\
&* \prod_{n=1}^{m} \left\{ \mathbf{C}_{log_2 N+n} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{C}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} = \\
&\overset{(2.13b)}{=} \left\{ \mathbf{C}_{log_2 N+m+1} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{m+1} \mathbf{P}_k \mathbf{C}_{log_2 N+m+1} \mathbf{P}_k^T \prod_{k=1}^{m} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} * \\
&* \left\{ \left(\prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{m} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} = \\
&\overset{(2.48)}{=} \left(\mathbf{C}_{log_2 N+m+1} \otimes \bar{\mathbf{I}} \right) \cdot \left[\left(\prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} \right] + \\
&+ \mathbf{Z} \left[\left(\prod_{k=1}^{m+1} \mathbf{P}_k \mathbf{C}_{log_2 N+m+1} \mathbf{P}_k^T \prod_{k=1}^{m} \right) \otimes \widetilde{\mathbf{I}} \right] \cdot \left[\left(\prod_{n=1}^{m} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{log_2 N+m} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T = \\
&\underset{(2.47)}{\overset{(2.45)}{=}} \left(\prod_{n=log_2 N+1}^{log_2 N+m+1} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{m+1} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{log_2 N+m+1} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T
\end{aligned} \tag{2.51}$$

Wobec dowolności wyboru m powyższy wniosek wraz z równością (2.50) potwierdza na mocy zasady indukcji matematycznej prawdziwość tezy (2.49). Stąd zaś otrzymujemy natychmiast następującą zależność:

$$\begin{aligned}\prod_{n=1}^{log_2 N}\left\{\mathbf{C}_{log_2 N+n}\otimes\overline{\mathbf{I}}+\mathbf{Z}\left[\left(\prod_{k=1}^{n}\mathbf{P}_k\mathbf{C}_{log_2 N+n}\mathbf{P}_k^T\prod_{k=1}^{n-1}\right)\otimes\widetilde{\mathbf{I}}\right]\mathbf{Z}^T\right\}=\\ =\left(\prod_{n=log_2 N+1}^{2log_2 N}\mathbf{C}_n\right)\otimes\overline{\mathbf{I}}+\mathbf{Z}\left[\left(\prod_{n=1}^{log_2 N}\mathbf{P}_n\cdot\prod_{n=log_2 N+1}^{2log_2 N}\mathbf{C}_n\right)\otimes\widetilde{\mathbf{I}}\right]\mathbf{Z}^T\end{aligned}\tag{2.52}$$

Po zakończeniu analizy postaci składnika $\widetilde{\mathbf{C}}_{(3)}$ we wzorze (2.46) zajmijmy się uproszeniem macierzy $\widetilde{\mathbf{C}}_{(2)}$, z zależności (2.47) natychmiast otrzymujemy:

$$\prod_{n=1}^{2log_2 N-1}(\,\mathbf{I}\otimes\mathbf{C}_n\,)=\mathbf{I}\otimes\prod_{n=1}^{2log_2 N-1}\mathbf{C}_n\,;\qquad\prod_{n=1}^{log_2 N}(\,\mathbf{C}_n\otimes\mathbf{I}\,)=\left(\prod_{n=1}^{log_2 N}\mathbf{C}_n\right)\otimes\mathbf{I}\,;\tag{2.53}$$

Korzystając z powyższych zależności otrzymujemy:

$$\begin{aligned}&\left[\,\overline{\mathbf{Q}}\otimes\mathbf{C}_{2log_2 N}+\widetilde{\mathbf{Q}}\otimes(\,\mathbf{R}\,\mathbf{C}_{2log_2 N}\,)\right]\cdot\prod_{n=1}^{2log_2 N-1}(\,\mathbf{I}\otimes\mathbf{C}_n\,)=\\ &\overset{(2.53)}{=}\left[\,\overline{\mathbf{Q}}\otimes\mathbf{C}_{2log_2 N}+\widetilde{\mathbf{Q}}\otimes(\,\mathbf{R}\,\mathbf{C}_{2log_2 N}\,)\right]\cdot\mathbf{I}\otimes\prod_{n=1}^{2log_2 N-1}\mathbf{C}_n=\\ &=\left(\,\overline{\mathbf{Q}}\otimes\mathbf{C}_{2log_2 N}\,\right)\cdot\left(\mathbf{I}\otimes\prod_{n=1}^{2log_2 N-1}\mathbf{C}_n\right)+\left[\,\widetilde{\mathbf{Q}}\otimes(\,\mathbf{R}\,\mathbf{C}_{2log_2 N}\,)\right]\cdot\left(\mathbf{I}\otimes\prod_{n=1}^{2log_2 N-1}\mathbf{C}_n\right)=\\ &\overset{(2.47)}{=}\overline{\mathbf{Q}}\otimes\prod_{n=1}^{2log_2 N}\mathbf{C}_n+\widetilde{\mathbf{Q}}\otimes\left(\mathbf{R}\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)=\left(\,\overline{\mathbf{Q}}\cdot\mathbf{I}\,\right)\otimes\left(\mathbf{I}\cdot\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)+\\ &+\left(\,\widetilde{\mathbf{Q}}\cdot\mathbf{I}\,\right)\otimes\left(\mathbf{R}\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)\overset{(2.47)}{=}\left(\,\overline{\mathbf{Q}}\otimes\mathbf{I}\,\right)\cdot\left(\mathbf{I}\otimes\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)+\\ &+\left(\,\widetilde{\mathbf{Q}}\otimes\mathbf{R}\,\right)\cdot\left(\mathbf{I}\otimes\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)=\left[\left(\,\overline{\mathbf{Q}}\otimes\mathbf{I}\,\right)+\left(\,\widetilde{\mathbf{Q}}\otimes\mathbf{R}\,\right)\right]\cdot\left(\mathbf{I}\otimes\prod_{n=1}^{2log_2 N}\mathbf{C}_n\right)\end{aligned}\tag{2.54}$$

Ponadto łatwo zweryfikować prawdziwość poniższych równości pomocniczych:

$$\mathbf{Z}^T\left[\left(\,\overline{\mathbf{Q}}\otimes\mathbf{I}\,\right)+\left(\,\widetilde{\mathbf{Q}}\otimes\mathbf{R}\,\right)\right]=\mathbf{I}\otimes\widetilde{\mathbf{I}}\tag{2.55a}$$

$$\left[\left(\prod_{n=log_2 N+1}^{2log_2 N}\mathbf{C}_n\right)\otimes\overline{\mathbf{I}}\right]\cdot\left[\left(\,\overline{\mathbf{Q}}\otimes\mathbf{I}\,\right)+\left(\,\widetilde{\mathbf{Q}}\otimes\mathbf{R}\,\right)\right]=\left(\prod_{n=log_2 N+1}^{2log_2 N}\mathbf{C}_n\right)\otimes\overline{\mathbf{I}}\tag{2.55b}$$

Przejdźmy do ostatecznego ustalenia związku macierzy $\widetilde{\mathbf{C}}_{N\times N}$ z dwuwymiarową transformatą kosinusową:

$$\widetilde{\mathbf{C}}_{N\times N} \overset{(2.46)}{=} \prod_{n=1}^{log_2 N} \left\{ \mathbf{C}_{log_2 N+n} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{C}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} *$$

$$* \left\{ \left[\overline{\mathbf{Q}} \otimes \mathbf{C}_{2log_2 N} + \widetilde{\mathbf{Q}} \otimes (\mathbf{R}\, \mathbf{C}_{2log_2 N}) \right] \cdot \prod_{n=1}^{2log_2 N-1} (\mathbf{I} \otimes \mathbf{C}_n) \right\} * \left\{ \prod_{n=1}^{log_2 N} (\mathbf{C}_n \otimes \mathbf{I}) \right\} =$$

$$\underset{(2.54),(2.55)}{\overset{(2.52),(2.53)}{=}} \left\{ \left(\prod_{n=log_2 N+1}^{2log_2 N} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{2log_2 N} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \cdot \left(\mathbf{I} \otimes \widetilde{\mathbf{I}} \right) \right\} *$$

$$* \left\{ \left(\mathbf{I} \otimes \prod_{n=1}^{2log_2 N} \mathbf{C}_n \right) \cdot \left[\left(\prod_{n=1}^{log_2 N} \mathbf{C}_n \right) \otimes \mathbf{I} \right] \right\} = \tag{2.56}$$

$$\overset{(2.47)}{=} \left\{ \left(\prod_{n=log_2 N+1}^{2log_2 N} \mathbf{C}_n \right) \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{2log_2 N} \mathbf{C}_n \right) \otimes \widetilde{\mathbf{I}} \right] \right\} \cdot \left\{ \left(\prod_{n=1}^{log_2 N} \mathbf{C}_n \right) \otimes \prod_{n=1}^{2log_2 N} \mathbf{C}_n \right\} =$$

$$\overset{(2.47)}{=} \left(\prod_{n=1}^{2log_2 N} \mathbf{C}_n \right) \otimes \left[\bar{\mathbf{I}} \cdot \left(\prod_{n=1}^{2log_2 N} \mathbf{C}_n \right) \right] + \mathbf{Z} \left\{ \left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \cdot \prod_{n=log_2 N+1}^{2log_2 N} \mathbf{C}_n \right) \otimes \left[\widetilde{\mathbf{I}} \cdot \left(\prod_{n=1}^{2log_2 N} \mathbf{C}_n \right) \right] \right\} =$$

$$\overset{(2.44)}{=} \widetilde{\mathbf{C}}_N \otimes \left(\bar{\mathbf{I}} \cdot \widetilde{\mathbf{C}}_N \right) + \mathbf{Z} \left\{ \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \right) \cdot \widetilde{\mathbf{C}}_N \right] \otimes \left(\widetilde{\mathbf{I}} \cdot \widetilde{\mathbf{C}}_N \right) \right\} =$$

$$= \left(\mathbf{I} \cdot \widetilde{\mathbf{C}}_N \right) \otimes \left(\bar{\mathbf{I}} \cdot \widetilde{\mathbf{C}}_N \right) + \mathbf{Z} \left\{ \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \right) \cdot \widetilde{\mathbf{C}}_N \right] \otimes \left(\widetilde{\mathbf{I}} \cdot \widetilde{\mathbf{C}}_N \right) \right\} =$$

$$\overset{(2.47)}{=} \left(\mathbf{I} \otimes \bar{\mathbf{I}} \right) \cdot \left(\widetilde{\mathbf{C}}_N \otimes \widetilde{\mathbf{C}}_N \right) + \mathbf{Z} \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \right) \otimes \widetilde{\mathbf{I}} \right] \cdot \left(\widetilde{\mathbf{C}}_N \otimes \widetilde{\mathbf{C}}_N \right) =$$

$$= \left\{ \left(\mathbf{I} \otimes \bar{\mathbf{I}} \right) + \mathbf{Z} \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \right) \otimes \widetilde{\mathbf{I}} \right] \right\} \cdot \left(\widetilde{\mathbf{C}}_N \otimes \widetilde{\mathbf{C}}_N \right) = \widetilde{\mathbf{P}}_N \left(\widetilde{\mathbf{C}}_N \otimes \widetilde{\mathbf{C}}_N \right)$$

Przepisując w krótkiej formie ostateczną konkluzję, wynikającą z powyższego ciągu równości (2.56) otrzymujemy: (2.57)

$$\widetilde{\mathbf{C}}_{N\times N} \overset{(2.56)}{=} \widetilde{\mathbf{P}}_N \left(\widetilde{\mathbf{C}}_N \otimes \widetilde{\mathbf{C}}_N \right), \quad \widetilde{\mathbf{P}}_N \triangleq \left\{ \left(\mathbf{I} \otimes \bar{\mathbf{I}} \right) + \mathbf{Z} \left[\left(\prod_{n=1}^{log_2 N} \mathbf{P}_n \right) \otimes \widetilde{\mathbf{I}} \right] \right\}$$

gdzie macierz $\widetilde{\mathbf{P}}_N$ jest $N^2 \times N^2$–elementową macierzą permutacji. Z konkluzji (2.57), konstrukcji macierzy $\widetilde{\mathbf{C}}_{N\times N}$ oraz definicji 2.2 i 2.6 jąder jedno i dwuwymiarowych skalowanych transformat kosinusowych wynika natychmiast, że macierz ta realizuje, z dokładnością do permutacji zdeterminowanej przez macierz $\widetilde{\mathbf{P}}_N$, $N \times N$–punktowe dwuwymiarowe skalowane przekształcenie kosinusowe i dodatkowo struktura rozważanej macierzy $\widetilde{\mathbf{C}}_{N\times N}$ określa graf przepływu danych szybkiego algorytmu zunifikowanego wyznaczania dwuwymiarowej dyskretnej transformaty kosinusowej, uwidoczniony na rysunkach 2.15a oraz 2.15b, zawarty w strukturze przedstawionych

wcześniej szybkich zunifikowanych algorytmów wyznaczania przekształceń jednowymiarowych. Wniosek ten oznacza, że zarówno wspomniane algorytmy szybkie dla transformat jednowymiarowych jak i metoda (2.46), określana dalej skrótowo mianem algorytmu JFCT2D, wyznaczania dwuwymiarowego skalowanego przekształcenia kosinusowego mogą być zrealizowane za pomocą procedur obliczeniowych o identycznych strukturach grafów przepływowych. Bezpośrednio z konstrukcji algorytmu JFCT2D można też stwierdzić, iż dla dowolnego $N = 2^m$ wynikający z niego graf przepływu danych składa się z $2N$, odpowiednio zaaranżowanych w ramach struktury wspomnianego grafu, macierzy N punktowych jednowymiarowych przekształceń kosinusowych, co pozwala natychmiast wnioskować, na mocy zależności (2.20), o dokładnych wartościach złożoności obliczeniowych dla operacji arytmetycznych algorytmu (2.46), które wynoszą dla rzeczywistych dodawań/odejmowań oraz rzeczywistych mnożeń odpowiednio:

$$\widetilde{\mathcal{C}}^{+}_{N\times N} \;=\; 2N^2(\,log_2 N^2 - 3\,) + 6N \;\;, \quad \widetilde{\mathcal{C}}^{*}_{N\times N} \;=\; 2N^2(\,log_2 N^2 - 2\,) + 4N \tag{2.58}$$

A zatem JFCT2D jest procedurą szybką, która redukuje rząd złożoności obliczeniowej metody bezpośredniej (2.42) wyznaczania współczynników dwuwymiarowego $N \times N$ punktowego skalowanego przekształcenia kosinusowego z poziomu $\mathcal{O}(N^4)$ do wartości $\mathcal{O}\,(N^2 log_2 N^2)$. Jak wynika natychmiast z postaci grafów przepływowych algorytmu JFCT2D jego złożoność pamięciowa dla przekształcenia $N \times N$–punktowego wynosi $\mathcal{O}(N^2)$. W tym miejscu zakończmy rozważania dotyczące bazowego dla wszystkich pozostałych dwuwymiarowych szybkich procedur zunifikowanych algorytmu JFCT2D. W kolejnych paragrafach niniejszego podrozdziału zajmiemy się analogicznymi algorytmami dla pozostałych, przykładowych transformat dwuwymiarowych, tzn. dwuwymiarowych przekształceń sinusowego, Hartleya oraz Fouriera.

2.3.2.2. Szybkie zunifikowane dwuwymiarowe dyskretne przekształcenie sinusowe

W tej części opracowania przedstawiony zostanie szybki zunifikowany algorytm wyznaczania dwuwymiarowej dyskretnej transformaty sinusowej, którego bazę stanowi zaprezentowany w poprzednim paragrafie algorytm wyznaczania dwuwymiarowego przekształcenia kosinusowego JFCT2D. W celu uniknięcia nieścisłości podajmy najpierw definicję rozważanego dalej przekształcenia dwuwymiarowego.

Definicja 2.7.

Dla dowolnej liczby $N = 2^m$, $m \in \mathbb{N}$ funkcję $\overline{S}_{N\times N} \colon \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ określoną w następujący sposób:

$$\forall\, \mathbf{X} \in \mathbb{R}^{N\times N} \quad \forall\, k,\, l \in \{\,0, \ldots, N-1\}$$

$$\overline{S}_{N\times N}(\,\mathbf{X}\,)_{k,l} \;\triangleq\; \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x_{m,n}\; sin\left[\,\frac{(2\,m+1)\,(k+1)\,\pi}{2\,M}\,\right]\; sin\left[\,\frac{(2\,n+1)\,(l+1)\,\pi}{2\,N}\,\right] \tag{2.59}$$

nazywamy $N \times N$–*punktowym skalowanym dwuwymiarowym dyskretnym przekształceniem sinusowym.*

Z definicji 2.3 i 2.7 odpowiednio jedno i dwuwymiarowych skalowanych transformat sinusowych wynika natychmiast, iż ostatnie z wymienionych przed chwilą przekształceń jest dla dowolnego $N = 2^m$ przekształceniem separowalnym, tzn. dla jąder obu tych przekształceń zachodzi równość $\overline{\mathbf{S}}_{N\times N} = \overline{\mathbf{S}}_N \otimes \overline{\mathbf{S}}_N$. Wobec powyższego faktu, na mocy wzoru rozkładu (2.46) dla dwuwymiarowego dyskretnego skalowanego przekształcenia kosinusowego oraz ze wzoru (2.24) rozkładu jednowymiarowego dyskretnego skalowanego przekształcenia sinusowego, podanego w poprzedniej części rozdziału, można wysnuć natychmiastowy wniosek o postaci szybkiego algorytmu dla dwuwymiarowej skalowanej transformaty sinusowej $\widetilde{\mathbf{S}}_{N\times N}$. Zgodnie z przedstawionymi rozważaniami postać tę można zapisać w następujący sposób:

$$\widetilde{\mathbf{S}}_{N\times N} \overset{(2.21),\,(2.24)}{\underset{(2.46),\,(2.59)}{=}} \mathbf{P}_S \prod_{n=1}^{log_2 N} \left\{ \mathbf{S}_{log_2 N+n} \otimes \overline{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{S}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} * \\ * \left\{ \left[\overline{\mathbf{Q}} \otimes \mathbf{S}_{2log_2 N} + \widetilde{\mathbf{Q}} \otimes (\mathbf{R}\, \mathbf{S}_{2log_2 N}) \right] \cdot \prod_{n=1}^{2log_2 N-1} (\mathbf{I} \otimes \mathbf{S}_n) \right\} * \left\{ \prod_{n=1}^{log_2 N} (\mathbf{S}_n \otimes \mathbf{I}) \right\} \tag{2.60}$$

gdzie dla ustalonego $N \in \mathbb{N}$ będącego dowolną naturalną potęgą liczby dwa $N \times N$ elementowe macierze $\mathbf{S}_n$, $n = 2, \ldots, 2\, log_2 N$ są równe swoim odpowiednikom $\mathbf{C}_n$ o tych samych indeksach występującym we wzorze (2.46), macierz $\mathbf{S}_1$ o identycznej strukturze motylkowej, co macierz $\mathbf{C}_1$ uzyskana jest z tej ostatniej za pomocą reguły zamiany współczynników przedstawionej na rysunku 2.7, zaś $\mathbf{P}_S$ jest $N^2 \times N^2$–elementową macierzą permutacji z główną przeciwdiagonalą wypełnioną jedynkami. Metoda (2.60) wyznaczania współczynników transformaty (2.59) definiuje identyczny, co w przypadku algorytmu `JFCT2D`, graf przepływu danych i dlatego określana będzie, poprzez analogię do procedur obliczeniowych dla przekształceń rozważanych wcześniej, mianem szybkiego zunifikowanego algorytmu wyznaczania dwuwymiarowego skalowanego przekształcenia sinusowego, oraz oznaczana będzie skrótem `JFST2D`. Ze wzorów rozkładu (2.19) i (2.24) zunifikowanych algorytmów jednowymiarowych oraz analogicznych zależności (2.46) i (2.60) dla szybkich procedur obliczeniowych skalowanych przekształceń dwuwymiarowych kosinusowego oraz sinusowego, a także na mocy konkluzji (2.20), (2.25) i (2.58) o złożonościach obliczeniowych i pamięciowych wspomnianych transformat jednowymiarowych oraz, odpowiednio, dwuwymiarowego przekształcenia kosinusowego, można wysnuć natychmiastowy wniosek o złożoności obliczeniowej oraz zapotrzebowaniu pamięciowym procedury `JFST2D`. A zatem dokładna ilość prostych operacji rzeczywistych $\widetilde{\mathcal{S}}^{+}_{N\times N}$ oraz rzeczywistych mnożeń $\widetilde{\mathcal{S}}^{*}_{N\times N}$ dla algorytmu `JFST2D` wynosi odpowiednio:

$$\widetilde{\mathcal{S}}^{+}_{N\times N} \;=\; 2N^2(log_2 N^2 - 3) + 6N \;\;, \quad \widetilde{\mathcal{S}}^{*}_{N\times N} \;=\; 2N^2(log_2 N^2 - 2) + 4N \tag{2.61}$$

zaś zapotrzebowanie pamięciowe jest liniowe względem całkowitego rozmiaru wyznaczanego przekształcenia. Wobec tego algorytm `JFST2D` obliczania $N \times N$–punktowej transformaty sinusowej jest efektywną pamięciowo, zunifikowaną procedurą szybką o rzędach złożoności obliczeniowej i pamięciowej równych odpowiednio wartościom $\mathcal{O}(N^2\, log_2\, N^2)$ oraz $\mathcal{O}(N^2)$. Na tym zakończmy rozważania i przejdźmy do przedstawienia zunifikowanego algorytmu szybkiego dla kolejnego przekształcenia przykładowego, tj. dyskretnej transformaty Hartleya.

2.3.2.3. Szybkie zunifikowane dwuwymiarowe dyskretne przekształcenie Hartleya

W niniejszym podpunkcie zaprezentowany będzie szybki zunifikowany algorytm wyznaczania dwuwymiarowej dyskretnej transformaty Hartleya, bazujący na algorytmie wyznaczania dwuwymiarowego przekształcenia kosinusowego `JFCT2D`. Podajmy definicję rozważanego dalej przekształcenia dwuwymiarowego.

Definicja 2.8.

Dla dowolnej liczby $N = 2^m$, $m \in \mathbb{N}$ funkcję $\overline{H}_{N\times N} : \mathbb{R}^{N\times N} \rightarrow \mathbb{R}^{N\times N}$ określoną w następujący sposób:

$$\forall\, \mathbf{X} \in \mathbb{R}^{N\times N} \quad \forall\, k,\, l \in \{0, \ldots, N-1\}$$
$$\overline{H}_{N\times N}(\mathbf{X})_{k,l} \triangleq \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x_{m,n}\; cas\left[\frac{2\,\pi\, m\, k}{M}\right] cas\left[\frac{2\,\pi\, n\, l}{N}\right] \tag{2.62}$$

nazywamy $N \times N$–*punktowym skalowanym dwuwymiarowym dyskretnym przekształceniem Hartleya.*

Przedstawione tu przekształcenie, określane często w literaturze fachowej mianem transformaty *cas-cas*, jest dwuwymiarowym separowalnym rozszerzeniem dyskretnego przekształcenia Hartleya podanego definicją 2.4 i oryginalne zaproponowane zostało w pracy [93]. Warto przypomnieć, że dla dowolnego $\alpha \in \mathbb{R}$ symbol *cas* definiuje się jako $cas\,\alpha \triangleq cos\,\alpha + sin\,\alpha$. Z separowalności przekształcenia (2.62) wynika, że dla macierzy $\overline{\mathbf{H}}_N$ i $\overline{\mathbf{H}}_{N\times N}$ jedno i dwuwymiarowych skalowanych przekształceń Hartleya, o postaciach wynikających odpowiednio z definicji 2.4 oraz 2.8, zachodzi związek $\overline{\mathbf{H}}_{N\times N} = \overline{\mathbf{H}}_N \otimes \overline{\mathbf{H}}_N$. Stąd zatem oraz na mocy wzorów rozkładu (2.46) dla dwuwymiarowego dyskretnego skalowanego przekształcenia kosinusowego i (2.32) jednowymiarowego dyskretnego skalowanego przekształcenia Hartleya podanego w poprzedniej części rozdziału można wnioskować natychmiast o postaci szybkiego algorytmu obliczeniowego dla dwuwymiarowej skalowanej transformaty Hartleya $\widetilde{\mathbf{H}}_{N\times N}$. Postać ta jest następująca:

$$\widetilde{\mathbf{H}}_{N\times N} \overset{\substack{(2.29),\,(2.32)\\(2.46),\,(2.62)}}{=} \prod_{n=1}^{log_2 N} \left\{ \mathbf{H}_{log_2 N+n} \otimes \overline{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{H}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} *$$
$$* \left\{ \left[\overline{\mathbf{Q}} \otimes \mathbf{H}_{2log_2 N} + \widetilde{\mathbf{Q}} \otimes (\mathbf{R}\, \mathbf{H}_{2log_2 N}) \right] \cdot \prod_{n=1}^{2log_2 N-1} (\mathbf{I} \otimes \mathbf{H}_n) \right\} * \left\{ \prod_{n=1}^{log_2 N} (\mathbf{H}_n \otimes \mathbf{I}) \right\} \cdot \mathbf{P}_H \tag{2.63}$$

gdzie macierz $\mathbf{P}_H$ jest $N^2 \times N^2$–elementową macierzą permutacji określoną podaną niżej zależnością:

$$\mathbf{P}_H = (\mathbf{B} \otimes \mathbf{B}) \cdot (\mathbf{P}_h \otimes \mathbf{P}_h) \cdot (\mathbf{B} \otimes \mathbf{B});$$
$$\mathbf{P}_h \in \mathbb{R}^{N\times N}, \quad \forall\, m \in \{0, \ldots, N/2-1\} \quad \forall\, n \in \{0, \ldots, N-1\} \tag{2.64}$$
$$(\mathbf{P}_h)_{2m,n} = \delta_{m,n} \quad \wedge \quad (\mathbf{P}_h)_{2m+1,n} = \delta_{m,N-n-1};$$

gdzie, jak wcześniej, $\mathbf{B}$ jest $N \times N$–elementową macierzą odwróconego porządku bitowego. Obecność składników $(\mathbf{B} \otimes \mathbf{B})$ w zależności (2.64) określającej postać macierzy permutacji wektora wejściowego we wzorze (2.63) wynika z konwencji zapisu prezentowanych w niniejszej części pracy szybkich algorytmów obliczeniowych dla przekształceń dwuwymiarowych, która zakłada milcząco, iż ich wektory wejściowe są wstępnie zpermutowane zgodnie z dwuwymiarowym odwróconym porządkiem bitowym - założenie to jest konsekwencją zależności (2.44) i definicji (2.46) macierzy określającej postać szybkiego algorytmu wyznaczania dyskretnego przekształcenia kosinusowego. Powracając do opisu macierzy stanowiących główne składniki równania (2.63) należy stwierdzić, iż dla ustalonego N będącego dowolną naturalną potęgą liczby dwa $N \times N$–elementowe macierze $\mathbf{H}_n,\ n = 1, \ldots, 2\, log_2 N - 1$ są równe swoim odpowiednikom $\mathbf{C}_n$ o tych samych indeksach występującym we wzorze (2.46), zaś macierz $\mathbf{H}_{2log_2 N}$ o identycznej strukturze motylkowej co macierz $\mathbf{C}_{2log_2 N}$ uzyskana jest z tej ostatniej za pomocą reguły zamiany współczynników zaprezentowanej w poprzednim podrozdziale na rysunku 2.10.

Na podstawie analogicznej argumentacji do tej przedstawionej w poprzednim podpunkcie, a dotyczącej dwuwymiarowego przekształcenia sinusowego, można stosunkowo łatwo stwierdzić, iż dla dowolnego $N = 2^m$ grafy przepływu danych obydwu metod (2.46) oraz (2.63) wyznaczania współczynników odpowiednio dwuwymiarowego skalowanego przekształcenia kosinusowego i Hartleya są identyczne. Taki sam wniosek dotyczy także wszystkich parametrów efektywnościowych i pojemnościowych rozważanej tu metody, określanej w dalszym ciągu skrótowo mianem algorytmu `JFHT2D`. A zatem algorytm ten jest efektywną pamięciowo zunifikowaną procedurą szybką kalkulacji dwuwymiarowej dyskretnej transformaty Hartleya o złożonościach obliczeniowej i pamięciowej równych rzędami, dla dowolnego przekształcenia $N \times N$ punktowego, odpowiednio $\mathcal{O}(N^2 log_2 N^2)$ oraz $\mathcal{O}(N^2)$. Dla porządku podajmy dokładne ilości prostych operacji rzeczywistych $\widetilde{\mathcal{H}}^{+}_{N\times N}$ i rzeczywistych mnożeń $\widetilde{\mathcal{H}}^{*}_{N\times N}$ dla algorytmu `JFHT2D`, są one równe odpowiednio:

$$\widetilde{\mathcal{H}}^{+}_{N\times N} = 2N^2(\, log_2 N^2 - 3\,) + 6N \ , \quad \widetilde{\mathcal{H}}^{*}_{N\times N} = 2N^2(\, log_2 N^2 - 2\,) + 4N \tag{2.65}$$

Zakończmy w tym miejscu rozważania dotyczące dwuwymiarowego dyskretnego przekształcenia Hartleya i zajmijmy się ostatnim już z dwuwymiarowych algorytmów szybkich prezentowanych w niniejszego opracowania, a przeznaczonym dla celów efektywnego obliczania dwuwymiarowej transformaty Fouriera.

2.3.2.4. Szybkie zunifikowane dwuwymiarowe dyskretne przekształcenie Fouriera

W niniejszym podpunkcie zaprezentowany będzie ostatni już szybki zunifikowany algorytm obliczeniowy przeznacznoy do kalkulacji dwuwymiarowej dyskretnej transformaty Fouriera dla rzeczywistych danych wejściwoych, bazujący na algorytmie wyznaczania dwuwymiarowego przekształcenia kosinusowego `JFCT2D`. Tak jak wcześniej podajmy dla ścisłości definicję rozważanej transformaty dwuwymiarowej.

Definicja 2.9.

Dla dowolnej liczby $N = 2^m$, $m \in \mathbb{N}$ funkcję $\overline{F}_{N\times N}\colon \mathbb{R}^{N\times N} \to \mathbb{C}^{N\times N}$ określoną w następujący sposób:

$$\begin{gathered} \forall\, \mathbf{X} \in \mathbb{R}^{N\times N} \quad \forall\, k,\, l \in \{0, \ldots, N-1\} \\ \overline{F}_{N\times N}(\mathbf{X})_{k,l} \triangleq \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x_{m,n} \cdot e^{-j2\pi mk/M} \cdot e^{-j2\pi nl/N} \end{gathered} \tag{2.66}$$

nazywamy $N \times N$–*punktowym skalowanym dwuwymiarowym dyskretnym przekształceniem Fouriera*.

Bezpośrednio z powyższej definicji oraz definicji 2.5 jednowymiarowego skalowanego przekształcenia Fouriera wynika, że transformata (2.66) jest, podobnie do wszystkich rozważanych wcześniej przekształceń dwuwymiarowych, przekształceniem separowalnym. A zatem macierze transformat (2.35) i (2.66) są dla dowolnych $N = 2^m$ związane zależnością $\overline{\mathbf{F}}_{N\times N} = \overline{\mathbf{F}}_N \otimes \overline{\mathbf{F}}_N$. Wobec tego, biorąc pod uwagę wzory rozkładu (2.46) oraz (2.39), odpowiednio dla dwuwymiarowego dyskretnego skalowanego przekształcenia kosinusowego i jednowymiarowej skalowanej transformaty Fouriera, można wnioskować bezpośrednio o postaci szybkiego algorytmu wyznaczania dwuwymiarowej skalowanej transformaty Fouriera $\overline{\mathbf{F}}_{N\times N}$. Zgodnie z przedstawionymi przed chwilą rozważaniami postać ta może być określona w następujący sposób:

$$\begin{aligned} \widetilde{\mathbf{F}}_{N\times N} &\overset{(2.35),\,(2.39)}{\underset{(2.46),\,(2.66)}{=}} \prod_{n=1}^{log_2 N} \left\{ \mathbf{F}_{log_2 N+n} \otimes \bar{\mathbf{I}} + \mathbf{Z} \left[\left(\prod_{k=1}^{n} \mathbf{P}_k \mathbf{F}_{log_2 N+n} \mathbf{P}_k^T \prod_{k=1}^{n-1} \right) \otimes \widetilde{\mathbf{I}} \right] \mathbf{Z}^T \right\} * \\ &* \left\{ \left[\overline{\mathbf{Q}} \otimes \mathbf{F}_{2log_2 N} + \widetilde{\mathbf{Q}} \otimes (\mathbf{R}\, \mathbf{F}_{2log_2 N}) \right] \cdot \prod_{n=1}^{2log_2 N-1} (\mathbf{I} \otimes \mathbf{F}_n) \right\} * \left\{ \prod_{n=1}^{log_2 N} (\mathbf{F}_n \otimes \mathbf{I}) \right\} \cdot \mathbf{P}_F \end{aligned} \tag{2.67}$$

gdzie macierz $\mathbf{P}_F$ jest $N^2 \times N^2$–elementową macierzą permutacji określoną identycznie jak macierz permutacji $\mathbf{P}_H$ wektora wejściowego dwuwymiarowej skalowanej transformaty Hartleya przedstawiona w poprzednim paragrafie. Równość obydwu

$$\mathbf{J}_F = \mathbf{J}_f \otimes \mathbf{J}_f\,, \qquad \mathbf{J}_f = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & \mathbf{0} & & j \\ & & \ddots & & & \iddots & \\ & & & 1 & j & & \\ & \mathbf{0} & & 1 & & \mathbf{0} & \\ & & & 1 & -j & & \\ & & \iddots & & & \ddots & \\ & 1 & & & \mathbf{0} & & -j \end{pmatrix}_{N\times N} \tag{2.68}$$

macierzy wynika z faktu jednakowych permutacji wektorów wejściowych dla algorytmów szybkich wyznaczania jednowymiarowych skalowanych transformat Hartleya i Fouriera podanych odpowiednio wzorami rozkładów (2.32) i (2.39) w poprzednim podrozdziale. Wszystkie prezentowane w niniejszej pracy algorytmy szybkie,

włącznie z parą rozważanych tu szybkich metod obliczeniowych dla jedno i dwuwymiarowej transformaty Fouriera, są w sensie formalnym przekształceniami rzeczywistymi operującymi na danych rzeczywistych. W przypadku przekształcenia szybkiego (2.67) oznacza to, iż w praktyce bezpośrednie wyznaczenie faktycznych wartości współczynników zespolonych transformaty (2.66) wymaga pewnego dodatkowego, znikomego wysiłku obliczeniowego w końcowym kroku rozpatrywanej szybkiej procedury obliczeniowej. A mianowicie, przy założeniu naturalnego[1] porządku współczynników wektora wyjściowego w (2.67) uzyskanie wspomnianych współczynników zespolonych wymaga przemnożenia jego współrzędnych przez zespoloną macierz rzadką o podanej wzorem (2.68) postaci, co w przypadku operacji jedynie na danych rzeczywistych oznacza konieczność wykonania jeszcze dodatkowych N^2-2 dodawań i $2\,(N^2-1)$ mnożeń rzeczywistych na ostanim etapie działania rozważanego algorytmu.

Powracając do opisu czynników we wzorze rozkładu (2.67) mamy - dla ustalonego N będącego dowolną naturalną potęgą liczby dwa $N\times N$–elementowe macierze rzeczywiste $\mathbf{F}_n,\ n=1,\ldots,2\,log_2N-1$ są równe swoim odpowiednikom $\mathbf{C}_n$ o tych samych indeksach występującym w definicji (2.46), zaś macierz $\mathbf{F}_{2log_2N}$ o identycznej strukturze motylkowej co macierz $\mathbf{C}_{2log_2N}$ uzyskana jest z tej ostatniej za pomocą reguły zamiany współczynników przedstawionej w poprzednim podrozdziale na rysunku 2.12. Podsumowując, z przedstawionych powyżej rozważań i z postaci wzoru (2.67) wynika natychmiast fakt, iż macierz $\widetilde{\mathbf{F}}_{N\times N}$ określa szybką metodę obliczeniową wyznaczania współczynników dwuwymiarowego dyskretnego skalowanego przekształcenia Fouriera, zwaną dalej skrótowo algorytmem `JFFT2D`, o identycznej strukturze grafu przepływu danych, co wszystkie przedstawione wcześniej szybkie metody wyznaczania przekształceń jedno i dwuwymiarowych. Stąd zaś na mocy argumentacji analogicznej do tej zastosowanej w przypadku dwóch poprzednich przekształceń dwuwymiarowych i poczynionych wyżej uwag na temat ostatniego kroku algorytmu `JFFT2D` można łatwo dociec, że dokładne liczby dodawań/odejmowań rzeczywistych oraz rzeczywistych mnożeń niezbędnych dla obliczenia $N\times N$ - punktowej skalowanej transformaty Fouriera metodą (2.67) wynoszą:

$$\widetilde{\mathcal{F}}^{+}_{N\times N}\ =\ 2N^2(\,log_2N^2-2\,\tfrac{1}{2}\,)+6N-2\ ,\quad \widetilde{\mathcal{F}}^{*}_{N\times N}\ =\ 2N^2(\,log_2N^2-1\,)+4N-2 \qquad (2.69)$$

co oznacza, że złożoność obliczeniowa rozważanego algorytmu jest rzędu $\mathcal{O}(N^2log_2N^2)$. W przypadku złożoności pamięciowej oczywisty jest fakt, wynikający chociażby ze wzoru rozkładu (2.67), że tak samo jak dla wszystkich rozważanych dotychczas zunifikowanych dwuwymiarowych algorytmów szybkich, tak i dla analizowanej tu metody `JFFT2D` rozpatrywana złożoność jest liniowa względem całkowitego rozmiaru wybranego $N\times N$ - punktowego przekształcenia dyskretnego i wynosi $\mathcal{O}(N^2)$. A zatem algorytm `JFFT2D` zdefiniowany wzorem (2.67) jest efektywną pamięciowo zunifikowaną procedurą szybką kalkulacji dwuwymiarowej dyskretnej transformaty Fouriera. Metoda szybka `JFFT2D` obliczania dwuwymiarowego dyskretnego przekształcenia Fouriera jest już ostatnią z serii prezentowanych w niniejszej pracy zunifikowanych metod wyznaczania dwuwymiarowych przekształceń dyskretnych. W następnym podpunkcie przedstawione zostanie podsumowanie oraz wnioski, użyteczne z punktu

[1] to znaczy porządku *leksykograficznego*, zgodnego ze sposobem działania operatora $vec\,\{\,\cdot\,\}$

widzenia celów niniejszej pracy, jakie nasuwają się po przedstwieniu i analizie konstrukcji zaprezentowanych tu zunifikowanych metod szybkich wyznaczania czterech przykładowych, dyskretnych przekształceń dwuwymiarowych.

2.3.2.5. Uwagi końcowe

W bieżącym podrozdziale zaprezentowano szybkie algorytmy zunifikowane wyznaczania wybranych dwuwymiarowych przekształceń dyskretnych i poddano analizie ich charakterystyki efektywnościowo pamięciowe.

Z rozważań tych wynika, że prezentowane algorytmy dla wybranych przekształceń przykładowych posiadają wspólny, ogólny schemat obliczeniowy, co więcej schemat ten w sensie grafów przepływu danych jest identyczny z przedstawionym w poprzednim podrozdziale schematem obliczeniowym dla przekształceń jednowymiarowych. Jak wynika z wniosków (2.58), (2.61), (2.65) i (2.69) złożoność obliczeniowa zunifikowanych algorytmów szybkich wyznzaczania przekształceń dwuwymiarowych ma charakter liniowo-logarytmiczny względem całkowitego rozmiaru wybranego przekształcenia $N \times N$–punktowego, tzn. wynosi $\mathcal{O}\,(N^2 log_2 N^2)$, redukując tym samym złożoność obliczeniową procedur bezpośrednich wyznaczania rozważanych przekształceń o rząd wielkości z poziomu $\mathcal{O}(N^4)$. Zapotrzebowanie pamięciowe jest liniowe względem wspomnianego rozmiaru i jest równe rzędem $\mathcal{O}(N^2)$. Jak wynika z przytoczonych wyżej zależności dokładne liczby rzeczywistych operacji arytmetycznych potrzebnych do wyznaczenia współczynników wynikowych każdego[1] z przykładowych przekształceń dyskretnych są równe:

$$\widetilde{\mathcal{J}}^{+}_{N\times N} \;=\; 2N^2(\,log_2 N^2 - 3\,) + 6N \;\;, \quad \widetilde{\mathcal{J}}^{*}_{N\times N} \;=\; 2N^2(\,log_2 N^2 - 2\,) + 4N \tag{2.70}$$

gdzie $\widetilde{\mathcal{J}}^{+}_{N\times N}$ oraz $\widetilde{\mathcal{J}}^{*}_{N\times N}$ są odpowiednio liczbą dodawań/odejmowań rzeczywistych oraz rzeczywistych mnożeń niezbędnych do wyznaczenia współczynników wybranej dwuwymiarowej transformaty docelowej.

Podsumowując, w niniejszym podrozdziale przedstawiono algorytmy kalkulacji wybranych, znanych dwuwymiarowych transformat ortogonalnych prowadzące do sformułowania jednolitego, ogólnego, szybkiego i efektywnego pamięciowo schematu ich obliczania. Na podstawie tych przesłanek można wnioskować, że schemat ten ma efektywny i uniwersalny zarazem charakter i jako taki może stanowić bezpośrednią podstawę dla konstrukcji algorytmów obliczeniowych wyznaczania innych, nowych dwuwymiarowych przekształceń dyskretnych, dedykowanych do wybranych zadań dwuwymiarowej filtracji liniowej sygnałów, w szczególności do zadania kompresji obrazów. Co więcej, korzystna forma grafów przepływu danych proponowanego szybkiego schematu obliczeniowego może być w prosty sposób zaimplementowana w postaci sieci neuronowej, służącej do adaptacji dwuwymiarowych przekształceń dyskretnych do zadania kompresji wybranych rodzajów obrazów naturalnych. Architekturze takiej właśnie sieci poświęcony jest następny rozdział.

[1] pomijając ostatni krok procedury `JFFT2D` konwersji wartości pomocniczych do współczynników wynikowych

3. REALIZACJE NEURONOWE

3.1. Wprowadzenie

W niniejszej części monografii przedstawiona zostanie architektura sieci neuronowej przeznaczonej do zadania kompresji obrazów. Architektura ta, opierająca się bezpośrednio o przedstwione w poprzednim podrozdziale szybkie algorytmy zunifikowane wyznaczania dwuwymiarowych przekształceń dyskretnych, została oryginalnie zaproponowana w pracy [27], a następnie poddana badaniom, których rezultaty zostały w istotnej części przedstawione między innymi w publikacjach [28, 32, 65, 67].

Podstawowym wyróżnikiem proponowanej architektury sieci neuronowej wśród innych architektur sieci neuronowych przeznaczonych do zadania kompresji obrazów, przedstwionych chocażby w opracowaniach [29, 30, 62–64, 94, 95], jest jej efektywność obliczeniowa i związana z nią bezpośrednio charakterystyka pamięciowa, co w przypadku zagadnienia kompresji ma kluczowe znaczenie, a także prostota implementacyjna objawiająca się szybkością realizacji procesów treningu i wyznaczania odpowiedzi nauczonej sieci, regularnością struktury rzadkiej połączeń w ramach poszczególnych jej warstw oraz możliwością zastosowania bez żadnych modyfikacji standardowych procedur optymalizacyjnych dla wielowarstwowych sieci neuronowych, takich jak chociażby metoda wstecznej propagacji błędu, która zostanie omówiona dalej. Następny podpunkt dotyczy już szczegółowych kwestii dotyczących konstrukcji i sposobów optymalizacji zaprezentowanej w niniejszym opracowaniu sieci neuronowej do kompresji obrazów.

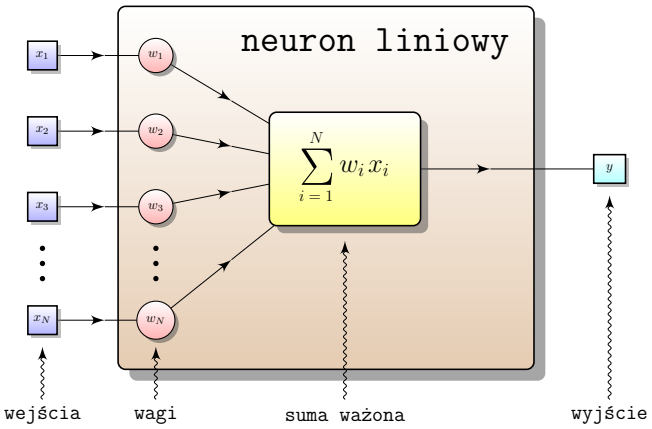
3.2. Perceptron wielowarstwowy i metoda wstecznej propagacji błędu

Jednokierunkowe, wielowarstwowe sieci neuronowe, a w szczególności sieć typu *perceptron*, ze względu na prostotę budowy, wysoką efektywność zarówno procesu optymalizacji jak i późniejszego etapu operacyjnego ich cyklu życiowego oraz dobre charakterystyki jakościowe uzyskiwanych przez nie rozwiązań stanowią jedną z najczęściej stosowanych klas sieci neuronowych, szczególnie w problemach związanych z przetwarzaniem sygnałów. Literatura poświęcona rozważanej grupie sieci jest bardzo bogata, zaś dostępne na ich temat pozycje obejmują często wyczerpujące opisy praktycznych technik zastosowania rozważanych sieci jak i zagadnień teoretycznych ich dotyczących, patrz np. [28–30, 63, 64, 95, 96]. W zaprezentowanym w niniejszym opracowaniu rozwiązaniu problemu kompresji obrazów sieć typu wie-

lowarstwowy perceptron stanowi jego integralny i bardzo istotny komponent. Z tego powodu autor postanowił poświęcić tematyce tej sieci bieżący podrozdział i przybliżyć te jej własności, które wydają się być najistotniejszymi z punktu widzenia celów niniejszego opracowania. Na wstępie zatem podana oraz krótko scharakteryzowana zostanie architektura rozważanej sieci neuronowej wraz z metodą treningu. Kolejny podpunkt dotyczyć będzie analizy efektywnościowo – pojemnościowej cyklu roboczego wspomnianej sieci oraz procesu jej optymalizacji wybraną metodą treningową. Zagadnienia te mają zasadniczy wpływ na całościową wydajność algorytmów kompresji prezentowanych w niniejszym opracowaniu.

3.2.1. Architektura perceptronu wielowarstwowego

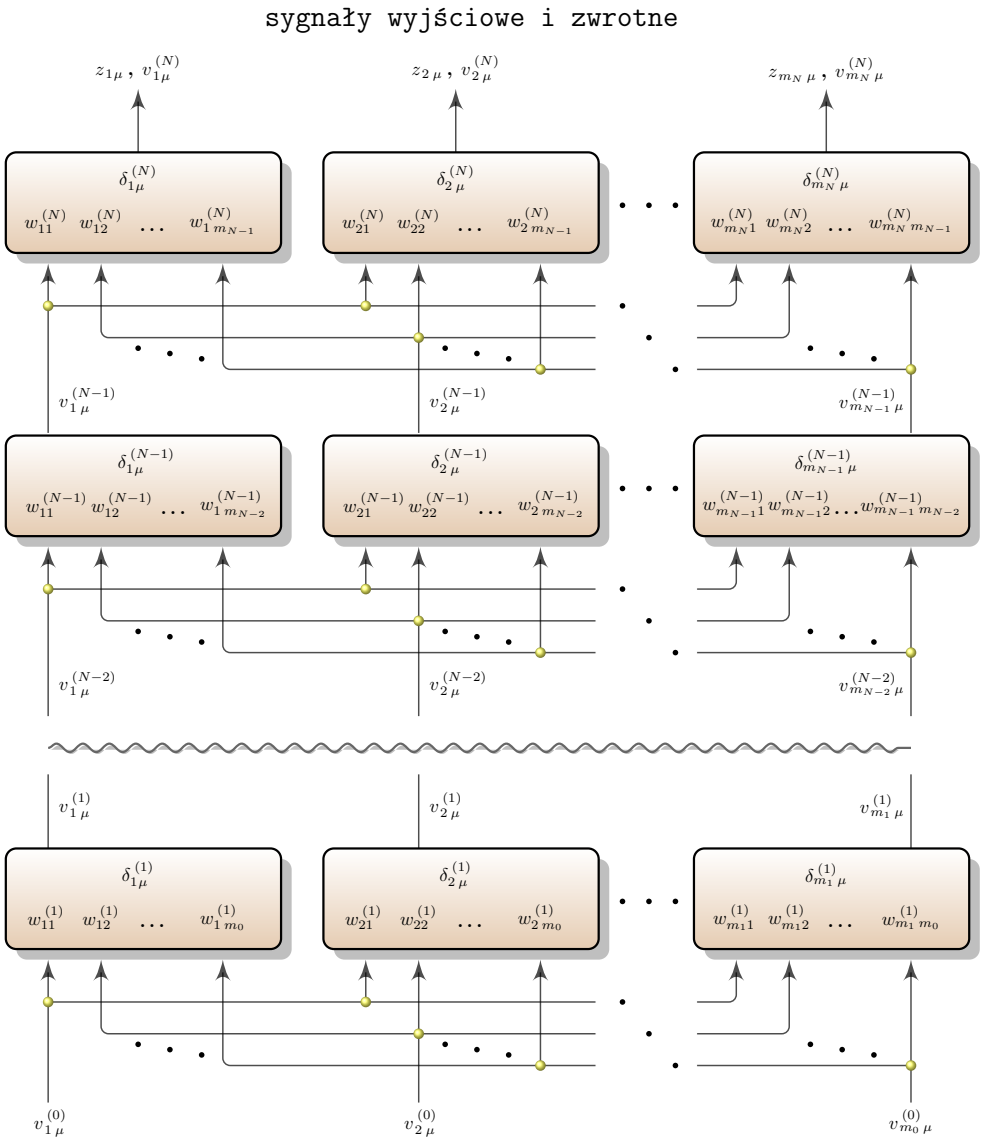
Jak wspomniano we wprowadzeniu do niniejszego rozdziału proponowana architektura sieci neuronowej do kompresji obrazów opiera się bezpośrednio na postaciach zaprezentowanych wcześniej szybkich algorytmów zunifikowanych wyznaczania jedno/dwuwymiarowych przekształceń dyskretnych z wykorzystaniem standardowych liniowych sieci neuronowych. Ich podstawowymi elementami obliczeniowymi są neurony liniowe. Schemat funkcjonalny neuronu liniowego przedstawia poniższy rysunek.



Rys. 3.1: Schemat funkcjonalny standardowego neuronu liniowego

Neuron liniowy oblicza sumę ważoną składników swojego wektora wejściowego $\mathbf{x} = [\, x_1\ x_2\ \ldots\ x_N \,]^T \in \mathbb{R}^N$, $N \in \mathbb{N}$ z odpowiednimi współczynnikami jego wektora wagowego $\mathbf{w} = [\, w_1\ w_2\ \ldots\ w_N \,]^T \in \mathbb{R}^N$ determinującymi funkcjonalność neuronu w ramach sieci. Wyjście neuronu liniowego $y \in \mathbb{R}$ jest zatem iloczynem skalarnym jego wektora wejściowego $\mathbf{x}$ i wektora wagowego $\mathbf{w}$. Elementy tego typu, zgrupowane w warstwy i odpowiednio połączone w sposób jednokierunkowy tworzą sieci neuronowe zwane w literaturze perceptronami. Rysunek 3.2 przedstawia ogólny model wielowarstwowej, jednokierunkowej liniowej sieci neuronowej, czyli wielowarstwowego perceptronu, określanego często po angielsku mianem sieci `MLP`.[1] Symbole przedstawione na tym rysunku zostaną omówione dokładnie w dalszej części podpunktu. Sieci tego typu wraz z matematycznym modelem neuronu zostały oryginalnie zaproponowane

[1] ang. *multilayer* (*linear*) *perceptron*



Rys. 3.2: Interpretacja graficzna modelu wielowarstwowego perceptronu

i szeroko rozpropagowane w środowisku naukowym między innymi za sprawą publikacji [97–99]. Obecnie ze względu na szereg bardzo istotnych zalet, takich jak prostota budowy, niskie koszty implementacji, zarówno sprzętowej (analogowej, cyfrowej i hybrydowej – [100–103]) jak i programowej, wysoka efektywność czasowa związana między innymi z możliwością zrównoleglenia obliczeń oraz opracowanie skutecznych metod treningowych tego typu sieci sprawiły, że perceptrony wielowarstwowe (w tym również liniowe) wraz z techniką adaptacji wag metodą wstecznej propagacji błędu są jednym z najczęściej wykorzystywanych typów sieci neuronowych w rozmaitych, rzeczywistych zagadnieniach technicznych. Poniżej przedstawione zostanie teoretyczne uzasadnienie poprawności metody wstecznej propagacji błedu, jako procedury obliczeniowej prowadzącej do prawidłowej adaptacji wag w zadaniach opty-

malizacyjnych wykorzystujących wielowarstwowe jednokierunkowe sieci neuronowe [28–30, 63].

Dla uproszczenia dalszego opisu i bez utraty ogólności podjętych rozważań załóżmy, że mamy do czynienia z siecią liniową $\mathcal{P}$ o pełnym schemacie połączeń między neuronami sąsiednich warstw. Przyjmijmy ponadto, że neurony warstwy wejściowej (nie przedstawionej na rysunku 3.2) oznaczonej indeksem zero są jednostkami kopiującymi, nie biorącymi udziału w procesie treningowym, których jedynym zadaniem jest propagacja sygnałów wejściowych sieci do wejść neuronów jej pierwszej warstwy podlegających adaptacji. Zgodnie z tak sformułowaną konwencją indeksacyjną badana sieć jest $N+1$, $N \in \mathbb{N}$ warstwową siecią liniową o warstwie wejściowej oznaczonej indeksem zero, $N-1$ warstwach ukrytych ponumerowanych odpowiednio od 1 do $N-1$ oraz warstwie wyjściowej o indeksie N. Oznaczmy symbolami $m_k \in \mathbb{N}$, $k = 0, \ldots, N$ liczby neuronów w każdej z warstw rozważanej sieci. Zdefiniujmy teraz zbiór wzorców treningowych sieci $\mathcal{P}$

$$\mathbf{\Omega} \triangleq \{ ([v_{1\mu}^{(0)} v_{2\mu}^{(0)} \ldots v_{m_0\mu}^{(0)}]^T, [z_{1\mu} z_{2\mu} \ldots z_{m_N\mu}]^T) \in \mathbb{R}^{m_0} \times \mathbb{R}^{m_N} ; \mu = 1, \ldots, M \in \mathbb{N} \} \quad (3.1)$$

gdzie $\mathbf{\Omega}$ jest zbiorem $M \in \mathbb{N}$ par wektorów wejściowych i odpowiadających im spodziewanych wektorów wyjściowych sieci, stanowiących jej pożądane odpowiedzi na wspomniane wyżej sekwencje pobudzające. Niech $\mathbf{w} \in \mathbb{W}$ będzie (uporządkowanym w dowolny sposób) wektorem wszystkich rzeczywistych wag sieci $\mathcal{P}$, gdzie $\mathbb{W}$ jest przestrzenią wag rozważanej sieci. Zdefiniujmy problem nauki sieci $\mathcal{P}$ następująco:

$$\underset{\mathbf{w} \in \mathbb{W}}{argmin} \{ E : \mathbb{W} \to \mathbb{R} ; \quad E(\mathbf{w}) \triangleq \frac{1}{2} \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} (z_{i\mu} - v_{i\mu}^{(N)})^2 \} \quad (3.2)$$

Zakładając istnienie rozwiązania problemu (3.2), z podanej wyżej definicji funkcji błędu E sieci wynika natychmiast, że jej wartością minimalną jest zero, oraz wartość ta osiągana jest w punktach $\mathbf{w} \in \mathbb{W}$ przestrzeni wag sieci, w których odpowiedzi na wszystkich jej wyjściach są identyczne z zadanymi odpowiedziami spodziewanymi dla każdego ze wzorców uczących, stanowiących elementy zdefiniowanego zależnością (3.1) zbioru treningowego $\mathbf{\Omega}$. Minimalizację E funkcji błędu można przeprowadzić metodą najszybszego spadku gradientu. Jedynym warunkiem koniecznym, niezbędnym dla próby zastosowania tej metody do rozwiązania problemu (3.2) jest znalezienie ogólnej postaci (ujemnego) gradientu funkcji błędu E w dowolnym punkcie $\mathbf{w}$ przestrzeni wag $\mathbb{W}$ sieci $\mathcal{P}$. Spróbujmy zatem znaleźć postać wspomnianego gradientu, najpierw jednak, dla sformalizowania dalszych rozważań, wprowadźmy oznaczenia: dla i, j, k zmieniających się zgodnie z topologią badanej sieci $\mathcal{P}$, zdefiniujmy:

$w_{ij}^{(k)}$ - *j-ta waga i-tego neuronu k-tej warstwy sieci* (3.3a)

$v_{i\mu}^{(k)} = \sum_{j=1}^{m_{k-1}} w_{ij}^{(k)} v_{j\mu}^{(k-1)}$ - *wyjście i-tego neuronu k-tej warstwy dla μ-tego wzorca* (3.3b)

Określmy rekursywnie *sygnał zwrotny* dla i-tego neuronu k-tej warstwy sieci przy μ-tym wzorcu uczącym.

Definiujemy:

$$\delta_{i\mu}^{(k)} \stackrel{\Delta}{=} \begin{cases} z_{i\mu} - v_{i\mu}^{(k)} & dla \quad k = N \\ \sum\limits_{j=1}^{m_{k+1}} w_{ji}^{(k+1)} \delta_{j\mu}^{(k+1)} & dla \quad k = N-1, \ldots, 1 \end{cases} \tag{3.4}$$

Załóżmy dalej, że $\mathbb{N} \ni l \leqslant k \in \{\, 1, \ldots, N-1 \,\}$, $\mu \in \{\, 1, \ldots, M \,\}$, $p \in \{\, 1, \ldots, m_l \,\}$ oraz $q \in \{\, 1, \ldots, m_{l-1} \,\}$. Wówczas korzystając z (3.3) i (3.4) dla dowolnego punktu $\mathbf{w} \in \mathbb{W}$ przestrzeni wag sieci otrzymujemy:

$$\sum_{j=1}^{m_{k+1}} \delta_{j\mu}^{(k+1)} \frac{\partial\, v_{j\mu}^{(k+1)}}{\partial\, w_{pq}^{(l)}} \stackrel{(3.3)}{=} \sum_{j=1}^{m_{k+1}} \delta_{j\mu}^{(k+1)} \frac{\partial \sum_{i=1}^{m_k} w_{ji}^{(k+1)} v_{i\mu}^{(k)}}{\partial\, w_{pq}^{(l)}} = \sum_{j=1}^{m_{k+1}} \delta_{j\mu}^{(k+1)} \left(\sum_{i=1}^{m_k} w_{ji}^{(k+1)} \frac{\partial\, v_{i\mu}^{(k)}}{\partial\, w_{pq}^{(l)}} \right) =$$

$$= \sum_{i=1}^{m_k} \left(\sum_{j=1}^{m_{k+1}} w_{ji}^{(k+1)} \delta_{j\mu}^{(k+1)} \right) \frac{\partial\, v_{i\mu}^{(k)}}{\partial\, w_{pq}^{(l)}} \stackrel{(3.4)}{=} \sum_{i=1}^{m_k} \delta_{i\mu}^{(k)} \frac{\partial\, v_{i\mu}^{(k)}}{\partial\, w_{pq}^{(l)}} \tag{3.5}$$

Przepisując zależność (3.5) w bardziej przystępnej formie dostajemy: dla dowolnego $\mathbf{w} \in \mathbb{W}$ zachodzi:

$$\sum_{i=1}^{m_{k+1}} \delta_{i\mu}^{(k+1)} \frac{\partial\, v_{i\mu}^{(k+1)}}{\partial\, w_{pq}^{(l)}} \stackrel{(3.5)}{=} \sum_{i=1}^{m_k} \delta_{i\mu}^{(k)} \frac{\partial\, v_{i\mu}^{(k)}}{\partial\, w_{pq}^{(l)}} \tag{3.6}$$

Wobec tego dowolna współrzędna ujemnego gradientu $-\nabla E$ funkcji błędu w punkcie $\mathbf{w} \in \mathbb{W}$ jest równa[1]

$$-\frac{\partial\, E\,(\,\mathbf{w}\,)}{\partial\, w_{pq}^{(l)}} \stackrel{(3.2)}{=} \frac{\partial \left[-\frac{1}{2} \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} (\; z_{i\mu} - v_{i\mu}^{(N)} \;)^2 \right]}{\partial\, w_{pq}^{(l)}} = -\sum_{\mu=1}^{M} \sum_{i=1}^{m_N} (\; z_{i\mu} - v_{i\mu}^{(N)} \;) \frac{\partial\,(\; z_{i\mu} - v_{i\mu}^{(N)} \;)}{\partial\, w_{pq}^{(l)}} =$$

$$= \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} (\; z_{i\mu} - v_{i\mu}^{(N)} \;) \frac{\partial\, v_{i\mu}^{(N)}}{\partial\, w_{pq}^{(l)}} \stackrel{(3.4)}{=} \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} \delta_{i\mu}^{(N)} \frac{\partial\, v_{i\mu}^{(N)}}{\partial\, w_{pq}^{(l)}} =$$

$$\stackrel{(n-l)\,\times\,(3.6)}{=} \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} \delta_{i\mu}^{(l)} \frac{\partial\, v_{i\mu}^{(l)}}{\partial\, w_{pq}^{(l)}} \stackrel{(3.3)}{=} \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} \delta_{i\mu}^{(l)} \frac{\partial \left(\sum_{j=1}^{m_{l-1}} w_{ij}^{(l)} v_{j\mu}^{(l-1)} \right)}{\partial\, w_{pq}^{(l)}} = \tag{3.7}$$

$$= \sum_{\mu=1}^{M} \sum_{i=1}^{m_N} \sum_{j=1}^{m_{l-1}} \delta_{i\mu}^{(l)} \frac{\partial\, v_{j\mu}^{(l-1)} w_{ij}^{(l)}}{\partial\, w_{pq}^{(l)}} = \sum_{\mu=1}^{M} \delta_{p\mu}^{(l)}\, v_{q\mu}^{(l-1)}$$

Ostatecznie więc, na mocy dowolności wyboru $\mathbf{w}$, k, l, p, q, przepisując w przystępnej formie powyższą zależność otrzymujemy: dla dowolnych $l \in \{1, \ldots, N\}$, $p \in \{1, \ldots, m_l\}$ i $q \in \{\, 1, \ldots, m_{l-1} \,\}$ zachodzi:

$$\forall\, \mathbf{w} \in \mathbb{W} \quad -\frac{\partial\, E\,(\,\mathbf{w}\,)}{\partial\, w_{pq}^{(l)}} \stackrel{(3.7)}{=} \sum_{\mu=1}^{M} \delta_{p\mu}^{(l)}\, v_{q\mu}^{(l-1)} \tag{3.8}$$

A zatem biorąc pod uwagę zależność (3.8), można wyznaczyć kolejne przybliżenie rozwiązania problemu (3.2) metodą najszybszego spadku gradientu w $t+1$ kroku

[1] zapis $(n-l) \times (3.6)$ w jednym z przejść wyprowadzenia (3.7) oznacza $(n-l)$ - krotne zastosowanie zależności (3.6)

iteracyjnym rozważanej procedury optymalizacyjnej w następujący sposób:

$$w_{pq}^{(l)}(t+1) = w_{pq}^{(l)}(t) - \eta \frac{\partial E(\mathbf{w}(t))}{\partial w_{pq}^{(l)}} \overset{(3.8)}{=} w_{pq}^{(l)}(t) + \eta \sum_{\mu=1}^{M} \delta_{p\mu}^{(l)}(t) v_{q\mu}^{(l-1)}(t) \quad (3.9)$$

Refleksja nad formą zależności (3.9) określającą pojedynczy krok iteracyjnej techniki minimalizacji funkcji błędu (3.2) w przestrzeni wag sieci $\mathcal{P}$ metodą najszybszego spadku gradientu, skłania natychmiast do wniosku, że wyznaczenie (ujemnego) gradientu funkcji kosztu E w dowolnym punkcie $\mathbf{w} \in \mathbb{W}$ może być zrealizowane z wykorzystaniem dostępnej struktury optymalizowanej sieci neuronowej, kolejno, poprzez propagację prostą wejściowych sygnałów uczących $v_{i\mu}^{(0)},\ i = 1, \ldots, m_0,\ \mu = 1, \ldots, M$ do momentu uzyskania odpowiadających im wartości $v_{i\mu}^{(N)},\ i = 1, \ldots, m_N$ na wyjściach rozważanej sieci, wyznaczenie błędów $\delta_{i\mu}^{(N)},\ i = 1, \ldots, m_N$ odpowiedzi sieci względem przyporządkowanych im wartości oczekiwanych $z_{i\mu},\ i = 1, \ldots, m_N$ ze zbioru treningowego $\mathbf{\Omega}$ i propagację wsteczną określonych w ten sposób sygnałów zwrotnych $\delta_{i\mu}^{(k)}$, $k = 1, \ldots, N-1,\ i = 1, \ldots, m_k$ dla poszczególnych neuronów sieci odpowiednio od jednostek wyjściowych, poprzez neurony warstw ukrytych, sukcesywnie, aż do jednostek pierwszej warstwy sieci $\mathcal{P}$, podlegających adaptacji.

Klasa procedur optymalizacyjnych wykorzystujących opisaną wyżej technikę sieciowej implementacji wyznaczania gradientu funkcji kosztu w przestrzeni wag wybranych (liniowych) sieci neuronowych poprzez wsteczną propagację odpowiednio zdefiniowanych sygnałów zwrotnych dla poszczególnych jednostek obliczeniowych rozważanych sieci neuronowych określana jest właśnie ogólnym mianem rodziny *metod wstecznej propagacji błędu* i stanowi jedną z najczęściej wykorzystywanych grup procedur optymalizacyjnych dla sieci wielowarstwowych. W szczególności technika wyznaczania gradientu funkcji kosztu oparta o bezpośrednie zastosowanie zależności (3.9) nazywana jest często w literaturze fachowej metodą *off-line* wstecznej propagacji błędu. Choć metoda ta bywa używana w praktyce to jednak posiada ona kilka wad, które ograniczają jej szersze zastosowanie [28–30, 63]. Wady te doprowadziły do powstania tzw. metody *on-line*[1] wstecznej propagacji błędu nauczania sieci wielowarstwowych. W metodzie tej poprawka wektora wagowego dokonywana jest na podstawie znajomości pojedynczego zbioru wyjść i sygnałów zwrotnych poszczególnych neuronów sieci, wyznaczonego dla ustalonego, pojedynczego wzorca uczącego. Tutaj wzór poprawki wektora wagowego pojedynczego kroku nauki sieci ma postać:

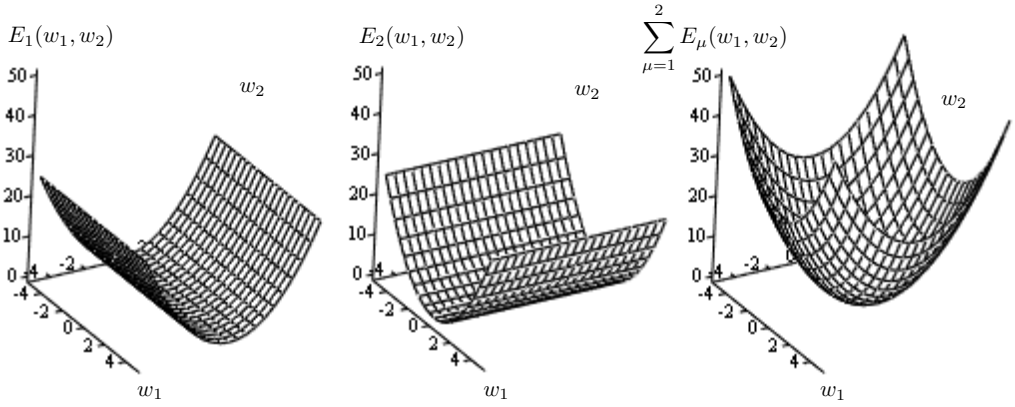
$$w_{pq}^{(l)}(t+1) = w_{pq}^{(l)}(t) - \eta \frac{\partial E_{\mu_0}(\mathbf{w}(t))}{\partial w_{pq}^{(l)}} \underset{(3.10b)}{\overset{(3.2),(3.8)}{=}} w_{pq}^{(l)}(t) + \eta\, \delta_{p\mu_0}^{(l)}(t) v_{q\mu_0}^{(l-1)}(t) \quad (3.10a)$$

$$E_{\mu_0}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{m_N} (z_{i\mu_0} - v_{i\mu_0}^{(N)})^2\,; \quad \mu_0 \in \{1, \ldots, M\} \quad (3.10b)$$

gdzie E_{μ_0} pojedynczym składnikiem funkcji błędu (3.2) dla ustalonego wzorca uczącego $\mu_0 \in \{1, \ldots, M\}$, zaś krok $t \in \mathbb{N}$ iteracji metody (3.10) on-line wstecznej propagacji błędu może być utożsamiany z prezentacją uczonej sieci pojedynczego wzorca treningowego. Technika on-line nauki sieci dokonuje optymalizacji każdego składnika

[1] zwanej także często *metodą gradientu stochastycznego*

funkcji celu (3.2) z osobna. Rysunek 3.3 ilustruje różnicę podejść on-line i off-line w problemie treningu pojedynczego neuronu liniowego o dwóch wejściach dla dwóch wzorców uczących tzw. *regułą delta*, będącą zredukowaną wersją metody wstecznej propagacji błędu, celem wyznaczenia współczynników jednorodnego układu dwóch równań liniowych z dwiema niewiadomymi.



Rys. 3.3: Przykłady kształtów funkcji błędów optymalizacji sieci liniowej metodami *on-line* i *off-line*

3.2.2. Analiza efektywności perceptronu wielowarstwowego

Metoda wstecznej propagacji błędu z techniką on-line nauki wielowarstwowego perceptronu jest używana podczas wszystkich testów sieci wykonanych w ramach niniejszej pracy. Dla potrzeb analizy efektywnościowo pojemnościowej rozważanej sieci warto podać jawnie postać algorytmu wstecznej propagacji błędu. Jest on przedstawiony na schemacie 3.1 i określany będzie dalej zwyczajowo przyjętym angielskim skrótem BP.[1] Poddajmy najpierw analizie zagadnienie złożoności obliczeniowej pojedynczego kroku algorytmu 3.1 rozszerzając rozważania, co niezbędne dla dociekań zawartych dalszej części pracy, na przypadek liniowych sieci wielowarstwowych nie cechujących się jednak pełnym schematem połączeń w ramach kolejnych swoich warstw. Łatwo stwierdzić, że również i w tej sytuacji algorytm 3.1 pozostaje w mocy – wystarczy tutaj przyjąć we wzorze (3.1.2) zerowe wartości wag odpowiadających nieistniejącym połączeniom neuronów w ramach kolejnych warstw sieci i, podobnie, pominąć we wzorze (3.1.3) wspomniane przed chwilą wagi. Pojedynczy krok optymalizacji sieci procedurą 3.1 składa się z trzech zasadniczych punktów:

- propagacji prostej wybranego sygnału wejściowego w przód sieci aż do jej warstwy wyjściowej,
- wstecznej propagacji sygnałów zwrotnych określonych na podstawie różnic wartości oczekiwanych na wyjściach sieci z sygnałami uzyskanymi w wyniku wykonania poprzedniego punktu,
- dokonaniu na podstawie danych uzyskanych w dwóch poprzednich krokach poprawek wszystkich wag sieci podlegających adaptacji.

[1] ang. – *error backpropagation*

Alg. 3.1: Metoda wstecznej propagacji błędu perceptronu wielowarstwowego

$\mathcal{P}$: sieć o $N+1$, $N \in \mathbb{N}$ warstwach i $m_k \in \mathbb{N}$, $k = 0, \ldots, N$ neuronach w każdej z warstw
$\mathbf{\Omega}$: $\mathbb{N} \ni M$ – elementowy zbiór treningowy określony tak, jak w definicji (3.1)
$\mathcal{L}$: lista wzorców uczących
$\mathcal{S}$: kryterium stopu
η : współczynnik nauczania będący niewielką liczbą dodatnią

Wylosuj każdą z wag $w_{pq}^{(l)}$, $l = 1, \ldots, N-1$, $p = 1, \ldots, m_l$ oraz $q = 1, \ldots, m_{l-1}$ sieci $\mathcal{P}$ jako małą liczbę rzeczywistą ;

powtarzaj

Wstaw do listy $\mathcal{L}$ wszystkie wzorce treningowe ze zbioru $\mathbf{\Omega}$;

powtarzaj

Wybierz losowo wzorzec $([\, v_{1\mu}^{(0)} \ \ldots \ v_{m_0\mu}^{(0)} \,]^T, \ [\, z_{1\mu} \ \ldots \ z_{m_N\mu} \,]^T) \in \mathbb{R}^{m_0} \times \mathbb{R}^{m_N}$, $\mu \in \{ 1, \ldots, |\mathcal{L}| \}$ z listy wzorców $\mathcal{L}$, a następnie usuń go ze wspomnianej listy ;

Przepuść sygnał wejściowy $[\, v_{1\mu}^{(0)} \ v_{2\mu}^{(0)} \ \ldots \ v_{m_0\mu}^{(0)} \,]^T$w przód sieci $\mathcal{P}$ poczynając od warstwy 0, na warstwie N zaś kończąc, wyznaczając i zapamiętując jednocześnie wyjścia $v_{i\mu}^{(k)}$, $k = 0, \ldots, N$, $i = 0, \ldots, m_k$ wszystkich neuronów sieci ;

Oblicz sygnały zwrotne $\delta_{i\mu}^{(N)}$, $i = 1, \ldots, m_N$ wszystkich neuronów wyjściowych:

$$\delta_{i\mu}^{(N)} = z_{i\mu} - v_{i\mu}^{(k)} \ ; \tag{3.1.1}$$

Oblicz sygnały zwrotne $\delta_{i\mu}^{(k)}$, $k = N-1, \ldots, 1$, $i = 1, \ldots, m_k$ dla wszystkich neuronów warstw poprzednich sieci propagując te sygnały kolejno wstecz sieci $\mathcal{P}$ poczynając od warstwy $N-1$ aż do warstwy pierwszej korzystając ze wzoru:

$$\delta_{i\mu}^{(k)} = \sum_{j=1}^{m_{k+1}} w_{ji}^{(k+1)} \delta_{j\mu}^{(k+1)} \ ; \tag{3.1.2}$$

Korzystając z wyznaczonych wzorami (3.1.1) oraz (3.1.2) i zapamiętanych w poprzednich krokach algorytmu wartości wyjść i sygnałów zwrotnych $v_{i\mu}^{(k)}$, $k = 0, \ldots, N$, $i = 0, \ldots, m_k$ $\delta_{i\mu}^{(k)}$, $k = 1, \ldots, N$, $i = 1, \ldots, m_k$ dla neuronów sieci $\mathcal{P}$ uaktualnij każdą z wag sieci $w_{pq}^{(l)}$, $l = 1, \ldots, N-1$, $p = 1, \ldots, m_l$, $q = 1, \ldots, m_{l-1}$ według następującego wzoru:

$$w_{pq}^{(l)} = w_{pq}^{(l)} + \eta \ \delta_{p\mu}^{(l)} \ v_{q\mu_0}^{(l-1)} \ ; \tag{3.1.3}$$

dopóki *Lista wzorców treningowych $\mathcal{L}$ jest pusta ;*

dopóki *Kryterium stopu $\mathcal{S}$ jest spełnione ;*

Zgodnie z przyjętą symboliką, niech N oznacza ilość warstw sieci, dodatkowo oznaczmy przez $\mathcal{K}$ całkowitą liczbę neuronów[1] sieci, zaś przez $\mathcal{W}$ całkowitą ilość wag[2] liniowego perceptronu $\mathcal{P}$.

Nietrudno zauważyć, że ilości rzeczywistych operacji prostych i mnożeń dla pierwszego zasadniczego punktu pojedynczego kroku iteracyjnego algorytmu 3.1 są równe: $\sum_{k=1}^{N} \sum_{i=1}^{m_k} (dim (\mathbf{w}_i^{(k)}) - 1) = \mathcal{W} - \mathcal{K}$ i $\sum_{k=1}^{N} \sum_{i=1}^{m_k} dim (\mathbf{w}_i^{(k)}) = \mathcal{W}$, gdzie pojedynczy symbol $dim (\mathbf{w}_i^{(k)})$, $k = 1, \ldots, N$, $i = 1, \ldots, m_k$ oznacza wymiar i–tego wektora wagowego k–tej warstwy rozważanej sieci. Ze wzorów (3.1.1) i (3.1.2) wynika natychmiast, że ilość rzeczywistych operacji prostych potrzebnych do realizacji drugiego ze wspomnianych wyżej etapów pojedynczego kroku optymalizacyjnego metody 3.1 wynosi $m_N + \sum_{k=2}^{K} ([\sum_{i=1}^{m_k} dim (\mathbf{w}_i^{(k)})] - m_{k-1}) = \mathcal{W} - \mathcal{W}_f - \mathcal{K} + 2\,\mathcal{K}_l$, gdzie $\mathcal{W}_f$ jest całkowitą ilością wag w pierwszej warstwie sieci podlegającej adaptacji, zaś $\mathcal{K}_l$ oznacza liczbę neuronów ostatniej warstwy sieci. Ilość rzeczywistych mnożeń drugiego zasadniczego punktu rozważanego algoryrmu 3.1 jest natomiast równa wartości $\sum_{k=2}^{N} \sum_{i=1}^{m_k} dim (\mathbf{w}_i^{(k)}) = \mathcal{W} - \mathcal{W}_f$. Wreszcie wzór (3.1.3) dla trzeciego etapu implementacyjnego metody wstecznej propagacji błędu wymaga wykonania $\sum_{k=1}^{N} \sum_{i=1}^{m_k} dim (\mathbf{w}_i^{(k)}) = \mathcal{W}$ rzeczywistych dodawań oraz odpowiednio $2 \sum_{k=1}^{N} \sum_{i=1}^{m_k} dim (\mathbf{w}_i^{(k)}) = 2\,\mathcal{W}$ rzeczywistych mnożeń.

Oznaczmy symbolami $\overrightarrow{\mathcal{P}}_{\mathcal{W}}^{+}/\overrightarrow{\mathcal{P}}_{\mathcal{W}}^{*}$, $\overleftrightarrow{\mathcal{P}}_{\mathcal{W}}^{+}/\overleftrightarrow{\mathcal{P}}_{\mathcal{W}}^{*}$ liczbę rzeczywistych operacji prostych oraz mnożeń odpowiednio: niezbędnych dla propagacji prostej sygnału od wejścia do wyjścia rozważanej sieci (co odpowiada pierwszemu zasadniczemu punktowi metody 3.1) oraz koniecznych dla przeprowadzenia pojedynczego, pełnego kroku optymalizacyjnego algorytmu wstecznej propagacji błędu. Wtedy liczby te dla *perceptoronu liniowego o dowolnej topologii* równe są wartościom:

$$\overrightarrow{\mathcal{P}}_{\mathcal{W}}^{+} = \mathcal{W} - \mathcal{K}\ , \quad \overrightarrow{\mathcal{P}}_{\mathcal{W}}^{*} = \mathcal{W} \tag{3.11a}$$

$$\overleftrightarrow{\mathcal{P}}_{\mathcal{W}}^{+} = 3\,\mathcal{W} - \mathcal{W}_f - 2\,\mathcal{K} + 2\,\mathcal{K}_l\ , \quad \overleftrightarrow{\mathcal{P}}_{\mathcal{W}}^{*} = 4\,\mathcal{W} - \mathcal{W}_f \tag{3.11b}$$

Rozważmy jedynie złożoności obliczeniowe i pamięciowe pełnego kroku optymalizacyjnego procedury 3.1, gdyż propagacja prosta sygnału w sieci (oznaczana dalej skrótowo symbolem `FP`) stanowi jego część składową. A zatem całkowita złożoność obliczeniowa pojedynczego kroku optymalizacyjnego metody wstecznej propagacji błędu jest liniowa względem ilości wag wybranej sieci.

Zastanówmy się jeszcze nad złożonością pamięciową procesów wyznaczania odpowiedzi i nauczania liniowego perceptronu wielowarstwowego metodą 3.1. Jak wynika z postaci modelu 3.2 do poprawnej symulacji procesu wyznaczania odpowiedzi wyjściowej sieci dla pojedynczego sygnału pobudzającego niezbędne jest zapamiętanie wszystkich $\mathcal{W}$, chwilowych lub permanentnych, wartości wagowych każdego z jej neuronów. Konstrukcja algorytmu 3.1 z kolei implikuje konieczność ciągłego przechowywania w pamięci systemu komputerowego wartości sygnałów wyjściowych wszystkich $\mathcal{K}$ jednostek omawianej sieci. Ponadto, wymaga także dalszej rezerwacji zasobów systemowych w liczbie jednej dodatkowej zmiennej/rejestru rzeczywistego na neuron w celu przechowywania sygnałów zwrotnych każdej z jednostek rozpatrywanej sieci dla potrzeb prawidłowego przebiegu procedury korekty jej wag przy

[1] nie licząc neuronów warstwy sieci o indeksie zero, tzn. warstwy kopiującej sieci
[2] podobnie, nie biorąc pod uwagę wag warstwy kopiującej sieci

użyciu wzoru (3.1.3). Oznacza to, że całkowite zapotrzebowanie pamięciowe pozwalające na implementację programową lub/i sprzętową perceptronu liniowego wraz z techniką jego treningu metodą wstecznej propagacji błędu wynosi $\mathcal{W} + 2\,\mathcal{K}$. A zatem złożoność ta jest również liniowa względem liczby wag wybranej sieci.

3.3. Szybka sieć neuronowa do kompresji obrazów

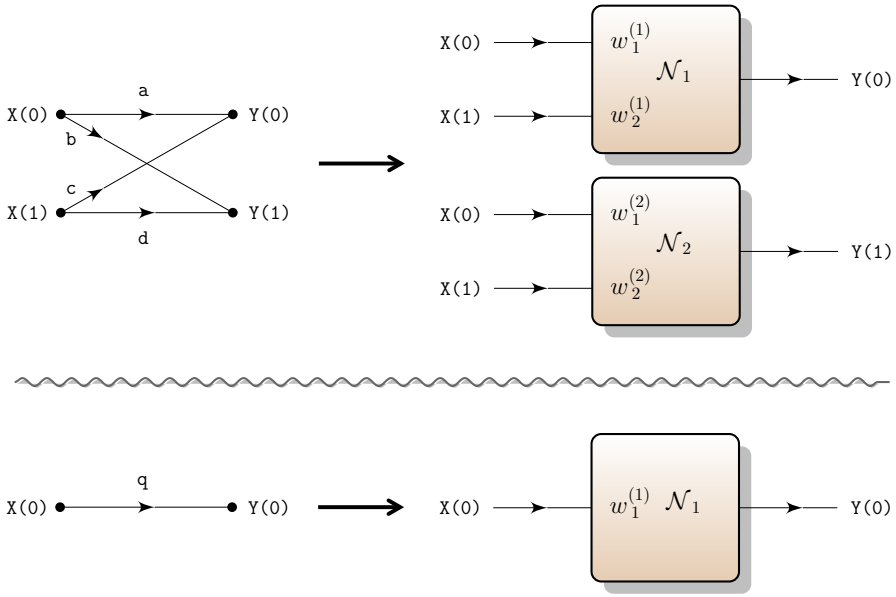
W niniejszym rozdziale przedstawiona będzie koncepcja i architektura sieci neuronowych realizujących szybkie zunifikowane przekształcenia dyskretne dedykowane problemowi kompresji obrazów. Zaproponowana w ten sposób metoda kompresji obrazów zostanie następnie skonfrontowana z innymi technikami adaptacyjnymi i suboptymalnymi stosowanymi obecnie w praktyce względem kryteriów teoretycznych efektywności obliczeniowej i złożoności pamięciowej. Przejdźmy zatem do przedstawienia architektury szybkiej sieci neuronowej do kompresji obrazów.

3.3.1. Architektura szybkiej sieci neuronowej do kompresji obrazów

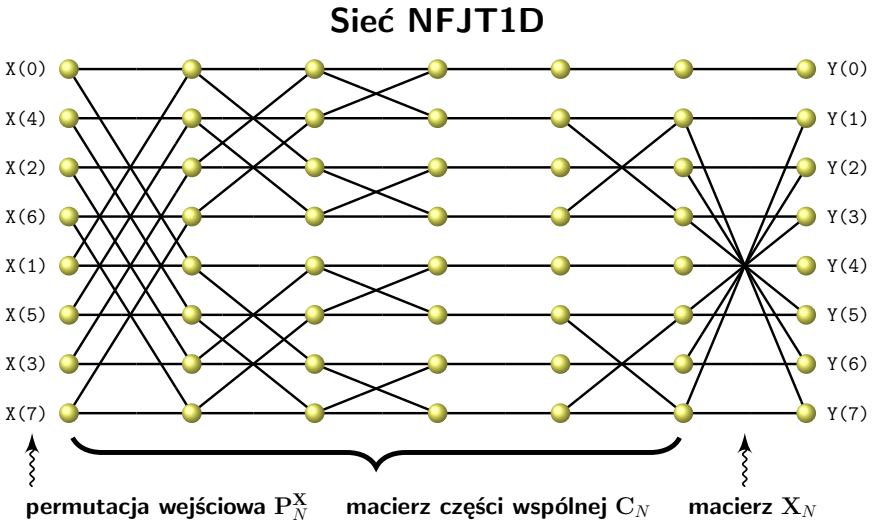
Prezentowana architektura sieci neuronowej do kompresji obrazów opiera się bezpośrednio na przedstawionych wcześniej szybkich algorytmach zunifikowanych wyznaczania jedno/dwuwymiarowych przekształceń dyskretnych z wykorzystaniem liniowych sieci neuronowych typu perceptron wielowarstwowy. Z rysunków przedstawiających grafy przepływu danych jednowymiarowych algorytmów zunifikowanych wynika, że można w naturalny i prosty sposób dokonać konwersji ich postaci do odpowiedniej struktury liniowej sieci neuronowej. Struktura taka byłaby strukturą warstwową i składałaby się z liniowych neuronów, z których każda para w ramach danej warstwy realizowałaby pojedynczą operację motylkową. Utworzone w ten sposób jednostki obliczeniowe byłyby neuronami liniowymi o jednym lub dwu wejściach oraz wyjściu połączonym odpowiednio z jednym lub dwoma neuronami kolejnej warstwy sieci. Ten prosty i naturalny zarazem sposób konwersji przedstawiony jest w formie graficznej na rys. 3.4. Przykładowy schemat 8–punktowej sieci neuronowej, zwanej dalej skrótowo siecią `NFJT1D`, powstałej w wyniku tak przeprowadzonej konwersji postaci grafów przepływu danych dla przypadku szybkich algorytmów zunifikowanych realizujących przekształcenia jednowymiarowe przedstawia rysunek 3.5. Na schemacie tym zaznaczona jest przykładowa permutacja wejściowa, odpowiadająca przemieszaniu elementów wektorów wejściowych właściwemu dla algorytmu zunifikowanego wyznaczania dyskretnej transformaty kosinusowej oraz zaznaczone są też obszary warstw odpowiadające macierzom ogólnego schematu obliczeniowego dla jednowymiarowych algorytmów zunifikowanych, pokazanego na rys. 2.14.

Warto zauważyć, że choć zawsze można optymalizować sieć jednowymiarową przedstawioną na rysunku 3.5 w całości, to zgodnie z wnioskami dotyczącymi ogólnego schematu obliczeniowego dla jednowymiarowych algorytmów zunifikowanych, aby zrealizować wszystkie cztery przykładowe transformaty jednowymiarowe za jej pomocą wystarczy poddać optymalizacji jedynie ostatnią warstwę tej sieci, odpowiadającą macierzy $\mathbf{X}_N$ wspomnianego schematu obliczeniowego. Oznacza to,

że w przypadku N–punktowych jednowymiarowych przekształceń liniowych adaptacja wag sieci do wybranego zadania filtracji może dotyczyć jedynie $2\,(N-1)$ jej połączeń wagowych, zawartych w ostatniej warstwie sieci, co znacznie redukuje oryginalny rozmiar zadania optymalizacyjnego, niezależnie od wyboru metody treningowej jaką jest ono przeprowadzane. Pojedyncza sieć `NFJT1D` jest zgodnie z rozważaniami podjętymi w poprzednim podrozdziale podstawą konstrukcji architektury neuronowej przeznaczonej do przetwarzania sygnałów dwuwymiarowych jakimi modelowane są obrazy. Przejdźmy zatem do analizy tejże architektury.



Rys. 3.4: Konwersja operacji bazowych zunifikowanych algorytmów szybkich do postaci neuronowej



Rys. 3.5: Przykładowa sieć neuronowa realizująca 8–punktowe przekształcenie jednowymiarowe

Biorąc pod uwagę wnioski płynące z treści wcześniejszych rozważań, w przypadku analizowanego w niniejszym opracowaniu problemu efektywnej adaptacyjnej kompresji obrazów naturalnych należy, jak stwierdzono przed chwilą, wziąć pod uwagę dwuwymiarowy charakter sygnałów je reprezentujących i w konsekwencji podjąć próbę konstrukcji architektury sieci neuronowej w oparciu o szybkie algorytmy zunifikowane realizujące dwuwymiarowe przekształcenia dyskretne.

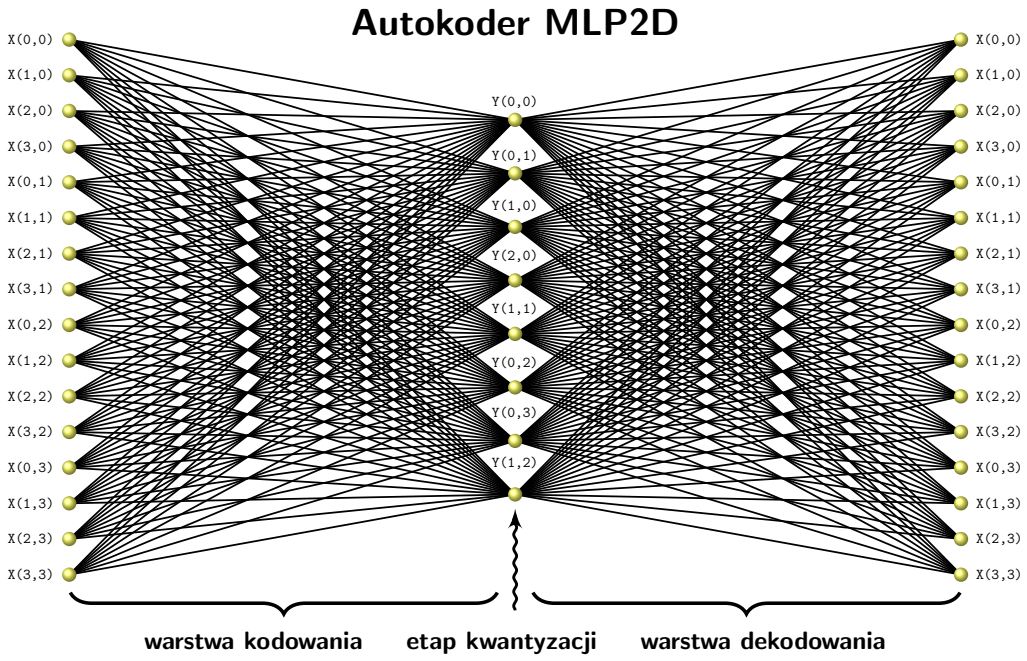
W tym celu, aby dokonać właściwego wyboru architektury sieci neuronowej do kompresji obrazów, warto skorzystać ze znanych i sprawdzonych podejść do tak sformułowanego problemu optymalizacyjnego. Jednym z takich właśnie najczęściej stosowanych i opisywanych w literaturze schematów konstrukcyjnych sieci neuronowych do kompresji obrazów jest tak zwany schemat *autokodera* [29, 63, 64, 95]. Schemat ten wykorzystuje standardowy perceptron liniowy[1] o trzech warstwach przetwarzających, pokazany na rysunku 3.6a, z których to warstwa ukryta, zwana zwyczajowo warstwą kwantyzacji, ma istotnie mniej neuronów od pozostałych dwóch warstw – wejściowej i wyjściowej o identycznych liczbach neuronów. To właśnie warstwa kwantyzacji wymusza kompresję sygnału poprzez realizację efektywnego, z jakościowego punktu widzenia, kodowania elementów macierzy wejściowych na zredukowanej liczbie współczynników przekształcenia prostego. W takim schemacie, przy domyślnym założeniu, że pojedyncze elementy macierzy wejściowych oraz współczynniki przekształcenia kodującego zapamiętywane są na identycznej liczbie bitów, zadany poziom kompresji sygnału dwuwymiarowego definiowany jest z reguły jako procent wyzerowanych współczynników transformaty kodującej według wzoru:

$$R \triangleq \left(1 - \frac{m}{M}\right) \cdot 100\ [\,\%\,] \tag{3.12}$$

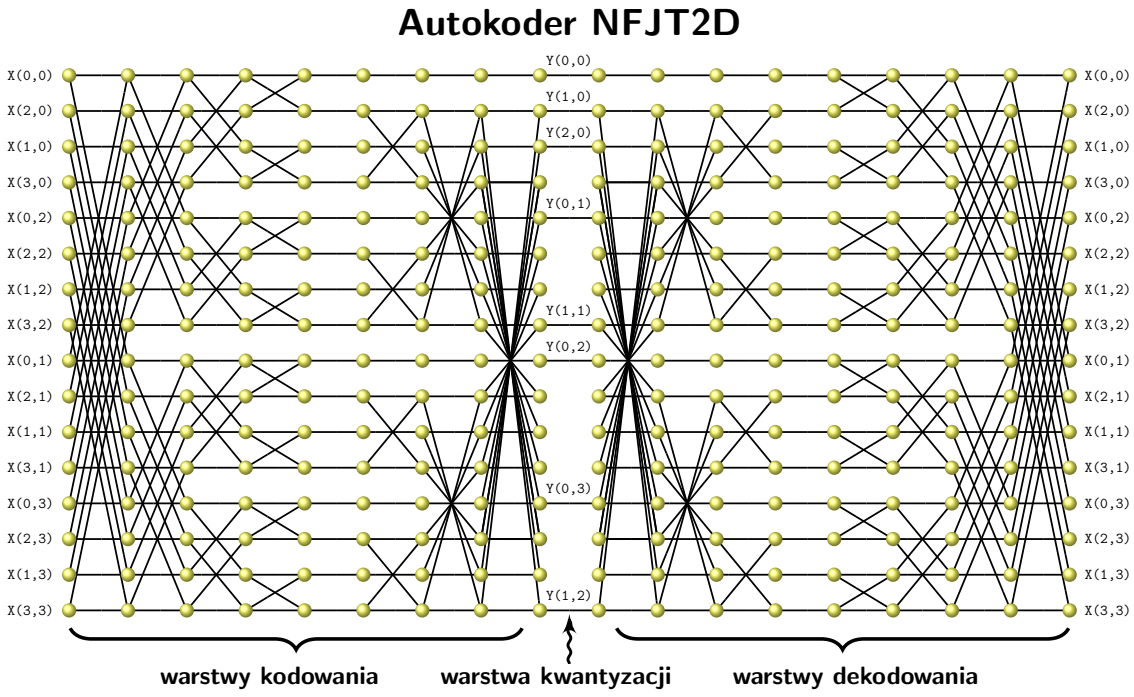
gdzie $M = N^2$ jest całkowitym rozmiarem kompresowanego $N \times N$–elementowego sygnału dwuwymiarowego, zaś m jest liczbą neuronów w warstwie kwantyzacji.

Zadaniem rozważanej sieci jest praca na zasadzie autoasocjacyjnej, innymi słowy głównym celem jej treningu jest ustawienie wag warstw ukrytej i wyjściowej sieci w taki sposób, aby sieć odtwarzała jak najwierniej, względem wybranej miary błędu, sygnały wejściowe na swoich wyjściach, pomimo zredukowanej w stosunku do wymiarów wektorów wejściowych/wyjściowych liczebności neuronów w warstwie ukrytej. Sieć taka jest następnie trenowana najczęściej standardową metodą wstecznej propagacji błędu (z błędem średniokwadratowym jako kryterium minimalizacyjnym) na przykładowych sygnałach dwuwymiarowych reprezentujących fragmenty obrazów zadanej klasy obrazów naturalnych, przeznaczonych do przyszłej kompresji. Znanym powszechnie jest fakt [28, 29, 64, 94, 95], iż sieć taka może być interpretowana jako neuronowa implementacja jedno lub/i dwuwymiarowego zadania ekstrakcji gównych składowych sygnału wejściowego (ang. PCA - *principal component analysis*). Zakładając dostatecznie długi okres procesu treningowego sieci i przyjmując, że został on zakończony sukcesem, autokoder powinien, z teoretycznego punktu widzenia, realizować po zakończeniu procesu nauki sieci estymatory przekształceń prostego (warstwa ukryta) i odwrotnego (warstwa wyjściowa) Karhunena-Loève'go sygnału wejściowego, a ściśle rzecz ujmując powinien realizować, do czego jest teoretycznie

[1] tzn. sieć `MLP`, ang. *multilayer perceptron*



Rys. 3.6a: Autkododer `MLP2D` wykorzystujący standardowy wielowarstwowy perceptron liniowy



Rys. 3.6b: Autkododer `NFJT2D` wykorzystujący dwuwymiarowe szybkie algorytmy zunifikowane

zdolny, dowolne przekształcenia kodujące i dekodujące dokonujące rzutu, i operacji do niego odwrotnej, wejściowych wektorów treningowych na przestrzeń pierwszych m wektorów własnych standardowego estymatora macierzy autokowariancji zbioru wektorów uczących, gdzie jak wspomniano wcześniej, m jest liczbą neuronów warstwy ukrytej autokodera. Zgodnie z przedstawionymi wcześniej rozważaniami dotyczącymi problemu kompresji obrazów wymienione przed chwilą przekształcenia są przekształceniami optymalnymi dla sformułowanego w taki sposób zadania kompresji obrazów naturalnych.

Realizacja szybkich sieci neuronowych do kompresji obrazów opiera się właśnie na pomyśle zastosowania przedstawionego tu ogólnego mechanizmu autokodera do adaptacyjnego doboru przekształceń roboczych procesu kompresji obrazu z jednoczesnym zastąpieniem standardowej architektury perceptronowej, charakteryzującej się pełnym schematem połączeń pomiędzy poszczególnymi warstwami sieci `MLP`, strukturą szybką, opartą o postaci grafów przepływu danych zunifikowanych algorytmów wyznaczania dwuwymiarowych przekształceń dyskretnych, przedstawionych w poprzednim podrozdziale. Poglądowe porównanie obydwu typów omawianych tu architektur, zwanych dalej umownie sieciami `MLP2D` oraz `NFJT2D`,[1] dla przykładowych 4×4–punktowych dwuwymiarowych przekształceń liniowych i współczynniku kompresji R równym w obydwu przypadkach 50% znajduje się na rysunkach 3.6a oraz 3.6b.

Warto zwrócić uwagę na fakt, że o ile w przypadku sieci `MLP2D` rozmieszczenie elementów macierzy wejściowych (wyjściowych) sygnału dwuwymiarowego na wejściach (wyjściach) rozważanej sieci, jak również ich kolejność w warstwie kwantyzacji jest wyłącznie umowna i w praktyce każde przemieszanie jest tutaj możliwe, o tyle dla sieci `NFJT2D` permutacje tych elementów nie mogą być już dowolne i muszą być odpowiednio dobrane. Wynika to z faktu, iż struktura sieci `NFJT2D` jako neuronowej implementacji algorytmu szybkiego o zredukowanej liczbie współczynników macierzy składowych zakładającego dodatkowo separowalność reprezentowanych przezeń przekształceń dwuwymiarowych, niewątpliwie nie jest w stanie realizować przekształceń dyskretnych o dowolnej postaci z dowolnie dobranymi permutacjami elementów macierzy wejściowych/wyjściowych. Należy zatem, jak wspomniano przed chwilą, zastanowić się nad wyborem odpowiedniej permutacji w warstwach skrajnych sieci `NFJT2D`, a także równie istotnym doborem przemieszania elementów macierzy współczynników przekształcenia dwuwymiarowego w warstwie kwantyzacji wiążącym się ściśle z problemem wyboru wyjść tejże warstwy, których sygnał nie jest propagowany w kierunku warstwy dekodującej. Dobór wspomnianych permutacji i współczynników z powodu natury zastosowanych w niniejszej pracy technik treningowych sieci musi zostać dokonany i założony *a-priori*. Wymienione problemy mogą być rozwiązane na wiele różnych sposobów, z których wybrane podejścia prezentowane są szczegółowo w dalszej części pracy przy okazji omówienia metod treningowych i testów eksperymentalnych kompresji dla proponowanych sieci.

W przypadku schematu z rysunku 3.6b przykładowej sieci 4×4–punktowej pokazano jeden z zastosowanych podczas późniejszych eksperymentów i, wydawałoby się, najbardziej intuicyjnych z punktu widzenia konstrukcji dwuwymiarowych algo-

[1] ang. `MLP2D - 2D` *multilayer perceptron* oraz `NFJT2D - 2D` *neural fast Jacymirski transform*

rytmów szybkich oraz wnisoków dotyczących optymalności transformaty DCT w problemie kompresji obrazów naturalnych, sposobów doboru rozważanych permutacji. A mianowicie przyjęto w tym przypadku permutację właściwą dla zunifikowanego algorytmu szybkiego JFCT2D wyznaczania dyskretnego skalowanego przekształcenia kosinusowego jako transformaty bliskiej przekształceniu potencjalnie optymalnemu. Dla doboru pozostawionych w warstwie kwantyzacji współczynników przekształcenia kodującego skorzystano zaś z permutacji ZIG-ZAG (patrz [4] oraz [104]) jako tej, która również może okazać się potencjalnie bliska doborowi optymalnemu dla wybranej klasy obrazów. W trakcie eksperymentów z adaptacyjną kompresją obrazów za pomocą zaproponowanej architektury do optymalizacji sieci zastosowano przedstawioną wcześniej metodę wstecznej propagacji błędu. Dzięki jawnemu wyróżnieniu warstwy kwantyzacji pomiędzy przekształceniami kodującym i dekodującym sieci NFJT2D wspomniana metoda nauki może zostać zastosowana bez żadnych dodatkowych, niestandardowych modyfikacji jej podstawowej postaci. Wystarczy na początku procesu treningowego sieci przyjąć zerowe wartości wag w neuronach ostatniej warstwy przekształcenia kodującego i warstwy pierwszej przekształcenia odtwarzającego, reprezentujących usuwane w procesie kompresji współczynniki transformaty kodującej, a sam sposób działania metody wstecznej propagacji błędu zapewnia niezmienność zerowych wartości tych wag w trakcie całego procesu treningu sieci aż do jego zakończenia.

Na koniec warto dodać, że trening sieci zarówno tej standardowej jak i proponowanej, w schemacie autokodera posiada jeszcze jedną zaletę, polegającą na tym, że po zakończeniu procesu nauki użytkownik dysponuje od razu przekształceniami kodującym jak i odtwarzającym zakodowany sygnał dwuwymiarowy bez konieczności dokonywania dodatkowych operacji obliczeniowych.

Po omówieniu zaproponowanej architektury sieci neuronowej do kompresji obrazów warto rozpatrzyć zagadnienie analizy efektywności obliczeniowej i pamięciowej zaproponowanej architektury sieciowej.

3.3.2. Analiza efektywności szybkiej sieci neuronowej do kompresji obrazów

W niniejszym podrozdziale przedstawiona zostanie analiza porównawcza teoretycznych efektywności obliczeniowch procesów treningu i późniejszego wyznaczania odpowiedzi nauczonych sieci dla standardowego autokodera MLP2D oraz proponowanej architektury szybkiej NFJT2D. W przypadku operacji wyznaczania odpowiedzi nauczonych sieci rozważane charakterystyki efektywnościowe dotyczyć będą jedynie części analizowanych sieci odpowiedzialnych za dekodowanie skompresowanego sygnału jako operacji najbardziej interesującej z punktu widzenia przeciętnego użytkownika systemu kompresji. Analizie porównawczej poddane zostaną również charakterystyki pojemnościowe sieci MLP2D i NFJT2D. Przejdźmy zatem do wyznaczenia wspomnianych charakterystyk. Przyjmijmy, że mamy do czynienia z $N \times N$ punktowym adaptacyjnym przekształceniem dyskretnym. Dla uproszczenia zapisu zdefiniujmy parametr $\mu \stackrel{\Delta}{=} m/N^2$, gdzie m oznacza liczbę współczynników rozważanego przekształcenia mniejszą lub równą N^2, które biorą udział w kodowaniu

dwuwymiarowego sygnału wejściowego. Charakterystyki będące przedmiotem zainteresowania wyznaczone będą na podstawie podanych w rozdziale trzecim ogólnych wzorów złożonościowych (3.11) dla warstwowych sieci liniowych o dowolnej topologii, dlatego też warto zebrać w tabeli wszystkie niezbędne dane, służące obliczeniu badanych złożoności. Poniższa tabela zawiera podstawowe charakterystyki ilościowe będące parametrami wspomnianych wzorów złożonościowych (3.11).

Tab. 3.1: Charakterystyki ilościowe dla sieci `MLP2D` oraz `NFJT2D`

Typ sieci	*Charakterystyki ilościowe części dekodującej sieci*			
	$\mathcal{W}$	$\mathcal{W}_f$	$\mathcal{K}$	$\mathcal{K}_l$
`MLP2D`	$\mu\, N^4$	—	N^2	—
`NFJT2D`	$N^2(\,4\,log_2 N^2 - 1\,) + 2$	—	$N^2(\,2\,log_2 N^2 + 1\,)$	—
Typ sieci	*Charakterystyki ilościowe dla schematu pełnego sieci*			
	$\mathcal{W}$	$\mathcal{W}_f$	$\mathcal{K}$	$\mathcal{K}_l$
`MLP2D`	$2\,\mu\, N^4$	$\mu\, N^4$	$(\,\mu + 1\,)N^2$	N^2
`NFJT2D`	$N^2(\,8\,log_2 N^2 - 3\,) + 4$	$2\,N^2$	$N^2(\,4\,log_2 N^2 + 1\,)$	N^2

gdzie, zgodnie z oznaczeniami dotyczącymi wzorów (3.11), $\mathcal{W}$ oraz $\mathcal{K}$ są odpowiednio całkowitą liczbą wag i neuronów wybranej części/całości sieci, z pominięciem wag i neuronów ich warstw kopiujących. W przypadku charakterystyk pełnego schematu rozważanych sieci $\mathcal{W}_f$ jest licznością wag w warstwach obu sieci o indeksach równych jeden, $\mathcal{K}_l$ zaś oznacza liczbę neuronów ostatnich warstw każdej z nich.

Tab. 3.2: Charakterystyki efektywnościwo-pojemnościowe dla sieci `MLP2D` i `NFJT2D`

Typ sieci	*Ilość rzeczywistych operacji arytmetycznych*		*Złożoność pamięciowa*
	operacje proste	*mnożenia*	
`MLP2D-FP`	$\mu\, N^4 - N^2$	$\mu\, N^4$	$2\,[\,\mu\, N^4 + (\,\mu + 1\,)N^2]$
`MLP2D-BP`	$\mu\,(\,5\, N^4 - 2\, N^2)$	$7\,\mu\, N^4$	
`NFJT2D-FP`	$2N^2(log_2 N^2 - 1) + 2$	$N^2(4\, log_2 N^2 - 1) + 2$	$N^2(16\, log_2 N^2 - 1) + 4$
`NFJT2D-BP`	$N^2(16\, log_2 N^2 - 11) + 12$	$2N^2(16\, log_2 N^2 - 7) + 16$	

W tabeli 3.2 przedstawiono charakterystyki efektywnościwo - pojemnościowe dla sieci `MLP2D` oraz `NFJT2D`. Zostały one uzyskane po krótkich obliczeniach poprzez podstawienie wartości odpowiednich charakterystyk ilościowych dla rozważanych sieci, pokazanych w tabeli 3.1, do wzorów złożonościowych (3.11) z jednoczesnym uwzględnieniem podanej wyżej definicji parametru μ. Charakterystyki opatrzone skrótowo symbolem `FP` dotyczą dokładnych liczb rzeczywistych operacji prostych oraz mnożeń

niezbędnych do wyznaczenia odpowiedzi obydwu nauczonych sieci w procesie dekodowania skompresowanego sygnału wejściowego, zaś skrót BP odnosi się do liczb wymienionych wyżej operacji arytmetycznych koniecznych do przeprowadzenia pełnego, pojedynczego kroku optymalizacji dla obu rozważanych autokoderów metodą wstecznej propagacji błędu. Złożoności pamięciowe zostały obliczone na podstawie zależności ogólnej ($\mathcal{W}+2\mathcal{K}$) wyznaczonej pod koniec rozdziału trzeciego i odnoszą się do całkowitego zapotrzebowania pamięciowego implementacji autokoderów MLP2D i NFJT2D trenowanych metodą wstecznej propagacji błędu.

Na podstawie wartości zgromadzonych w tabeli 3.2 można wnioskować bezpośrednio, że zarówno złożoność obliczeniowa jak i pamięciowa dla wszystkich trybów działania, tzn. treningu i późniejszego etapu operacyjnego nauczonej sieci, proponowanej architektury neuronowej do kompresji obrazów NFJT2D wynosi dla dwuwymiarowych $N \times N$–punktowych adaptacyjnych przekształceń dyskretnych $\mathcal{O}(N^2 log_2 N^2)$. Oznacza to, na podstawie odpowiednich wartości z tabeli 3.2, *redukcję o rząd wielkości* z poziomu $\mathcal{O}(N^4)$ każdego z wymienionych wyżej parametrów efektywnościowych proponowanej architektury neuronowej względem odpowiadającyh im charakterystyk właściwych dla standardowej sieci neuronowej MLP2D używanej zwykle w adaptacyjnej kompresji obrazów. Ponadto biorąc pod uwagę separowalność wszystkich rozważanych w niniejszej pracy przekształceń dyskretnych dwuwymiarowych i porównując złożoności obliczeniowe standardowych algorytmów wyznaczania przykładowych transformat jednowymiarowych FCT, FST, FHT oraz FFT (tabela 2.1) z odpowiadającymi im złożonościami wyznaczania tych przekształceń za pomocą proponowanych zunifikowanych procedur szybkich (wzór (2.41)) można dojść do wniosku, że ostatnie z wymienionych metod nieznacznie ustępują pod względem złożonościowym standardowym szybkim algorytmom dedykowanym specjalnie do obliczania wymienionych wyżej przekształceń dyskretnych.

Na koniec warto zauważyć, że z przedstawionych w niniejszym rozdziale postaci zunifikowanych algorytmów szybkich jedno i dwuwymiarowych wynika, że w przypadku autokodera NFJT2D w procesie adaptacji sieci do realizacji zadania kompresji można pominąć warstwy reprezentujące część wspólną szybkich dwuwymiarowych algorytmów zunifikowanych bazującą na przekształceniu kosinusowym. Zakładając indeksację warstw autokodera NFJT2D od wartości zero, oznacza to tyle, że w praktyce przy adaptacji sieci do realizacji dwuwymiarowej transformaty $N \times N$ punktowej można nauczać jedynie warstw autokodera o indeksach $3\,log_2 N$ i $4\,log_2 N$ w jego części kodującej oraz, symetrycznie, warstw o indeksach $4\,log_2 N+2$ i $5\,log_2 N+2$ w części dekodującej sieci, ustawiając jednocześnie pozostałe wagi sieci zgodnie z wartościami współczynników dwupunktowych obrotów ortogonalnych, właściwych dla bazowego dwuwymiarowego zunifikowanego prostego/odwrotnego przekształcenia kosinusowego, wynikających ze wzoru rozkładu (2.46). Powoduje to konieczność adaptacji neuronów sieci NFJT2D w liczbie $4\,(\,2\,N^2 - N - 1\,)$ redukując tym samym o kolejny rząd wielkości złożoności obliczeniową i pamięciową procesu treningu sieci do poziomu liniowego względem całkowitego rozmiaru optymalizowanego przekształcenia. Podobny wariant nauki sieci NFJT2D przyjęty został podczas ich testów jakościowych opisanych dokładnie w następnym rozdziale.

4. WYNIKI BADAŃ W PROBLEMIE KOMPRESJI OBRAZÓW

4.1. Wprowadzenie

W niniejszym rozdziale przedstawione zostaną wyniki badań eksperymentalnych dla zaprezentowanych w niniejszym opracowaniu szybkich sieci neuronowych do kompresji obrazów. Najpierw jednak przedstawiony zostanie probabilistyczny model obrazu naturalnego pozwalający w systematyczny sposób zweryfikować eksperymentalnie prawidłowość teoretycznych przewidywań dotyczących jakości procesu optymalizacji zaprezentowanych sieci neuronowych.

4.2. Probabilistyczny model obrazu

Modelowanie obrazów jest zagadnieniem ważnym, z teoretycznego punktu widzenia jednocześnie ciekawym i dość skomplikowanym. Istnieje wiele sposobów podejścia do problemu modelowania obrazów, takich jak modelowanie statystyczne, semantyczne, hybrydowe i inne [48]. Szczegółowe sposoby modelowania obrazów zależą przede wszystkim od przewidywanego zastosowania budowanego modelu w przyszłych zadaniach związanych z szeroko pojętym zagadnieniem przetwarzania obrazów. Zadaniem teoretycznego modelu obrazu jest dostarczenie narzędzi teoretycznych analizy obrazów, umożliwiających w perspektywie konstrukcję efektywnych algorytmów ich przetwarzania. W przypadku problemów kompresji, syntezy, rekonstrukcji czy rozpoznawania obrazów [96, 105–107], bardzo przydatne okazały się modele probabilistyczne, których założenia koncepcyjne pasują dobrze pod wieloma względami do charakteru wymienionych zadań. Bardzo przydatną klasą modeli probabilistycznych obrazów są pola losowe, a wśród nich szczególnie istotnymi z punktu widzenia zadania kompresji obrazów okazały się pola Gaussa–Markowa.

W niniejszym podrozdziale przedstawione będą odpowiednie definicje, sformułowany zostanie model probabilistyczny obrazu i zaprezentowane będą wybrane charakterystyki probabilistyczne wspomnianego modelu, najistotniejsze z punktu widzenia praktyki zadań kompresji i syntezy obrazów. Wymienione pojęcia są zgodne w sensie teoretycznym i zaczerpnięte są z prac [14, 45, 47, 49, 108].

4.2.1. Pole losowe Gaussa-Markowa jako model obrazu

W celu uniknięcia nieścisłości przedstawmy szereg podstawowych definicji i własności niezbędnych do pełnego i jednoznacznego sformułowania modelu obrazu naturalnego, który będzie wykorzystywany dalej w badaniach eksperymentalnych.

Definicja 4.1.

Rodzinę rzeczywistych zmiennych losowych $\{ X_{(m,n)};\ (m,n) \in \mathbb{Z}^2 \}$ zdefiniowanych na tej samej przestrzeni probabilistycznej określa się mianem *pola losowego* i oznacza skrótowo symbolem X.

Definicja 4.2.

Niech X będzie zadanym polem losowym. Rozważane pole określamy mianem *pola losowego o zerowej wartości oczekiwanej* jeśli dla dowolnego $(m,n) \in \mathbb{Z}^2$ mamy:

$$E\{X_{m,n}\} = 0 \tag{4.1}$$

Definicja 4.3.

Niech X będzie zadanym polem losowym o zerowej wartości oczekiwanej. Rozważane pole określamy mianem *stacjonarnego pola losowego* o zerowej wartości oczekiwanej jeśli dla dowolnych punktów (m_1,n_1), (m_2,n_2), (p_1,q_1), $(p_2,q_2) \in \mathbb{Z}^2$ zachodzi:

$$(m_1-m_2)=(p_1-p_2) \wedge (n_1-n_2)=(q_1-q_2) \Rightarrow E\{X_{m_1,n_1}X_{m_2,n_2}\} = E\{X_{p_1,q_1}X_{p_2,q_2}\} \tag{4.2}$$

Definicja 4.4.

Niech X będzie stacjonarnym polem losowym o zerowej wartości oczekiwanej. Rozważane pole określamy mianem *separowalnego* jeśli dla dowolnych punktów (m_1,n_1), $(m_2,n_2) \in \mathbb{Z}^2$ spełniony jest warunek:

$$\rho\{X_{m_1,n_1}X_{m_2,n_2}\} = \rho\{X_{m_1,n_1}X_{m_2,n_1}\} \cdot \rho\{X_{m_2,n_1}X_{m_2,n_2}\} \tag{4.3}$$

Separowalność stacjonarnego pola losowego oznacza, że korelacja dowolnie wybranej pary zmiennych losowych tego pola jest iloczynem korelacji zmiennych losowych znajdujących się odpowiednio w tym samym wierszu i tej samej kolumnie rozważanego pola, dla których odległości pionowa i pozioma są równe odpowiadającym im odległościom pionowej i poziomej oryginalnych zmiennych losowych. Podajmy niżej dobrze znaną [17] i istotną własność stacjonarnych, separowalnych pól losowych odnoszącą się do postaci ich macierzy autokowariancji.

Własność 4.1.

Niech X będzie separowalnym, stacjonarnym polem losowym o zerowej wartości oczekiwanej. Oznaczmy przez σ_X^2 wariancję zmiennych losowych pola X, oraz niech $M,\ N \in \mathbb{Z}$ i (m,n), (m_r,n_r), $(m_c,n_c) \in \mathbb{Z}^2$. Zdefiniujmy macierz $\mathbf{X}$ i wektory $\mathbf{x}_r$, $\mathbf{x}_c$ zmiennych losowych rozważanego pola w następujący sposób: $x_{i,j} \stackrel{\Delta}{=} X_{m+i-1,n+j-1}$; $(x_r)_j \stackrel{\Delta}{=} X_{m_r,n_r+j-1}$; $(x_c)_i \stackrel{\Delta}{=} X_{m_c+i-1,n_c}$; $i=1,\ldots,N$, $j=1,\ldots,M$. Wówczas zachodzi:

$$E\left\{vec(\mathbf{X})\, vec(\mathbf{X})^T\right\} = \sigma_X^2\ \rho\{\mathbf{x}_r\,\mathbf{x}_r^T\} \otimes \rho\{\mathbf{x}_c\,\mathbf{x}_c^T\} \tag{4.4}$$

Własność 4.1 orzeka, że macierz autokowariancji separowalnego pola losowego da się wyrazić poprzez produkt Kroneckera macierzy autokorelacji wektorów reprezentujących odpowiednio procesy wierszowy (poziomy) i kolumnowy (pionowy) rozważanego pola. Zaprezentujmy teraz najistotniejsze z punktu widzenia celów niniejszego opracowania definicje.

Definicja 4.5.

Niech X będzie zadanym polem losowym. Rozważane pole losowe nazywamy *polem normalnym* jeśli dla dowolnego $k \in \mathbb{N}$ i dowolnych punktów $(m_1, n_1), \ldots, (m_k, n_k) \in \mathbb{Z}^2$ oraz każdego niezerowego wektora $[\, a_1, \ldots, a_k \,]^T \in \mathbb{R}^k$ zmienna losowa $\sum_{i=1}^{k} a_i \, X_{m_i, n_i}$ jest zmienną losową normalną.

Definicja formalna zmiennej losowej normalnej podana jest chociażby w pracy [109] i jej jawna postać nie jest niezbędna dla dalszych rozważań. Następna definicje dotyczą już pól Gaussa-Markowa.

Definicja 4.6.

Niech X będzie zadanym polem losowym, zaś ζ będzie pewnym stacjonarnym polem losowym o zerowej wartości oczekiwanej. Niech $k \in \mathbb{N}$. Pole X jest *polem Markowa k–tego rzędu w szerokim sensie* jeśli:[1]

$$\exists \, \{\beta_{i,j}\}_{\substack{i,j=0,\ldots,k \\ (i,j)\neq(0,0)}} \subset \mathbb{R} \setminus \{0\} \quad \forall (m,n) \in \mathbb{Z}^2 \quad X_{m,n} = \sum_{\substack{i=0 \\ (i,j)\neq(0,0)}}^{k} \sum_{j=0}^{k} \beta_{i,j} \, X_{m-i,n-j} + \xi_{m,n} \tag{4.5a}$$

$$\forall \, (m_1,n_1), (m_2,n_2) \in \mathbb{Z}^2 \quad E\{\, \xi_{m_1,n_1} \xi_{m_2,n_2} \,\} = \delta_{m_1,m_2} \, \delta_{n_1,n_2} \, \sigma_\xi^2 \tag{4.5b}$$

Definicja 4.7.

Niech X będzie zadanym polem losowym. Pole to określane jest mianem *pola Gaussa Markowa rzędu k* jeśli pole to jest polem normalnym i jednocześnie dla pewnego $k \in \mathbb{N}$ jest polem Markowa w szerokim sensie o rzędzie równym k.

Warto skomentować krótko podane tu definicje. W definicji 4.6 pola losowego Markowa pojawia się określenie *w szerokim sensie*. Określenie to jest zgodnie z terminologią zastosowaną w pracy [47] i uzasadnione o tyle, że wspomniana definicja obejmuje sobą szerszą klasę dwuwymiarowych procesów stochastycznych niż klasycznie rozumiane (w ścisłym sensie) pojęcie pola Markowa, rozważane chociażby w publikacjach [109, 110]. Z drugiej strony dodanie założenia o własności normalności i stacjonarności zadanego losowego pola Markowa w szerokim sensie powoduje, że pole to staje się polem Markowa w zwykłym (ścisłym) sensie. Dowód przytoczonego przed chwilą stwierdzenia znajduje się w artykule [47]. Większość znanych autorowi opracowań, takich jak [16–18, 20, 21, 45–49, 53, 62, 104, 111, 112] w sposób bezpośredni lub domyślny dla potrzeb zagadnienia kompresji określa probabilistyczny model obrazu naturalnego jako stacjonarne, separowalne pole losowe Gaussa-Markowa pierwszego rzędu o zerowej wartości oczekiwanej.

[1] symbol $\delta_{m,n}$ jest oznaczeniem delty Kroneckera

Warto także skomentować przesłanki stojące za takim wyborem matematycznej definicji modelu obrazu naturalnego. Wybór taki motywowany jest z jednej strony prostotą rozważanego modelu i łatwością analizy istotnych z punktu widzenia problemu kompresji obrazu charakterystyk rozważanego modelu. Z drugiej strony opiera się na empirycznej obserwacji faktu, że w większości obrazów spotykanych w rzeczywistości lokalna zmienność jasności pikseli ma w sensie statystycznym podobny charakter w ramach ustalonych, pojedynczych klas obrazów. Innymi słowy dla zadanego typu obrazów rzeczywistych zmienność jasności sąsiednich pikseli obrazu zachowuje się w podobny sposób na całej powierzchni obrazu np. ich jasność jest bardzo zbliżona bądź oscyluje szybko w wybranych kierunkach. To ogólne spostrzeżenie skłania do teoretycznej konstrukcji stochastycznego modelu obrazu, w którym jasność pojedynczego piksela zależy wyłącznie od jasności pikseli w bezpośrednim jego sąsiedztwie, zaś lokalny charakter jej zmian pozostaje stały na całej powierzchni obrazu. Rozumowanie takie prowadzi do definicji modelu obrazu jako losowego pola Markowa, którego rząd określa powierzchnię sąsiedztwa wybranego piksela obrazu, jakie brane jest pod uwagę w procesie predykcji jego jasności, zaś korelacje sąsiednich pikseli, traktowanych jako realizacje zmiennych losowych rozważanego pola, określają lokalny charakter zmienności ich jasności w wybranych kierunkach.

4.2.2. Wybrane charakterystyki probabilistyczne pól Gaussa-Markowa

W niniejszym paragrafie podane zostaną kluczowe, z punktu widzenia praktyki problemu kompresji obrazów, charakterystyki probabilistyczne przyjętego modelu obrazu. Podajmy pierwszą z nich, która następnie będzie skomentowana.

Własność 4.2.

Niech X będzie stacjonarnym, separowalnym polem losowym Markowa w szerokim sensie o rzędzie równym jedności, zerowej wartości oczekiwanej, wariancji σ_X^2 i dodatnio określonej macierzy autokowariancji. Ponadto niech współczynniki autokorelacji wierszowej $\rho_r \triangleq \rho\,\{X_{m,n}X_{m,n+1}\}$ oraz kolumnowej $\rho_c \triangleq \rho\,\{X_{m,n}X_{m+1,n}\}$ będą dowolnymi liczbami rzeczywistymi z przedziału $(-1,\,1\,) \subset \mathbb{R}$, zaś $\xi^{\,(r)}, \xi^{\,(c)}$, ξ, ζ będą pewnymi stacjonarnymi polami losowymi o zerowych wartościach oczekiwanych. Wówczas dla dowolnych $m, n, k, l \in \mathbb{Z}$ zmienną $X_{m,n}$ pola X można zapisać na poniższe sposoby:

$$X_{m,n} \;=\; \rho_r\, X_{m,n-1} + \xi_n^{(r)}\,, \tag{4.6a}$$

$$E\,\{\,\xi_{n-k}^{(r)}\;\xi_n^{(r)}\,\} = \sigma_X^2\,(\,1-\rho_r^2\,)\,\delta_{k,0}\,; \tag{4.6b}$$

$$X_{m,n} \;=\; \rho_c\, X_{m-1,n} + \xi_m^{(c)}\,, \tag{4.7a}$$

$$E\,\{\,\xi_{m-k}^{(c)}\;\xi_m^{(c)}\,\} = \sigma_X^2\,(\,1-\rho_c^2\,)\,\delta_{k,0}\,; \tag{4.7b}$$

$$X_{m,n} = \rho_r X_{m,n-1} + \rho_c X_{m-1,n} - \rho_r\rho_c X_{m-1,n-1} + \xi_{\,m,n}\,, \tag{4.8a}$$

$$E\,\{\,\xi_{m-k,n-l}\;\xi_{m,n}\,\} = \sigma_X^2\,(\,1-\rho_r^2\,)\,(\,1-\rho_c^2\,)\,\delta_{k,0}\,\delta_{l,0}\,; \tag{4.8b}$$

Własność 4.2 wynikająca z definicji 4.6 i dobrze znana z literatury [47] dotyczy postaci reprezentacji wierszowej, kolumnowej i przyczynowej (unilateralnej) rozważanego separowalnego pola Markowa. Dowód na istnienie podanych postaci pola Markowa wynika niemalże natychmiast z faktów jego separowalności i przyczynowej reprezentacji pola (której jawna postać dana jest wzorem (86) w pracy [49]), będącej bezpośrednią konsekwencją definicji 4.6. Podajmy drugą z własności, która niemalże natychmiast wynika z tej powyższej, uwzględniając dodatkowo założenie o gaussowości rozważanego pola Markowa.

Własność 4.3.

Niech X będzie separowalnym, stacjonarnym, polem losowym Gaussa–Markowa w szerokim sensie o zerowej wartości oczekiwanej, rzędzie równym jedności i wariancji σ_X^2. Ponadto niech współczynniki autokorelacji wierszowej $\rho_r \triangleq \rho\{X_{m,n}X_{m,n+1}\}$ i kolumnowej $\rho_c \triangleq \rho\{X_{m,n}X_{m+1,n}\}$, dla każdego $m,n \in \mathbb{Z}$, będą dowolnymi liczbami rzeczywistymi z przedziału $(-1,\,1) \subset \mathbb{R}$. Niech $k \in \mathbb{N}$, zaś $\mathbf{\Omega}_{m,n} \subset \mathbb{Z}^2$ będzie uporządkowanym zbiorem indeksów takim, że: $\{(m-1,n),\,(m,n-1),\,(m-1,n-1)\} \subseteq \mathbf{\Omega}_{m,n} \subset \mathbb{P}_{m,n}$[1]. Wówczas dla dowolnych punktów $\mathbf{x} \in \mathbb{R}^{\overline{\Omega}_{m,n}}$ oraz $(x_{m-1,n},\ldots,x_{m-k,n})$, $(x_{m,n-1},\ldots,x_{m,n-k}) \in \mathbb{R}^k$ prawdziwe są następujące stwierdzenia i równości:

Pole X jest polem Markowa w ścisłym sensie o rzędzie równym jeden. (4.9)

$$f_{X_{m,n}}(x) = \mathcal{N}(0, \sigma_X^2)(x) \tag{4.10}$$

$$f_{X_{m,n}\,|\,X_{m,n-1}\ldots X_{m,n-k}}(x\,|\,x_{m,n-1},\ldots,x_{m,n-k}) = \mathcal{N}(\rho_r\,x_{m,n-1}, \sigma_X^2(1-\rho_r^2))(x) \tag{4.11a}$$

$$f_{X_{m,n}\,|\,X_{m-1,n}\ldots X_{m-k,n}}(x\,|\,x_{m-1,n},\ldots,x_{m-k,n}) = \mathcal{N}(\rho_c\,x_{m-1,n}, \sigma_X^2(1-\rho_c^2))(x) \tag{4.11b}$$

$$f_{X_{m,n}\,|\,\mathbf{X}_{\mathbf{\Omega}_{m,n}}}(x|\mathbf{X}) = \mathcal{N}(\rho_r x_{m,n-1}+\rho_c x_{m-1,n}-\rho_r\rho_c x_{m-1,n-1}, \sigma_X^2(1-\rho_r^2)(1-\rho_c^2))(x) \tag{4.12}$$

Własność 4.4.

Niech X będzie separowalnym, stacjonarnym polem losowym Markowa w szerokim sensie o zerowej wartości oczekiwanej, rzędzie równym jeden, wariancji σ_X^2 i dodatnio określonej macierzy autokowariancji. Dla dowolnych (m,n), (m_r,n_r), $(m_c,n_c) \in \mathbb{Z}^2$ zdefiniujmy macierz $\mathbf{X}$ i wektory $\mathbf{x}_r$, $\mathbf{x}_c$ zmiennych losowych rozważanego pola następująco $x_{i,j} \triangleq X_{m+i-1,n+j-1}$; $(x_r)_j \triangleq X_{m_r,n_r+j-1}$; $(x_c)_i \triangleq X_{m_c+i-1,n_c}$; $i = 1,\ldots,N$, $j = 1,\ldots,M$. Niech współczynniki autokorelacji wierszowej $\rho_r \triangleq \rho\{X_{m,n}X_{m,n+1}\}$ i kolumnowej $\rho_c \triangleq \rho\{X_{m,n}X_{m+1,n}\}$ będą dowolnie wybranymi liczbami rzeczywistymi z przedziału $(-1,\,1) \subset \mathbb{R}$. Zachodzą wówczas zależności:

$$(\rho\{\mathbf{x}_r\,\mathbf{x}_r^T\})_{i,j} = \rho_r^{\,|\,i-j\,|};\quad i,j = 1,\ldots,N \tag{4.13a}$$

$$(\rho\{\mathbf{x}_c\,\mathbf{x}_c^T\})_{i,j} = \rho_c^{\,|\,i-j\,|};\quad i,j = 1,\ldots,M \tag{4.13b}$$

$$E\left\{vec(\mathbf{X})\,vec(\mathbf{X})^T\right\} = \sigma_X^2\;\rho\{\mathbf{x}_r\,\mathbf{x}_r^T\} \otimes \rho\{\mathbf{x}_c\,\mathbf{x}_c^T\} \tag{4.14}$$

[1] gdzie $\mathbb{P}_{m,n}$ jest lewą górną półpłaszczyzną względem punktu (m,n) nie zawierającą tego punktu

Ostatnia z podanych własności, tzn. własność 4.4, określa postaci macierzy autokowariancji (autokorelacji) procesów wierszowego, kolumnowego i pola losowego modelu obrazu. Zapisując jawnie postaci wspomnianych macierzy i dla wygody zapisu przyjmując: $\mathbf{R}_r \triangleq \rho\{\mathbf{x}_r\,\mathbf{x}_r^T\}$, $\mathbf{R}_c \triangleq \rho\{\mathbf{x}_c\,\mathbf{x}_c^T\}$ oraz $\mathbf{K} \triangleq E\{vec(\mathbf{X})\,vec(\mathbf{X})^T\}$, mamy:

$$\mathbf{R}_r = \begin{pmatrix} 1 & \rho_r & \rho_r^2 & \cdots & \rho_r^{N-1} \\ \rho_r & 1 & \rho_r & \cdots & \rho_r^{N-2} \\ \rho_r^2 & \rho_r & 1 & \cdots & \rho_r^{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_r^{N-1} & \rho_r^{N-2} & \rho_r^{N-3} & \cdots & 1 \end{pmatrix}_{N\times N} ; \begin{pmatrix} 1 & \rho_c & \rho_c^2 & \cdots & \rho_c^{M-1} \\ \rho_c & 1 & \rho_c & \cdots & \rho_c^{M-2} \\ \rho_c^2 & \rho_c & 1 & \cdots & \rho_c^{M-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_c^{M-1} & \rho_c^{M-2} & \rho_c^{M-3} & \cdots & 1 \end{pmatrix}_{M\times M} = \mathbf{R}_c$$

$$\mathbf{K} = \sigma_X^2 \left[\begin{pmatrix} 1 & \rho_r & \rho_r^2 & \cdots & \rho_r^{N-1} \\ \rho_r & 1 & \rho_r & \cdots & \rho_r^{N-2} \\ \rho_r^2 & \rho_r & 1 & \cdots & \rho_r^{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_r^{N-1} & \rho_r^{N-2} & \rho_r^{N-3} & \cdots & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & \rho_c & \rho_c^2 & \cdots & \rho_c^{M-1} \\ \rho_c & 1 & \rho_c & \cdots & \rho_c^{M-2} \\ \rho_c^2 & \rho_c & 1 & \cdots & \rho_c^{M-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_c^{M-1} & \rho_c^{M-2} & \rho_c^{M-3} & \cdots & 1 \end{pmatrix} \right]$$

Podane tutaj własności mają znaczenie praktyczne i autor będzie się na nie powoływał w dalszej części badań eksperymentalnych w aspekcie syntezy obrazów modelowych służących do systematycznej weryfikacji charakterystyk jakościowych szybkich sieci neuronowych w problemie kompresji obrazów.

4.2.3. Generator obrazów modelowych

W niniejszym podrozdziale przedstawiony zostanie algorytm generowania obrazów modelowych realizujących przedstawiony model Gaussa-Markowa obrazu naturalnego. Otrzymane obrazy mogą posłużyć następnie do weryfikacji teoretycznych zdolności wybranych architektur sieci neuronowych do realizacji efektywnych jakościowo przekształceń dyskretnych przeznaczonych dla potrzeb zadania kompresji różnych statystycznie klas obrazów naturalnych zgodnych ze wspomnianym modelem obrazu. Takie podejście pozwala w prosty i systematyczny sposób ustalić potencjalną przydatność badanych architektur sieciowych w zadaniu kompresji wybranych rodzajów obrazów naturalnych bez uciążliwej konieczności przeszukiwania dostępnych baz danych obrazowych w celu znalezienia obrazów o zadanych charakterystykach statystycznych. Prezentowany algorytm, opisany w artykule [87], jest szczególnym przypadkiem ogólnej metody syntezy obrazów wzorcowych realizujących model Gaussa-Markowa obrazu naturalnego przedstawionej w pracy [47]. Propozycje podobnych metod syntezy pól losowych Gaussa-Markowa można znaleźć chociażby w pracach [15, 16, 45]. Przejdźmy zatem do prezentacji rozważanego algorytmu.

Korzystając bezpośrednio z podanych wcześniej wzorów (4.10), (4.11a), (4.11b) oraz (4.12) opisujących charakterystyki probabilistyczne rozważanych pól losowych algorytm syntezy K–elementowego zbioru $M \times N$–punktowych, $M, N, K \in \mathbb{N}$ realizacji

stacjonarnego pola losowego Gaussa-Markowa pierwszego rzędu o zadanej wariancji $\sigma^2 \in \mathbb{R}_+$ i współczynnikach autokorelacji wierszowej i kolumnowej $\rho_r, \rho_c \in (-1,\, 1\,)$ można zapisać następująco:

Alg. 4.1: Algorytm symulacji pola Gaussa-Markowa pierwszego rzędu

$M \in \mathbb{N}$: wymiar wertykalny symulowanego pola ;
$N \in \mathbb{N}$: wymiar horyzontalny symulowanego pola ;
$K \in \mathbb{N}$: ilość próbek do wygenerowania ;
$\sigma^2 \in \mathbb{R}_+$: wariancja symulowanego pola ;
$\rho_r \in (-1,\, 1\,)$: współczynnik autokorelacji wierszowej pola ;
$\rho_c \in (-1,\, 1\,)$: współczynnik autokorelacji kolumnowej pola ;
$\mathcal{X} = \{\ \mathbf{X}_k \in \mathbb{R}^{M \times N},\ \ k = 1, \ldots, K \in \mathbb{N}\ \}:$ K - elementowy zbiór próbek wyjściowych ;

---

dla $k = 1$ **do** K **powtarzaj**

Wylosuj piksel $x_{0,0}^{(k)}$ z następującego wzoru rozkładu:

$$x_{0,0}^{(k)} \sim \mathcal{N}\,(\,0\,,\sigma^2\,)\,; \tag{4.1.1}$$

dla $m = 1$ **do** M **powtarzaj**

Wylosuj piksel $x_{m,0}^{(k)}$ z następującego wzoru rozkładu:

$$x_{m,0}^{(k)} \sim \mathcal{N}\,(\,\rho_c\, x_{m-1,n}^{(k)}\,,\sigma^2\,(\,1-\rho_c^2\,)\,)\,; \tag{4.1.2}$$

koniec pętli

dla $n = 1$ **do** N **powtarzaj**

Wylosuj piksel $x_{0,n}^{(k)}$ z następującego wzoru rozkładu:

$$x_{0,n}^{(k)} \sim \mathcal{N}\,(\,\rho_r\, x_{m,n-1}^{(k)}\,,\sigma^2\,(\,1-\rho_r^2\,)\,)\,; \tag{4.1.3}$$

koniec pętli

dla $m = 1$ **do** M **powtarzaj**

dla $n = 1$ **do** N **powtarzaj**

Wylosuj piksel $x_{m,n}^{(k)}$ z następującego wzoru rozkładu:

$$x_{m,n}^{(k)} \sim \mathcal{N}(\rho_r x_{m,n-1}^{(k)} + \rho_c x_{m-1,n}^{(k)} - \rho_r \rho_c x_{m-1,n-1}^{(k)}, \sigma^2(1-\rho_r^2)(1-\rho_c^2))\,; \tag{4.1.4}$$

koniec pętli

koniec pętli

Zapisz wylosowaną w ten sposób macierz pikseli $\mathbf{X}_k$ w zbiorze wyjściowym $\mathcal{X}$;

koniec pętli

Można stosunkowo łatwo udowodnić fakt [47, 87], że procedura podana pseudokodem 4.1 określa prawidłowy sposób syntezy próbek stacjonarnego pola Gaussa-Markowa pierwszego rzędu o zadanych parametrach probabilistycznych. W implementacji kroków (4.1.1) - (4.1.4) rozważanego algorytmu do testów porównawczych wybranych technik kompresji obrazów opisanych w kolejnym rozdziale zastosowano generator próbek z rozkładu normalnego o regulowanych parametrach jakościowo

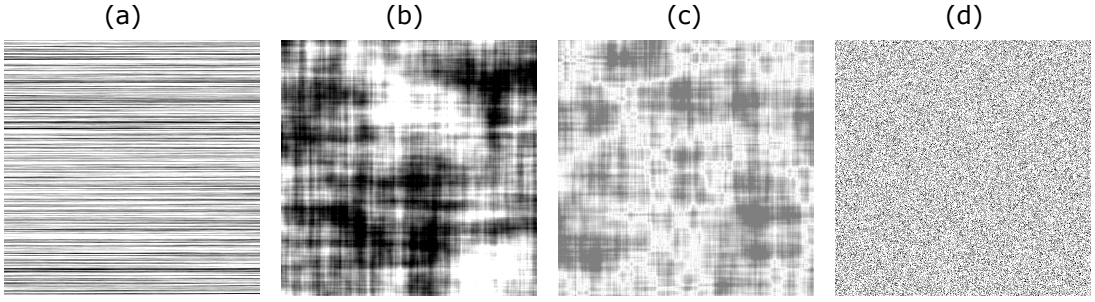
efektywnościowych przedstawiony w pracy [113], którego kod zapisany w języku C uwidoczniony jest na poniższym schemacie.

Alg. 4.2: Generator próbek z rozkładu Gaussa o regulowanych parametrach

```
//------------------------------------------------------------------------//
// Generator próbek z rozkładu Gaussa                                     //
// o regulowanych parametrach jakości/szybkości                           //
//------------------------------------------------------------------------//
// wejście:                                                               //
//  mean - wartość oczekiwana rozkładu                                    //
//  var  - wariancja rozkładu                                             //
//  n    - jakość                                                         //
//------------------------------------------------------------------------//
// wyjście:                                                               //
//  realizacja próbki z rozkładu normalnego                               //
//  o zadanych parametrach wejściowych                                    //
//------------------------------------------------------------------------//

float grand(float mean, float var, int n)
{
 float result = 0;
 for (int i = 0; i < n; i++)
  result += rand() / RAND_MAX - 0.5;
 return result * sqrt(12 * var / n) + mean;
}
```

Na poniższym rysunku przedstawiono kilka przykładów obrazów o zadanych parametrach statystycznych uzyskanych za pomocą generatora 4.1 pól losowych Gaussa-Markowa. Obrazy pokazane na rysunku 4.1 są obrazami o oryginalnych wymiarach 512×512 pikseli i zostały odpowiednio przeskalowane w taki sposób, aby możliwa była ich sugestywna prezentacja w 8-bitowej skali szarości.



Rys. 4.1: Przykładowe obrazy uzyskane za pomocą generatora pól Gaussa-Markowa

Pokazane na rysunku 4.1 przykładowe realizacje pól losowych Gaussa-Markowa o jednostkowej wariancji mają współczynniki autokorelacji wierszowej ρ_r oraz kolumnowej ρ_c równe odpowiednio wartościom (a) $\rho_r = 1, \rho_c = 0$; (b) $\rho_r = 0.99, \rho_c = 0.99$; (c)

$\rho_r = -0.99$, $\rho_c = 0.99$ oraz `(d)` $\rho_r = 0$, $\rho_c = 0$. Warto zweryfikować poprawność algorytmu 4.1 symulacji pól losowych Gaussa-Markowa w kontekście problemu kompresji obrazów. W następnym podpunkcie przedstawione zostaną zatem definicje odpowiednich miar jakości dla generowanych realizacji rozważanych pól wraz wynikami procesu ich symulacji.

4.2.4. Analiza jakościowa generatora obrazów modelowych

W niniejszym podpunkcie przedstawiona zostanie analiza jakościowa algorytmu 4.1 symulacji obrazów naturalnych dla testów technik kompresji obrazów. Na wstępie warto zauważyć, że równania definicyjne (4.1.1) - (4.1.4) metody 4.1 generowania obrazów wzorcowych implikują fakt [47, 87], że opisywane przez nie pole losowe jest w istocie stacjonarnym polem losowym Gaussa-Markowa pierwszego rzędu określonym na skończonym obszarze prostokątnym. W związku z tym powyższa kwestia nie musi być dalej przedmiotem analizy jakościowej rozważanego algorytmu, gdyż sama definicja metody 4.1 zapewnia jej spełnienie. Interesującymi natomiast z punktu widzenia modelowania zadania kompresji obrazów są przedstawione niżej trzy miary opisujące podobieństwo wybranych charakterystyk statystycznych zbiorów wygenerowanych za pomocą algorytmu 4.1 próbek obrazowych do ich odpowiedników probabilistycznych właściwych dla modelowanych pól Gaussa-Markowa, które określane są często w literaturze mianem pól typu GMRF.[1]

Przejdźmy zatem do definicji rozważanych miar oraz ich szczegółowego opisu. Niech $M, N \in \mathbb{N}$ załóżmy, że dysponujemy $K \in \mathbb{N}$ - elementowym zbiorem $\mathcal{X}$ złożonym z $M \times N$ - punktowych próbek rzeczywistych stanowiących pożądane realizacje pola GMRF o zerowej wartości oczekiwanej i zadanej wariancji $\sigma^2 \in \mathbb{R}_+$ wygenerowanym za pomocą dowolnej metody symulacyjnej, takiej jak chociażby algorytm 4.1. Podajmy najpierw definicje interesujących z punktu widzenia rozważanego problemu estymatorów wartości oczekiwanej $[\,\widetilde{\mu}_1 \ldots \widetilde{\mu}_{M\cdot N}\,]^T$ próbki obrazowej ze zbioru $\mathcal{X}$ oraz jego macierzy autokowariancji $\widetilde{\mathbf{K}}$. Dla $m, n \in \{1, \ldots, M \cdot N\}$ rozważane estymatory zdefiniowane są w następujący sposób:

$$\widetilde{\mu}\,(\,\mathcal{X}\,)_{\,n} \triangleq \frac{1}{K} \sum_{k=1}^{K} vec\,(\,\mathbf{X}_k)_{\,n} \tag{4.15a}$$

$$[\,\widetilde{\mathbf{K}}\,(\,\mathcal{X}\,)\,]_{\,m,n} \triangleq \frac{1}{K} \sum_{k=1}^{K} [\,vec\,(\,\mathbf{X}_k)_{\,m} - \widetilde{\mu}\,(\,\mathcal{X}\,)_{\,m}\,] \cdot [\,vec\,(\,\mathbf{X}_k)_{\,n} - \widetilde{\mu}\,(\,\mathcal{X}\,)_{\,n}\,] \tag{4.15b}$$

gdzie $\mathbf{X}_k$, $k = 1, \ldots, K$ są pojedynczymi $M \times N$–pikselowymi próbkami pochodzącymi ze zbioru $\mathcal{X}$, zaś $vec\{\cdot\}$ jest operatorem wektoryzacji macierzy. Powyższe estymatory są *jedynymi* parametrami statystycznymi mającymi wpływ na postać przekształceń optymalnych w problemie kompresji obrazów - patrz [44]. A zatem można dojść do wniosku, że aby poprawnie symulować za pomocą zbioru próbek obrazowych $\mathcal{X}$ proces optymalizacji sieci `MLP2D` oraz `NFJT2D` w zadaniu kompresji obrazów modelowanych wybranym polem losowym Gaussa-Markowa o ustalonych parametrach probabilistycznych należy zapewnić równość wymienionych wyżej dwóch charakterystyk statystycznych modelującego zbioru próbek treningowych $\mathcal{X}$ z odpowiadającymi im

[1] ang. *Gauss-Markov random field*

teoretycznymi parametrami - wartości oczekiwanej i macierzy autokowariancji symulowanego pola losowego. Wobec tego miary jakości algorytmu generowania obrazów modelowych dla symulacji optymalizacji rozważanych sieci w zadaniu kompresji, zgodnych z modelem stochastycznym Gaussa-Markowa, obrazów naturalnych mogą być zdefiniowane w następujący sposób:

$$e_{\mu}(\,\mathcal{X}\,) \stackrel{\Delta}{=} \max_{n\,\in\,\{\,1,\ldots,M\cdot N\,\}} |\;\widetilde{\mu}\,(\,\mathcal{X}\,)_{\,n}\;| \tag{4.16a}$$

$$e_{\,\mathbf{K}}(\,\mathcal{X}\,) \stackrel{\Delta}{=} \max_{m,n\,\in\,\{\,1,\ldots,M\cdot N\,\}} |\;[\,\widetilde{\mathbf{K}}\,(\,\mathcal{X}\,)\,]_{\,m,n}\;-\;(\,\mathbf{K}\,)_{\,m,n}\;| \tag{4.16b}$$

gdzie $MN\times MN$–elementowa macierz $\mathbf{K}$ jest macierzą autokowariancji symulowanego pola losowego Gaussa-Markowa. Zerowe wartości miar (4.16) dla ustalonego zbioru próbek $\mathcal{X}$ gwarantują równość zbiorów par macierzy optymalnych w zagadnieniu kompresji obrazów dla obrazów modelowanych teoretycznie zadanym polem Gaussa-Markowa oraz zbioru próbek wzorcowych $\mathcal{X}$ symulującym wspomniane pole losowe, co można uznać za główny cel realizacji rozważanej symulacji.

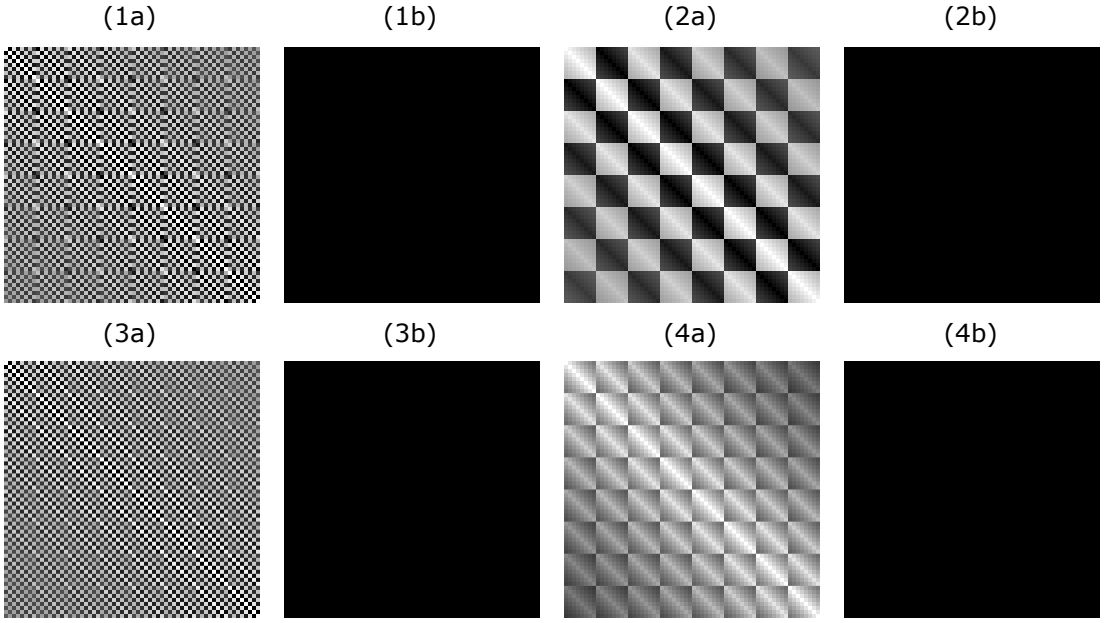
Zgodnie z powyższymi wnioskami miary zdefiniowane zależnościami (4.16) w pełni charakteryzują jakość wybranego zbioru próbek wzorcowych ze względu na postaci przekształceń optymalnych w rozważanych procesach kompresji. Należy jeszcze zastanowić się nad definicją miary określającej, chociażby w przybliżeniu, stopień w jakim rozkład wartości zbioru próbek wzorcowych naśladuje gaussowski charakter symulowanego pola losowego, gdyż charakterystyka taka może mieć również potencjalny wpływ na wyniki procesu optymalizacji rozważanych sieci neuronowych w symulacji zadania kompresji adaptacyjnej. Bezpośrednia weryfikacja podobieństw łącznych, teoretycznych funkcji gęstości prawdopodobieństw symulowanych pól losowych, stanowiących ich pełny opis probabilistyczny, do estymatorów tych funkcji uzyskanych ze zbiorów próbek treningowych nie jest możliwa w praktyce ze względu na rozmiary wspomnianych zbiorów. Dlatego też autor zdecydował się zdefiniować miarę, która może być uważana za pierwsze przybliżenie faktu gaussowskich charakterystyk wygenerowanych zbiorów próbek wzorcowych. Miara ta zatem charakteryzować będzie z konieczności jedynie podobieństwo, w ramach całości zbioru próbek, rozkładu wartości pojedynczych pikseli do jednowymiarowej gaussowskiej funkcji gęstości prawdopodobieństwa. Aby podać jej postać określmy histogram pojedynczego piksela zbioru obrazów wzorcowych $\mathcal{X}$: $\forall\, m \in \{\,0,\ldots,M-1\,\}$, $\forall\, n \in \{\,0,\ldots,N-1\,\}$

$$H(\;i\;)^{(\mathcal{X})}_{m,n} \stackrel{\Delta}{=} 1\;/\;(\,p\;\Delta\,)\sum_{x^{(k)}_{m,n}\in(\,i-\Delta,i+\Delta\,)} x^{(k)}_{m,n}\,;\quad i\in\mathbb{I}\,;\quad k=1,\ldots,K\,; \tag{4.17}$$

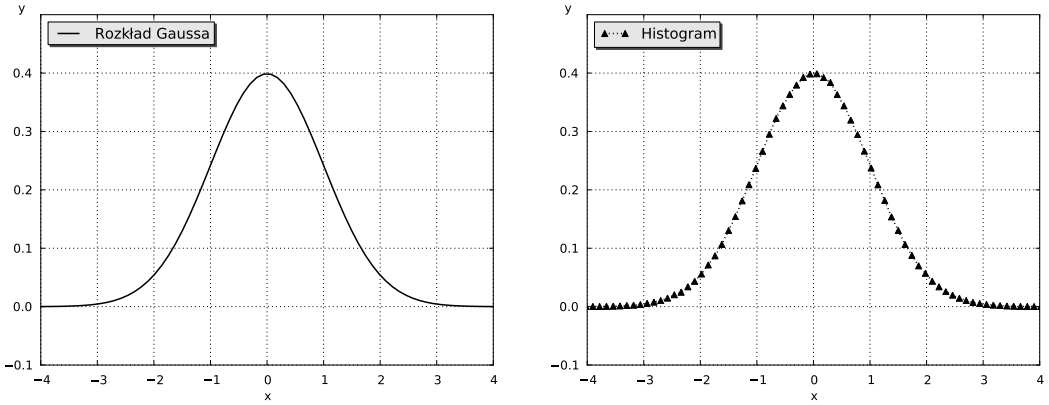
gdzie $p\in\mathbb{N}$ oznacza rozdzielczość histogramu, $\mathbb{I} \stackrel{\Delta}{=} \{\,i_{min},\, i_{min}+\Delta\,,\ldots,\, i_{max}-\Delta,\, i_{max}\,\}$, gdzie $\Delta\in\mathbb{R}$ jest długością pojedynczego przedziału histogramu równą, zgodnie z „regułą $4-\sigma$” [113], wartości $8\,\sigma/p$, zaś $i_{min}, i_{max}\in\mathbb{R}$ są punktami referencyjnymi histogramu równymi odpowiednio $-4\,\sigma+\Delta$ oraz $4\,\sigma-\Delta$. Wówczas rozważaną miarę jakości zadanego zbioru próbek wzorcowych można zdefiniować następująco:

$$e_{\mathcal{N}}(\,\mathcal{X}\,) \stackrel{\Delta}{=} \max_{(m,n)\,;\,i\,\in\,\mathbb{I}} |\,H(\;i\;)^{(\mathcal{X})}_{m,n} - \mathcal{N}\,(\,\mu\,,\,\sigma^2\,)(\;i\;)\;| \tag{4.18}$$

Diagramy poniżej są graficznymi interpretacjami wprowadzonych miar dla przykładowych zbiorów próbek.



Rys. 4.2: Macierze autokowariancji dla symulowanych pól losowych



Rys. 4.3: Rozkład Gaussa i histogram wybranego piksela jednego z pól losowych

Tab. 4.1: Miary błędów e_μ, $e_{\mathbf{K}}$ oraz $e_{\mathcal{N}}$ dla przykładowych realizacji pól losowych

Miary błędów	*Współczynniki autokorelacji wierszowej i kolumnowej*			
	$(-0.9, -0.9)$	$(-0.9, +0.9)$	$(+0.9, -0.9)$	$(+0.9, +0.9)$
$e_\mu(\mathcal{X})$	$1.00 \cdot 10^{-4}$	$-2.90 \cdot 10^{-4}$	$-9.00 \cdot 10^{-5}$	$-1.30 \cdot 10^{-4}$
$e_{\mathbf{K}}(\mathcal{X})$	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$
$e_{\mathcal{N}}(\mathcal{X})$	$2.71 \cdot 10^{-3}$	$3.56 \cdot 10^{-3}$	$4.20 \cdot 10^{-3}$	$2.58 \cdot 10^{-3}$

Na rysunku 4.2 przedstawiono estymatory macierzy autokowariancji (wzór (4.15b)) przykładowych zbiorów próbek obrazowych, o wymiarach próbki równych 8×8 pikseli oraz liczności każdego z tych zbiorów równej 10^6 próbek, uzyskanych za pomocą generatora 4.1 - rysunki `(1a)`, `(2a)`, `(3a)` i `(4a)`. Na rysunkach, odpowiednio: `(1b)`, `(2b)`, `(3b)` i `(4b)`, pokazano macierze różnicowe, których elementy są modułami różnic pomiędzy estymatorami macierzy autokowariancji wygenerowanych za pomocą algorytmu 4.1 zbiorów próbek obrazowych a odpowiadającymi im teoretycznymi macierzami autokowariancji pól Gaussa-Markowa modelowanych tymi zbiorami. Postaci teoretycznych macierzy autokowariancji wyznaczone były ze wzoru (4.14). Symulowane pola losowe miały jednostkową wariancję, zerową wartość oczekiwaną i współczynniki autokorelacji równe wartościom: `(1a)` $\rho_r = -0.9,\ \rho_c = -0.9$; `(2a)` $\rho_r = -0.9,\ \rho_c = 0.9$; `(3a)` $\rho_r = 0.9,\ \rho_c = -0.9$; `(4a)` $\rho_r = 0.9,\ \rho_c = 0.9$. Jak widać z rysunku 4.2 macierze różnicowe są złożone prawie wyłącznie z elementów zerowych, co oznacza, że uzyskano zbiory o charakterystykach autokorelacyjnych dobrze naśladujących założone parametry teoretyczne. Rysunek 4.3 pokazuje porównanie histogramu (4.17) o rozdzielczości $p = 100$ dla pojedynczego piksela o współrzędnych $(7, 7)$ - generowanego, zgodnie z przebiegiem algorytmu 4.1, jako ostatni piksel dla każdej z próbek - oraz założonego rozkładu Gaussa tego piksela dla przykładowego zbioru modelującego `(4a)` z rys. 4.2.

Jak widać założony rozkład został osiągnięty z dużą dokładnością. Wyniki liczbowe miar błędów e_μ, $e_{\mathbf{K}}$ i $e_{\mathcal{N}}$ dla uzyskanych zbiorów próbek zgromadzone w tabeli 4.1 potwierdzają wrażenia wizualne wynikające z przedstawionych wyżej rysunków o dobrej jakości symulacji pól Gaussa-Markowa dla potrzeb testów technik kompresji obrazów za pomocą metody 4.1. Dla wszystkich zbiorów próbek obrazów wzorcowych biorących udział w symulacjach teoretycznych procesów kompresji obrazów zgodnych z modelem Gaussa-Markowa przedstawionych w dalej wynikach badań eksperymentalnych wprowadzone wyżej miary błędów mieściły się w następujących zakresach wartości:

$$e_\mu < 10^{-3}\,; \quad e_{\mathbf{K}} < 10^{-2}\,; \quad e_{\mathcal{N}} < 10^{-3}\,; \tag{4.19}$$

Zakresy takie spowodowały, że maksymalne odchylenie faktycznych wartości miar PSNR od ich przewidywanych poziomów teoretycznych nie przekraczało 10^{-2} dB po wszystkich wykonanych symulacjach procesu kompresji z udziałem przedstawionych dalej przekształceń dyskretnych poddawanych analizie jakościowej. Oznacza to, że przedstawiony w niniejszym podrozdziale algorytm symulacji pól losowych Gaussa-Markowa pierwszego rzędu modelujących obrazy rzeczywiste jest dobrą jakościowo metodą weryfikacji teoretycznych własności technik kompresji obrazów, w tym rozważanych w niniejszej pracy neuronowych procedur optymalizacyjnych.

4.3. Wyniki badań eksperymentalnych

W niniejszej części opracowania przedstawione zostaną wyniki badań eksperymentalnych dla zaprezentowanych szybkich sieci neuronowych w problemie kompresji obrazów. Badania zostały podzielone na dwa opisane poniżej eksperymenty główne.

Eksperyment pierwszy miał na celu ustalenie w sposób systematyczny teoretycznych charakterystyk jakościowych zaproponowanej w pracy sieci neuronowej do kompresji obrazów. Do zbadania wspomnianych charakterystyk użyto generowanych sztucznie, za pomocą generatora 4.1 przedstawionego w poprzednim rozdziale, obrazów symulujących pola losowe Gaussa-Markowa o zadanych parametrach statystycznych, które to pola stanowią, zgodnie z rozważaniami poczynionymi w poprzednich rozdziałach, ogólnie przyjęty dla problemu kompresji obrazów probabilistyczny model obrazu naturalnego. Obrazy modelujące wspomniane pola losowe posłużyły za wzorce treningowe dla testowanych architektur sieci neuronowych, tzn. architektury szybkiej `NFJT2D` i standardowej `MLP2D`, a także użyte zostały później do przeprowadzenia testów porównawczych symulacji procesu kompresji obrazu dla wymienionych wyżej sieci oraz standardowych metod kompresji obrazów z użyciem dwuwymiarowych przekształceń dyskretnych - kosinusowego `FCT2D` oraz transformaty Karhunena-Loève'go `KLT2D`. Dzięki wynikom uzyskanym z eksperymentu pierwszego dokonano identyfikacji jednej z klas statystycznych obrazów naturalnych, dla której ogólne charakterystyki efektywnościowe proponowanej architektury neuronowej wyprzedzają istotnie te właściwe dla pozostałych metod kompresji, z którymi porównywana była proponowana metoda. Identyfikacja ta z kolei posłużyła jako wskazówka doboru rodzajów obrazów rzeczywistych dla testów przeprowadzonych w eksperymencie drugim.

Eksperyment drugi służył weryfikacji możliwości proponowanej architektury neuronowej w sytuacji rzeczywistej, obejmującej zadanie kompresji obrazów uzyskanych z rzeczywistych zdjęć przedstawiających wybrane przedmioty czy postaci. W eksperymencie tym do testów jakościowych szybkiej sieci neuronowej użyto obrazów o charakterystykach statystycznych zbliżonych do tych, właściwych obrazom sztucznym badanym w eksperymencie pierwszym. Obrazy eksperymentalne pogrupowane były w pary, z których każda poddana została standardowej procedurze badawczej [28, 30, 62] polegającej na optymalizacji badanych sieci neuronowych za pomocą pojedynczego obrazu trenującego i weryfikacji rezultatów jakościowych procesu nauki sieci zarówno dla obrazu treningowego jak i odpowiadającego mu obrazu testowego. Tutaj także proponowana architektura neuronowa porównywana była nie tylko z jej standardowym odpowiednikiem stosowanym najczęściej w zadaniu adaptacyjnej kompresji obrazów, tzn. siecią typu `MLP2D`, ale także ze standardowymi metodami kompresji z użyciem przekształcenia szybkiego `FCT2D` oraz optymalnej transformaty referencyjnej `KLT2D`.

Na końcu warto dodać, że obydwa eksperymenty obejmowały także kompleksowe porównania efektywnościowe (szybkościowe) analizowanych metod kompresji adaptacyjnej. Przejdźmy teraz do szczegółowego opisu przeprowadzonych eksperymentów.

Ponieważ w obydwu opisanych dalej szczegółowo eksperymentach jako przekształcenia referencyjnego, optymalnego w badanym zagadnieniu adaptacyjnej kompresji obrazów, użyto transformaty Karhunena-Loève'go warto dla uniknięcia nieścisłości podać metodę wyznaczania tego przekształcenia. Poniżej przedstawiono pseudokod procedury obliczania estymatora dwuwymiarowej transformaty Karhunena-Loève'go K–elementowego zbioru $M \times N$–punktowych próbek rzeczywistych.

Alg. 4.3: Estymacja dwuwymiarowej transformaty Karhunena-Loève'go

$M \in \mathbb{N}$: wymiar wertykalny elementów zbioru próbek ;
$N \in \mathbb{N}$: wymiar horyzontalny elementów zbioru próbek ;
$\mathcal{X} = \{ \mathbf{X}_k \in \mathbb{R}^{M \times N}, \quad k = 1, \ldots, K \in \mathbb{N} \}$: K - elementowy zbiór próbek wejściowych ;
$\mathbf{U} \in \mathbb{R}^{MN \times MN}$: przekształcenie wyjściowe ;

Wyznacz estymator $\widetilde{\mathbf{x}} \in \mathbb{R}^{MN}$ wektora oczekiwanego zbioru wejściowego $\mathcal{X}$ według wzoru:

$$\forall\, n \in \{ 1, \ldots, M \cdot N \} \quad \widetilde{x}_n = \frac{1}{K} \sum_{k=1}^{K} vec\,(\,\mathbf{X}_k)_{\,n} \,; \tag{4.2.1}$$

Utwórz ze zbioru $\mathcal{X}$ pomocniczy zbiór wektorów $\mathcal{Z} = \{ \mathbf{z}_k \in \mathbb{R}^{MN}, \quad k = 1, \ldots, K \}$ o zerowym estymatorze wektora oczekiwanego poprzez usunięcie składowej stałej pierwszego z wymienionych zbiorów w następujący sposób:

$$\forall\, k \in \{ 1, \ldots, K \} \quad \mathbf{z}_k = vec\,(\,\mathbf{X}_k) - \widetilde{\mathbf{x}} \,; \tag{4.2.2}$$

Z wektorów zbioru $\mathcal{Z}$ utwórz $M \cdot N \times K$ - elementową macierz $\mathbf{Z}$ w następujący sposób (zapis $(\,\mathbf{Z}\,)_k$ oznacza k - tą kolumnę macierzy $\mathbf{Z}$) :

$$\forall\, k \in \{ 1, \ldots, K \} \quad (\,\mathbf{Z}\,)_k = \mathbf{z}_k \,; \tag{4.2.3}$$

Wyznacz estymator $\widetilde{\mathbf{K}} \in \mathbb{R}^{MN \times MN}$ macierzy autokowariancji wektorów ze zbioru $\mathcal{Z}$ następująco:

$$\widetilde{\mathbf{K}} = \mathbf{Z} \cdot \mathbf{Z}^{\,T} \,; \tag{4.2.4}$$

Utwórz macierz $\mathbf{U} \in \mathbb{R}^{MN \times MN}$, której wiersze stanowią wektory własne macierzy $\widetilde{\mathbf{K}}$ uporządkowane zgodnie z malejącym porządkiem wartości własnych tej macierzy. Wówczas $\mathbf{U}$ jest szukanym estymatorem macierzy dwuwymiarowej transformaty Karhunena-Loève'go dla zbioru $\mathcal{X}$;

Dla obydwu eksperymentów głównych zbiory próbek, oznaczone w pseudokodzie 4.3 symbolem $\mathcal{X}$, pozyskiwane były opisanymi dalej szczegółowo metodami właściwymi dla wybranego eksperymentu. Warto dla ścisłości podać także jawnie definicję miary jakości PSNR (stosunku szczytowego sygnału do szumu) odtwarzania zakodowanych w procesie kompresji obrazów. Dla obydwu eksperymentów miara ta wynosiła:

$$\begin{aligned} PSNR &= 10\, log_{\,10} \left(\frac{x_{max}^2}{MSE} \right) \quad [\, dB \,] \\ MSE &= \frac{1}{KN^2} \sum_{k=1}^{K} \sum_{n=1}^{N^2} [\; vec\,(\,\mathbf{X}_k)_{\,n} - vec\,(\,\hat{\mathbf{X}}_k)_{\,n} \,]^{\,2} \end{aligned} \tag{4.21}$$

We wzorach (4.21) $x_{max} \in \mathbb{R}_+$ jest maksymalną dopuszczalną wartością przetwarzanego sygnału dwuwymiarowego, $K \in \mathbb{N}$ jest licznością zbioru próbek podlegającego symulacji procesu kompresji i dekodowania, $N = 2^{\,m}$, $m \in \mathbb{N}$ jest pionowym i poziomym zarazem wymiarem pojedynczej próbki wspomnianego zbioru - w trakcie wszystkich eksperymentów używano jedynie próbek o jednakowych wymiarach,

$\mathbf{X}_k, \hat{\mathbf{X}}_k \in \mathbb{R}^{N\times N}$, $k = 1, \ldots, K$ są zaś pojedynczymi, odpowiadającymi sobie próbkami odpowiednio oryginalnego zbioru próbek obrazowych poddawanych procesowi kompresji oraz zbioru próbek odtworzonych w procesie dekodowania tych pierwszych. Wszystkie wymienione wyżej wielkości są parametrami poszczególnych testów, a ich dokładne wartości podane są dalej w opisach przeprowadzonych eksperymentów. Następny punkt dotyczy już pierwszego z zapowiedzianych eksperymentów głównych.

4.3.1. Wyniki badań dla obrazów modelowych

W niniejszym punkcie opisany zostanie pierwszy z dwóch eksperymentów głównych wykonany w ramach przeprowadzonych badań. Jak wspomniano we wprowadzeniu ma on na celu dokonanie systematycznej analizy teoretycznych charakterystyk jakościowo efektywnościowych zaproponowanej w niniejszej pracy architektury sieci neuronowej do kompresji obrazów. Eksperyment polegał na symulacji procesu kompresji i dekodowania skompresowanego sygnału dwuwymiarowego dla zbiorów próbek obrazowych wygenerowanych sztucznie za pomocą symulatora pól losowych Gaussa-Markowa przedstawionego w poprzednim rozdziale. Do testów symulowano pola losowe, modelujące obrazy naturalne, o zerowej wartości oczekiwanej $\mu = 0$, jednostkowej wariancji $\sigma^2 = 1$ oraz parach współczynników autokorelacji wierszowej i kolumnowej poszczególnych pól o następującej postaci: $(\rho_r^{(k)}, \rho_c^{(k)}) = (-0.1\cdot k\,,\, 0.1\cdot k\,)\,,\ k = 1, \ldots, 9$. Przy takich postaciach współczynników autokorelacji symulowanych pól losowych uzyskiwane obrazy będące ich realizacjami przypominają pasiaste wzory tekstur, tkanin czy drewna. Pojedynczy zbiór symulujący składał się z 10^6 próbek obrazowych o identycznych wymiarach poziomym i pionowym $N = 8$.

W przypadku sieci neuronowych `MLP2D` oraz `NFJT2D` badanie jakościowo efektywnościowe polegało na ich treningu przeprowadzanym metodą wstecznej propagacji błędu 3.1 z wykorzystaniem opisanych wyżej zbiorów próbek, a następnie teście obu sieci w każdym przypadku przy użyciu tego samego zbioru, który służył wcześniej za zbiór uczący. Warto dodać, że podczas treningu obydwu sieci na ich wejścia podawane były wszystkie spośród 10^6 próbek obrazowych każdego ze zbiorów uczących. Ten sam schemat postępowania dotyczył przekształcenia `KLT2D`, którego wektory bazowe wyznaczane były dla każdego ze zbiorów próbek z osobna przy użyciu algorytmu 4.3, transformata `FCT2D` natomiast jako przekształcenie niezależne od postaci kompresowanego sygnału nie wymagała optymalizacji. Należy tutaj wspomnieć, że podczas wszystkich testów w przypadku autokodera `NFJT2D` zastosowano minimalny reżim treningowy wynikający z ogólnego schematu obliczeniowego (2.46) dla zunifikowanych przekształceń dyskretnych, tzn. trening obejmował jedynie warstwy ostatnią obszaru kodującego, warstwę kwantyzacji i pierwszą warstwę obszaru dekodującego rozważanej sieci. Perceptron `MLP2D` trenowany był w standardowy sposób z optymalizacją warstw ukrytej i wyjściowej autokodera. Sam test główny polegał na tym, że wszystkie obrazy każdego ze zbiorów próbek poddawane były kompresji i dekodowaniu za pomocą wymienionych, zoptymalizowanych wcześniej czterech przekształceń liniowych w celu wyznaczenia miary PSNR dla każdego ze zbiorów próbek względem wybranego przekształcenia. Podczas testów transformaty `KLT2D`

i FCT2D pracowały w analogiczny sposób do ich neuronowych odpowiedników, MLP2D oraz NFJT2D, w schemacie autokodera z odrzucanymi w procesie kwantyzacji dokładnej odpowiednimi liczbami współczynników przekształceń prostych, przy czym stopień kompresji obrazów wejściowych, raportowany w tabelach 4.2b - 4.7b obliczany był według wzoru (3.12). Miara PSNR wyznaczania była dla każdej z par zbiór - przekształcenie za pomocą wzoru (4.21) z wartością parametru x_{max} równą, zgodnie z cytowaną wcześniej [113] „regułą $4-\sigma$" dla pól gaussowskich, wartości $4 \cdot \sigma = 4$. Warto w tym miejscu jeszcze raz przypomnieć fakt, iż jak wspomniano podczas omówienia algorytmu generowania pól losowych dla testów technik kompresji obrazów 4.1, odchylenie wartości PSNR estymowanej w praktyce przy pomocy zależności (4.21) od jej teoretycznego odpowiednika podanego niżej wzorem (4.22) dla wszystkich przeprowadzonych w ramach niniejszej pracy badań eksperymentalnych nie przekroczyło wartości 10^{-2} dB. Warto jeszcze dodać, że teoretyczne wartości wskaźników PSNR względem których mierzone były odchylenia estymatora (4.21) miały, zgodnie z rozważaniami podjętymi wcześniej, następującą postać:

$$PSNR\,(\,\rho_r,\,\rho_c\,) = 10\;log_{\,10}\,\frac{\left(\,4\,\sqrt{\sigma^2}\,\right)^2}{tr\,\left[\,(\,\mathbf{I}-\mathbf{V}\mathbf{I}_m\mathbf{U}\,)\;\mathbf{K}\,(\,\rho_r,\,\rho_c\,)\;(\,\mathbf{I}-\mathbf{V}\mathbf{I}_m\mathbf{U}\,)^T\,\right]}\quad[\,\mathrm{dB}\,]\quad(4.22)$$

gdzie $\sigma^2 = 1$, macierz $\mathbf{K}\,(\,\rho_r,\,\rho_c\,)$ jest $N^2 \times N^2$–elementową teoretyczną macierzą autokowariancji dwuwymiarowego sygnału wejściowego daną zależnością (4.14), $\mathbf{I}_m \in \mathbb{R}^{N^2 \times N^2}$jest macierzą obcinającą dla której przy ustalonym poziomie kompresji $R \in \mathbb{R}$ liczba współczynników przekształceń kodującego/dekodującego $m \in \mathbb{N}$ podlegających kwantyzacji dokładnej wynosi $N^2\,(\,1 - R\,/\,100\,)$, wreszcie $N^2 \times N^2$–elementowe macierze rzeczywiste $\mathbf{U}$ i $\mathbf{V}$ reprezentują zoptymalizowane przekształcenia kodujące/dekodujące właściwe dla wybranej, jednej z czterech, analizowanych podczas badań metod obliczeniowych KLT2D, MLP2D, FCT2D czy NFJT2D.

Przejdźmy teraz do wyjaśnienia metodologii testów efektywnościowych rozważanych metod kompresji. W przypadku testów czasu odpowiedzi porównania efektywnościowe podane w tabelach 4.2a – 4.7a dotyczą łącznych czasów kodowania i dekompresji wszystkich z $10^{\,6}$ próbek obrazowych każdego ze zbiorów testowych dla wybranego przekształcenia. Podobnie, dla sieci neuronowych MLP2D oraz NFJT2D czas nauczania podany we wspomnianych tabelach oznacza całkowity czas pojedynczej epoki treningowej obejmującej prezentację wszystkich z $10^{\,6}$ próbek obrazowych wybranego zbioru uczącego. Przy ustalonym współczynniku kompresji rozważane porównania czasowe są niezależne od wybranego testu, gdyż dotyczą one we wszystkich przypadkach analizowanych przekształceń i sieci jedynie operacji arytmetycznych zaangażowanych bezpośrednio w proces kodowania i dekodowania wspomnianych obrazów (bez uwzględniania czasów takich operacji implementacyjnych jak inkrementacje liczników pętli, instrukcje warunkowe, skoki itp.). Czasy te biorą pod uwagę faktyczną liczbę operacji arytmetycznych zaangażowanych bezpośrednio w obliczenia i są wyznaczone ostatecznie na podstawie odpowiedniego przelicznika [114, 115] właściwego dla procesora firmy Intel, na którym wykonywane były testy. Warto na koniec dodać, że maksymalne współczynniki kompresji w przypadku wszystkich przedstawionych dalej testów (poza szczególnym przypadkiem testów przekształ-

ceń 16×16–punktowych, których cel zostanie wyjaśniony w dyskusji zamieszczonej na końcu niniejszego rozdziału) wybierane były w taki sposób, aby zapewnić w możliwie największym stopniu zrównanie się charakterystyk czasowych procesów wyznaczania odpowiedzi porównywanych ze sobą architektur neuronowych `MLP2D` i `NFJT2D` w trakcie ich działania na etapie operacyjnym, po zakończeniu procesu optymalizacji.

Na koniec warto wspomnieć, że sposób wyboru wyjść w obszarach kwantyzacji przekształceń szybkich, tzn. dla metod `FCT2D` oraz `NFJT2D`, które propagowały swoje sygnały wyjściowe w stronę obszarów dekodujących obydwu wymienionych autokoderów był zgodny z sekwencją ZIG-ZAG. W przypadku sieci neuronowej `NFJT2D` wybór taki towarzyszył zarówno procesowi treningowemu jak i sesji testowej rozważanej sieci. Jak wynika z rozważań podjętych w rozdziale czwartym omawiany tu problem doboru wyjść aktywnych obszaru kwantyzacyjnego dotyczy jedynie wymienionych wyżej metod kompresji z wykorzystaniem struktur szybkich `FCT2D` i `NFJT2D`. W przypadku pozostałych dwóch porównywanych dalej metod kompresji, tzn. autokodera `MLP2D` i jego analitycznego odpowiednika - transformaty `KLT2D` - z racji samej konstrukcji wymienionych wyżej metod sposób ten nie miał żadnego wpływu na metodykę eksperymentów z ich udziałem.

W tym miejscu zakończmy opis metodyki eksperymentu pierwszego i przedstawmy wyniki uzyskane podczas jego realizacji. Wszystkie wspomniane wyniki zebrane są w tabelach 4.2a – 4.7b i zobrazowane na rysunkach 4.4 – 4.9.

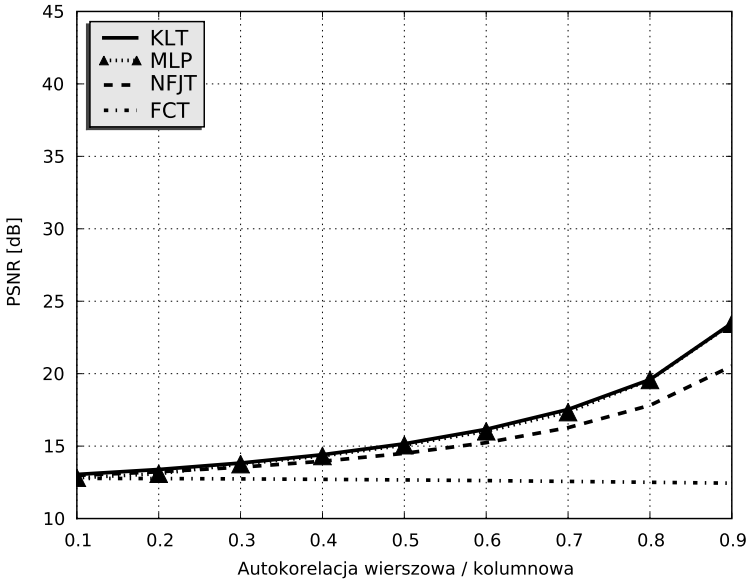
Porównania jakościowo–czasowe, wymiar: 08×08, poziom kompresji: 85%

Tab. 4.2a: Porównania czasowe, wymiar: 08×08, kompresja: 85%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*17.463*	*5.486*
NFJT	*6.546*	*5.331*
FCT	*0.0*	*2.571*

Tab. 4.2b: Porównania jakościowe, wymiar: 08×08, kompresja: 85%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*13.044*	*13.387*	*13.832*	*14.407*	*15.16*	*16.16*	*17.529*	*19.59*	*23.48*
MLP	*12.814*	*13.095*	*13.759*	*14.321*	*15.066*	*16.026*	*17.349*	*19.569*	*23.449*
NFJT	*12.977*	*13.226*	*13.544*	*13.955*	*14.5*	*15.239*	*16.273*	*17.812*	*20.524*
FCT	*12.773*	*12.76*	*12.737*	*12.706*	*12.666*	*12.619*	*12.564*	*12.503*	*12.442*



Rys. 4.4: Porównania jakościowe, wymiar: 08×08, kompresja: 85%

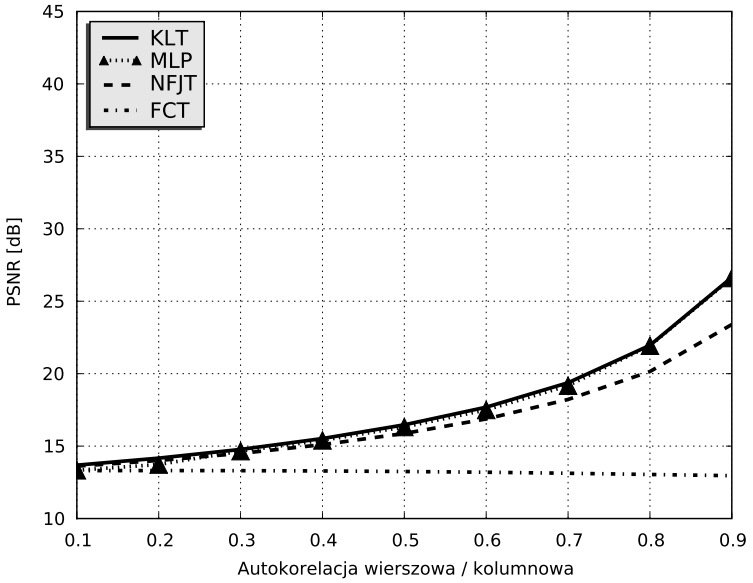
Porównania jakościowo–czasowe, wymiar: 08×08, poziom kompresji: 75%

Tab. 4.3a: Porównania czasowe, wymiar: 08×08, kompresja: 75%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*27.886*	*8.777*
NFJT	*6.546*	*5.331*
FCT	*0.0*	*2.571*

Tab. 4.3b: Porównania jakościowe, wymiar: 08×08, kompresja: 75%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*13.685*	*14.171*	*14.771*	*15.522*	*16.47*	*17.687*	*19.374*	*21.958*	*26.647*
MLP	*13.334*	*13.732*	*14.656*	*15.397*	*16.322*	*17.498*	*19.174*	*21.93*	*26.603*
NFJT	*13.614*	*14.009*	*14.495*	*15.102*	*15.873*	*16.875*	*18.22*	*20.148*	*23.393*
FCT	*13.307*	*13.313*	*13.306*	*13.285*	*13.248*	*13.196*	*13.126*	*13.042*	*12.958*



Rys. 4.5: Porównania jakościowe, wymiar: 08×08, kompresja: 75%

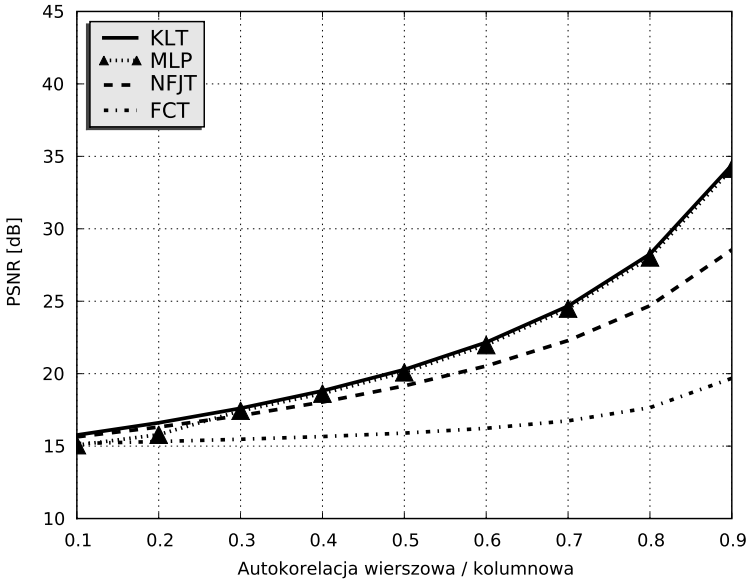
Porównania jakościowo–czasowe, wymiar: 08×08, poziom kompresji: 50%

Tab. 4.4a: Porównania czasowe, wymiar: 08×08, kompresja: 50%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*55.68*	*17.554*
NFJT	*6.546*	*5.331*
FCT	*0.0*	*2.571*

Tab. 4.4b: Porównania jakościowe, wymiar: 08×08, kompresja: 50%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*15.765*	*16.609*	*17.611*	*18.813*	*20.28*	*22.159*	*24.667*	*28.252*	*34.395*
MLP	*15.068*	*15.791*	*17.436*	*18.629*	*20.096*	*21.976*	*24.496*	*28.027*	*34.139*
NFJT	*15.637*	*16.316*	*17.107*	*18.039*	*19.157*	*20.53*	*22.284*	*24.69*	*28.551*
FCT	*15.178*	*15.316*	*15.473*	*15.66*	*15.898*	*16.23*	*16.744*	*17.655*	*19.682*



Rys. 4.6: Porównania jakościowe, wymiar: 08×08, kompresja: 50%

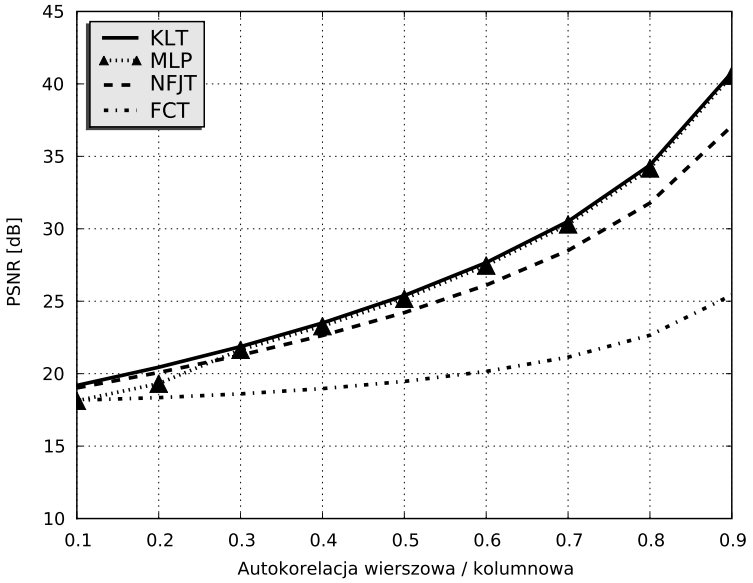
Porównania jakościowo–czasowe, wymiar: 08×08, poziom kompresji: 25%

Tab. 4.5a: Porównania czasowe, wymiar: 08×08, kompresja: 25%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*83.474*	*26.331*
NFJT	*6.546*	*5.331*
FCT	*0.0*	*2.571*

Tab. 4.5b: Porównania jakościowe, wymiar: 08×08, kompresja: 25%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*19.186*	*20.444*	*21.867*	*23.497*	*25.393*	*27.664*	*30.51*	*34.392*	*40.782*
MLP	*18.128*	*19.315*	*21.651*	*23.287*	*25.169*	*27.467*	*30.309*	*34.172*	*40.58*
NFJT	*19.015*	*20.079*	*21.274*	*22.634*	*24.214*	*26.109*	*28.5*	*31.777*	*37.096*
FCT	*18.168*	*18.346*	*18.606*	*18.968*	*19.465*	*20.155*	*21.139*	*22.652*	*25.428*



Rys. 4.7: Porównania jakościowe, wymiar: 08×08, kompresja: 25%

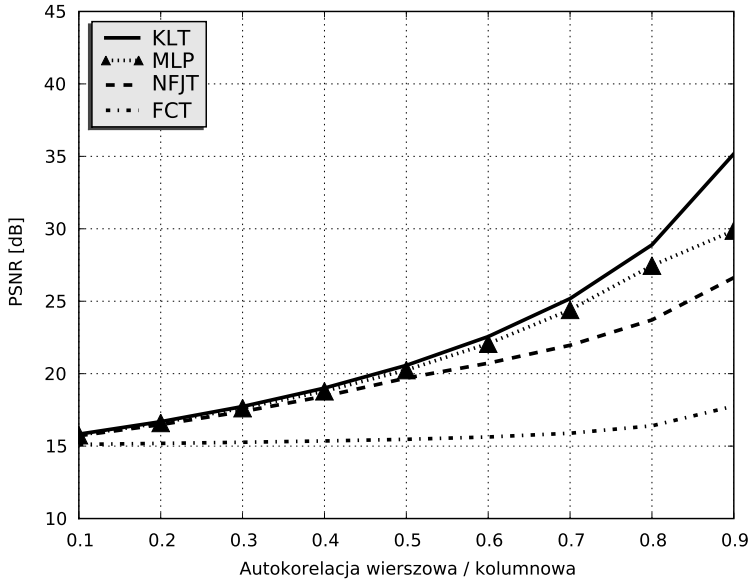
Porównania jakościowo–czasowe, wymiar: 16×16, poziom kompresji: 50%

Tab. 4.6a: Porównania czasowe, wymiar: 16×16, kompresja: 50%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*88.978*	*28.087*
NFJT	*4.715*	*3.002*
FCT	*0.0*	*1.464*

Tab. 4.6b: Porównania jakościowe, wymiar: 16×16, kompresja: 50%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*15.824*	*16.686*	*17.732*	*19.005*	*20.573*	*22.559*	*25.18*	*28.891*	*35.172*
MLP	*15.756*	*16.601*	*17.623*	*18.788*	*20.251*	*22.071*	*24.406*	*27.458*	*29.88*
NFJT	*15.713*	*16.485*	*17.384*	*18.442*	*19.696*	*20.723*	*21.96*	*23.708*	*26.626*
FCT	*15.118*	*15.188*	*15.266*	*15.356*	*15.469*	*15.63*	*15.889*	*16.399*	*17.763*



Rys. 4.8: Porównania jakościowe, wymiar: 16×16, kompresja: 50%

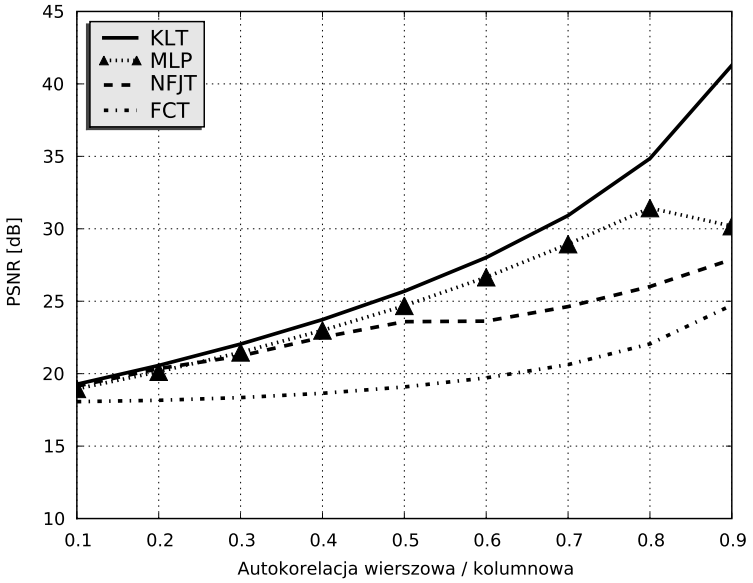
Porównania jakościowo–czasowe, wymiar: 16×16, poziom kompresji: 25%

Tab. 4.7a: Porównania czasowe, wymiar: 16×16, kompresja: 25%

Przekształcenie	*Czas nauczania [s]*	*Czas odpowiedzi [s]*
MLP	*133.449*	*42.13*
NFJT	*5.012*	*3.002*
FCT	*0.0*	*1.464*

Tab. 4.7b: Porównania jakościowe, wymiar: 16×16, kompresja: 25%

PSNR [dB]	***Moduły unormowanych współczynników autokorelacji***								
	0.1	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***
KLT	*19.269*	*20.566*	*22.044*	*23.732*	*25.688*	*28.016*	*30.916*	*34.848*	*41.281*
MLP	*18.952*	*20.136*	*21.472*	*22.987*	*24.682*	*26.659*	*28.942*	*31.432*	*30.185*
NFJT	*19.136*	*20.349*	*21.224*	*22.513*	*23.588*	*23.626*	*24.629*	*26.012*	*27.884*
FCT	*18.068*	*18.161*	*18.349*	*18.646*	*19.083*	*19.71*	*20.625*	*22.055*	*24.726*



Rys. 4.9: Porównania jakościowe, wymiar: 16×16, kompresja: 25%

4.3.2. Wyniki badań dla obrazów rzeczywistych

W niniejszym punkcie opisany zostanie drugi eksperyment główny wykonany w ramach przeprowadzonych badań. W eksperymencie tym do zbadania własności jakościowo efektywnościowych proponowanej architektury sieciowej `NFJT2D` użyto obrazów rzeczywistych, uzyskanych ze zdjęć przedstawiających wybrane przedmioty i postaci. Obrazy eksperymentalne, które służyły za źródło danych wejściowych dla testowanych w eksperymencie drugim technik kompresji dobrane były w taki sposób, aby ich charakterystyki statystyczne przypominały te właściwe dla sygnałów teoretycznych używanych podczas badań realizowanych w ramach eksperymentu pierwszego. Jedyne odstępstwo od wymienionych wyżej reguł doboru obrazów badawczych stanowi para obrazów widniejących na rysunkach 4.11a oraz 4.11b. Zostały one wygenerowane sztucznie za pomocą symulatora pól losowych Gaussa-Markowa przedstawionego w poprzednim rozdziale i stanowią realizacje wspomnianych pól o następujących parametrach statystycznych dotyczących wartości oczekiwanych, odchyleń standardowych oraz współczynników autokorelacji wierszowych i kolumnowych $\mu = 128$, $\sigma = 122.5$, $\rho_r = -0.9$ i $\rho_c = 0.9$ w przypadku obrazu treningowego z rys. 4.11a oraz, odpowiednio: $\mu = 128$, $\sigma = 70.5$, $\rho_r = -0.1$ i $\rho_c = 0.99$ dla obrazu testowego widniejącego na rysunku 4.11b. Jak wspomniano wyżej za wyjątkiem tych obrazów wszystkie pozostałe obrazy eksperymentalne są cyfrowymi reprezentacjami zdjęć przedmiotów i postaci rzeczywistych.

Poza użyciem obrazów rzeczywistych eksperyment drugi niewiele różnił się pod względem swej metodyki od przedstawionego wcześniej eksperymentu `(1)`. Dlatego też niniejsze omówienie dotyczyć będzie jedynie różnic metodycznych pomiędzy wspomnianymi eksperymentami, uznając pozostałe kwestie za wyjaśnione w punkcie 4.3.1. W niniejszym badaniu źródło sygnałów treningowych dla obu rodzajów testowanych sieci neuronowych, tzn. autokoderów `MLP2D` i `NFJT2D`, stanowiły obrazy z rysunków, kolejno 4.11a, 4.14a, 4.17a oraz 4.20a, zaś odpowiadające wyżej wymienionym obrazy, kolejno 4.11b, 4.14b, 4.17b i 4.20b nie brały udziału w procesach optymalizacji rozważanych sieci. Transformata `KLT2D` jako przekształcenie referencyjne do badań jakościowych wyznaczane było natomiast z osobna dla każdego z wymienionych wyżej ośmiu obrazów eksperymentalnych za pomocą algorytmu 4.3, co odzwierciedlają dane zgromadzone w tabelach 4.17b – 4.20b.

Zarówno w procesach optymalizacji sieci neuronowych `MLP2D` oraz `NFJT2D` jak również w przypadku wyznaczania wektorów bazowych przekształcenia `KLT2D` biorące w nich udział obrazy o wymiarach 512×512 w 8–bitowej skali szarości każdy dzielone były wstępnie na bloki 8×8–punktowe stanowiące już bezpośrednią formę sygnału źródłowego odpowiednio dla treningu wspomnianych sieci jak i obliczania przekształceń optymalnych `KLT2D`. W przypadku autokoderów `NFJT2D`, podobnie jak w eksperymencie `(1)`, proces treningowy dotyczył jedynie warstw ostatnich obszarów kodujących, warstw kwantyzacji i pierwszych warstw obszarów dekodujących rozważanych sieci. Współczynniki motylkowe pozostałych warstw rozważanych sieci, nie biorących udziału w procesach treningowych, ustawione były na stałe wartości wynikające z postaci podstawowego algorytmu zunifikowanego (2.46) wyznaczania dwuwymiarowego dyskretnego przekształcenia kosinusowego. W trakcie całego eksperymentu `(2)` zarówno podczas procesów treningowych autokoderów `NFJT2D` jak

i w trakcie sesji testowych z ich udziałem aktywne wyjścia obszarów kodujących rozważanych sieci, propagujące swoje sygnały w kierunku ich obszarów dekompresujących, wybierane były przy ustalonym poziomie kompresji, obliczanym według wzoru (3.12), zgodnie z sekwencją ZIG-ZAG obowiązującą w standardzie JPEG. Taki wybór towarzyszył również wszystkim testom standardowych przekształceń szybkich `FCT2D`.

Inaczej niż w przypadku eksperymentów opisanych wcześniej kryterium stopu obowiązujące podczas sesji treningowych sieci `MLP2D` oraz `NFJT2D` sformułowane było w taki sposób, że nauka wspomnianych sieci przerywana była w momencie gdy zmiana miary PSNR podczas dwóch kolejnych epok treningowych obejmujących losową prezentację 8×8–punktowych obszarów obrazów uczących na wejścia rozważanych autokoderów nie przekraczała wartotści 10^{-3} dB. Przy czym tak jak wcześniej, miara ta wyznaczana była za pomocą wzoru (4.21), tym razem jednak z wartością parametru x_{max} równą liczbie 255 jako maksymalnej możliwej wartości elementów macierzy sygnałów wejściowych, zarówno tych trenujących jak i testowych, stanowiących fragmenty obrazów o 8 bitowej skali szarości. Badania jakościowe czterech porównywanych wzajemnie rodzin metod obliczeniowych, których wyniki zebrane są w tabelach 4.8b – 4.11b, polegały tak jak poprzednio na przeprowadzeniu procesów kompresji i odtwarzania strumienia 8×8–punktowych bloków obrazów testujących z użyciem par zoptymalizowanych wcześniej przekształceń dyskretnych odpowiadających analizowanym algorytmom kalkulacyjnym i wyznaczeniu, w sposób podany wyżej, miar PSNR dla obrazów odtworzonych względem oryginalnych obrazów wejściowych podlegających procesowi kompresji. W badaniu charakterystyk jakościowych rozważanych czterech metod kompresji udział brały wszystkie obrazy, zarówno te treningowe jak i testowe względem procesów optymalizacyjnych sieci neuronowych `MLP2D` i `NFJT2D`, uwidocznione na rysunkach 4.11a, 4.11b 4.14a, 4.14b, 4.17a, 4.17b, 4.20a oraz 4.20b.

Porównania czasowe procesów optymalizacyjnych w przypadku sieci neuronowych obejmują cały opisany wyżej proces treningowy dla wybranego, pojedynczego obrazu uczącego przy zadanym współczynniku kompresji R. W przypadku badań złożoności czasowych etapów operacyjnych przetwarzania sygnałów wejściowych dla analizowanych czterech algorytmów kompresji obrazów wyniki podane niżej w tabelach porównań efektywnościowych 4.8a - 4.11a dotyczą łącznych czasów kodowania i dekompresji pojedynczego obrazu testującego przy pomocy wybranej pary zoptymalizowanych wcześniej przekształceń roboczych właściwych dla badanych tu metod `KLT2D`, `MLP2D`, `FCT2D` oraz `NFJT2D`. Przy ustalonym współczynniku kompresji rozważane porównania czasowe są niezależne od konkretnego testu z przyczyn analogicznych do tych przedstawionych w opisie eksperymentu pierwszego. Również wszystkie pozostałe kwestie dotyczące detali implementacyjnych i metodologicznych badań efektywnościowych rozważanych w niniejszym podrozdziale algorytmów kompresji są identyczne jak w przypadku analogicznych zagadnień w eksperymencie pierwszym i opisane zostały szczegółowo w punkcie 4.3.1. Przejdźmy już teraz do prezentacji wyników badań jakościowo - efektywnościowych uzyskanych po przeprowadzeniu eksperymentu nr `(2)`.

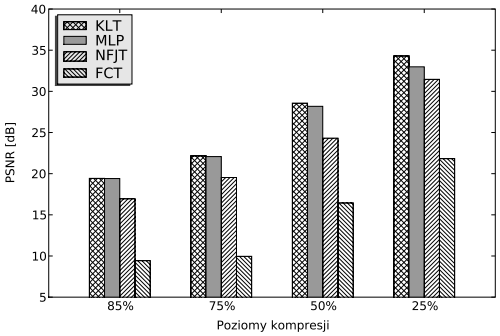
Porównania jakościowo-czasowe dla obrazów 4.11a i 4.11b

Tab. 4.8a: Porównania czasowe badanych przekształceń, obrazy 4.11a i 4.11b

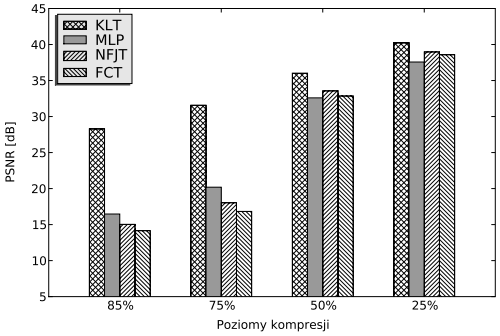
Czasy [s]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
MLP	*3.576*	*0.022*	*5.711*	*0.036*	*11.403*	*0.072*	*17.096*	*0.108*
NFJT	*1.34*	*0.021*	*1.34*	*0.021*	*1.34*	*0.021*	*1.34*	*0.021*
FCT	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*

Tab. 4.8b: Porównania jakościowe badanych przekształceń, obrazy 4.11a i 4.11b

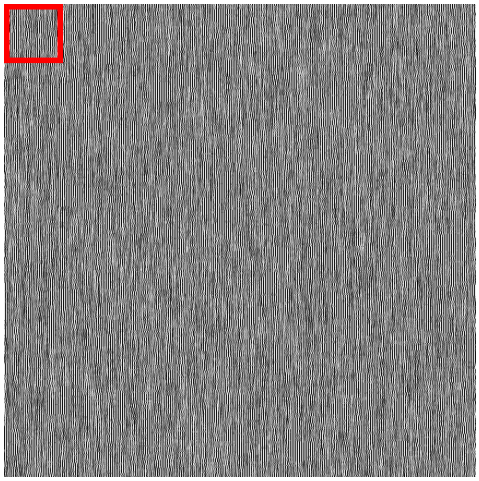
PSNR [dB]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
KLT	*19.430*	*28.288*	*22.167*	*31.583*	*28.571*	*36.026*	*34.301*	*40.239*
MLP	*19.408*	*16.469*	*22.074*	*20.193*	*28.182*	*32.595*	*32.981*	*37.571*
NFJT	*16.952*	*15.036*	*19.534*	*18.033*	*24.320*	*33.566*	*31.462*	*38.972*
FCT	*9.435*	*14.159*	*9.961*	*16.831*	*16.441*	*32.848*	*21.829*	*38.586*



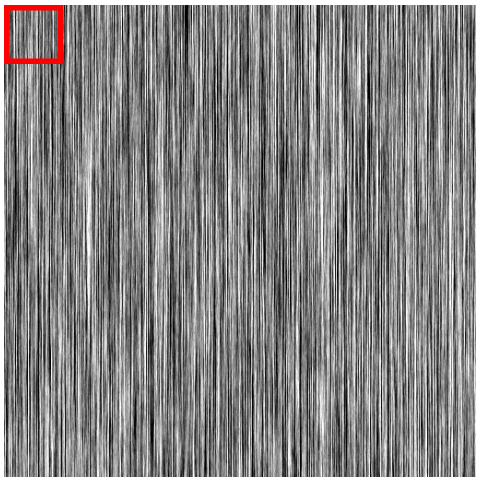
Rys. 4.10a: Wykresy porównań jakościowych dla obrazu uczącego 4.11a



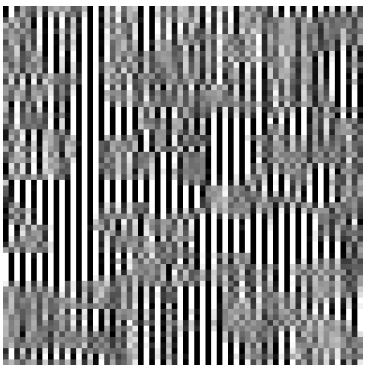
Rys. 4.10b: Wykresy porównań jakościowych dla obrazu testowego 4.11b



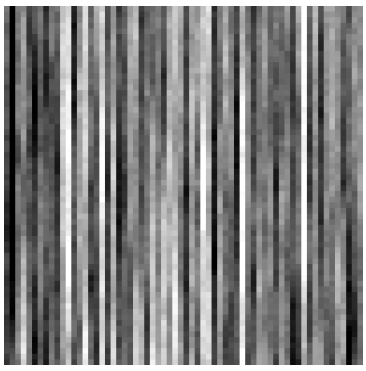
Rys. 4.11a: Obraz uczący (1): 512 ×512 pikseli z fragmentem 64 × 64 piksele



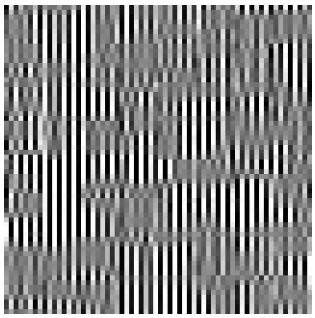
Rys. 4.11b: Obraz testowy (1): 512 ×512 pikseli z fragmentem 64 × 64 piksele



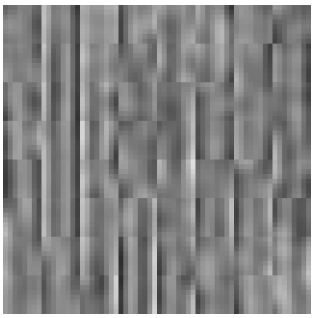
Rys. 4.11c: Fragment obrazu 4.11a



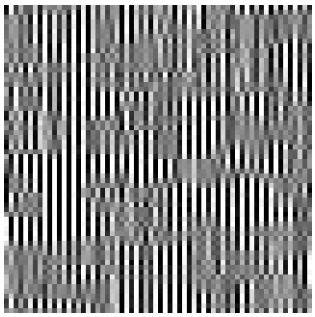
Rys. 4.11d: Fragment obrazu 4.11b



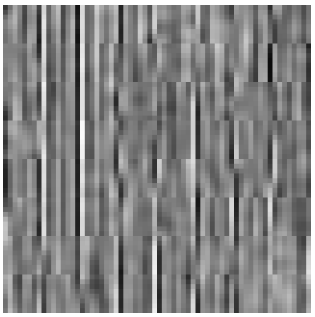
Rys. 4.12(i): KLT, 4.11c, 85%



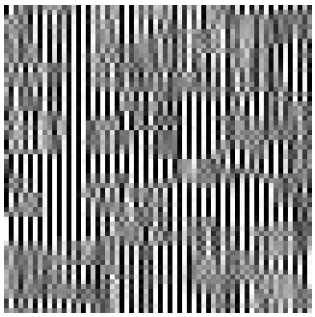
Rys. 4.12(ii): FCT, 4.11c, 85%



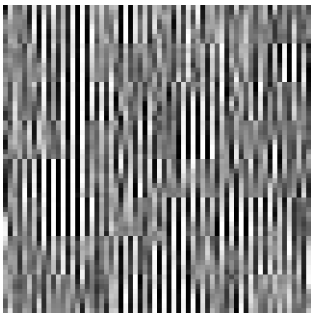
Rys. 4.12(iii): KLT, 4.11c, 75%



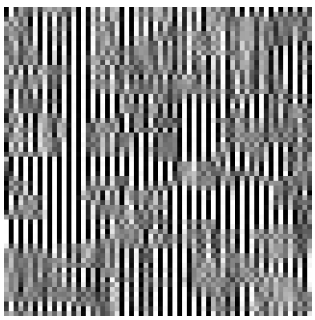
Rys. 4.12(iv): FCT, 4.11c, 75%



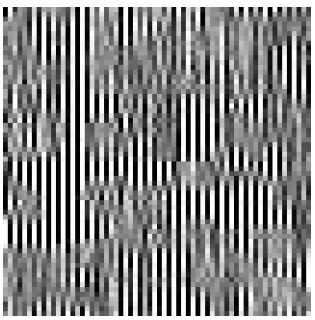
Rys. 4.12(v): KLT, 4.11c, 50%



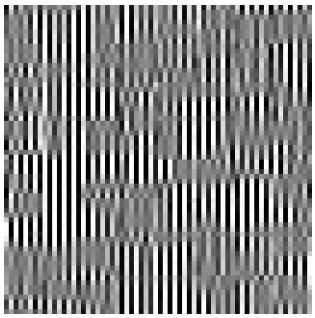
Rys. 4.12(vi): FCT, 4.11c, 50%



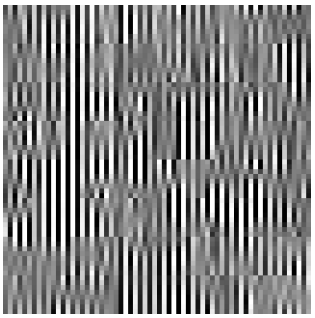
Rys. 4.12(vii): KLT, 4.11c, 25%



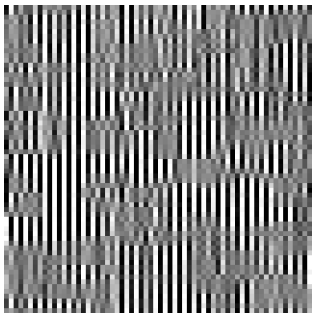
Rys. 4.12(viii): FCT, 4.11c, 25%



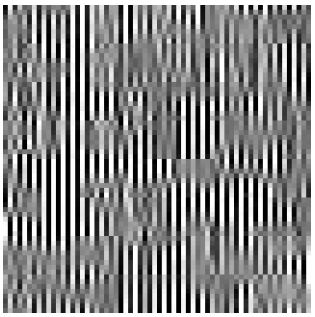
Rys. 4.12(ix): MLP, 4.11c, 85%



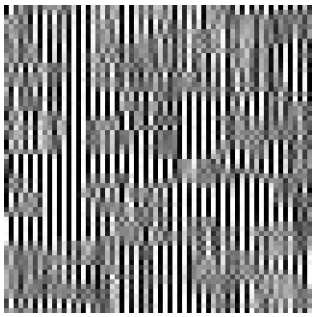
Rys. 4.12(x): NFJT, 4.11c, 85%



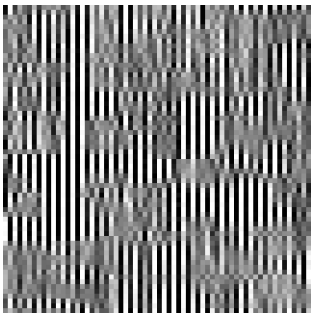
Rys. 4.12(xi): MLP, 4.11c, 75%



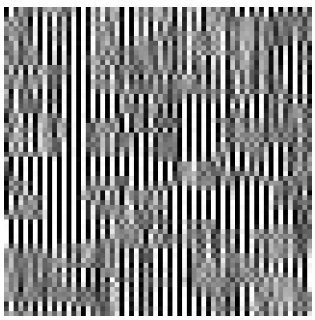
Rys. 4.12(xii): NFJT, 4.11c, 75%



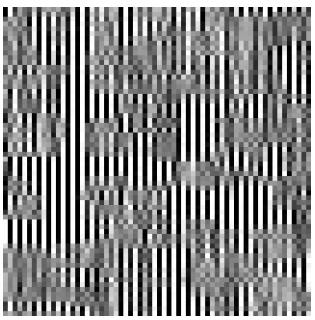
Rys. 4.12(xiii): MLP, 4.11c, 50%



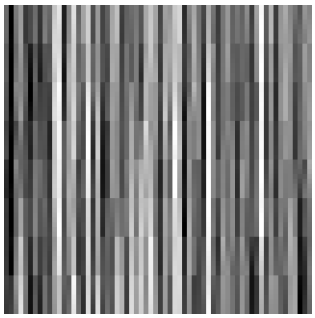
Rys. 4.12(xiv): NFJT, 4.11c, 50%



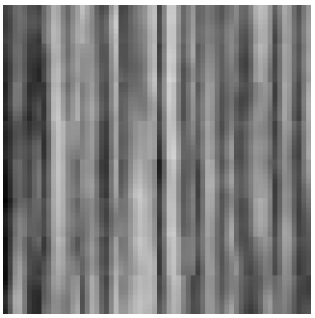
Rys. 4.12(xv): MLP, 4.11c, 25%



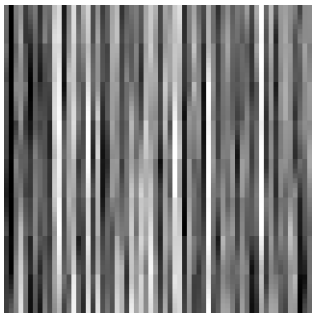
Rys. 4.12(xvi): NFJT, 4.11c, 25%



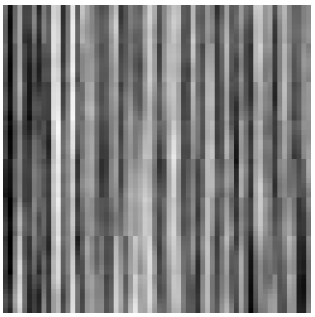
Rys. 4.12(xvii): KLT, 4.11d, 85%



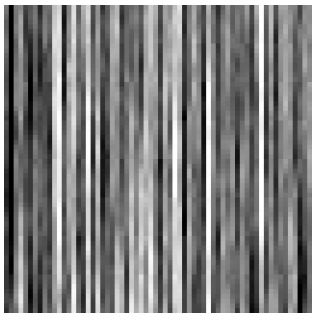
Rys. 4.12(xviii): FCT, 4.11d, 85%



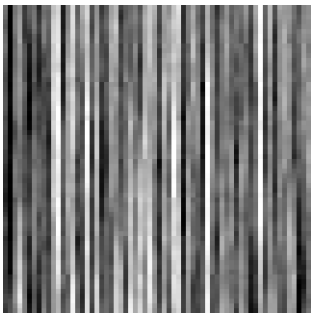
Rys. 4.12(xix): KLT, 4.11d, 75%



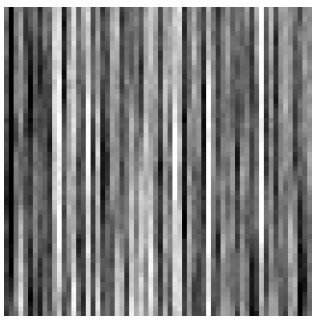
Rys. 4.12(xx): FCT, 4.11d, 75%



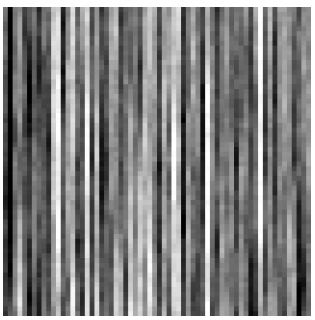
Rys. 4.12(xxi): KLT, 4.11d, 50%



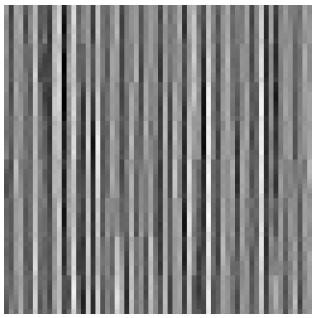
Rys. 4.12(xxii): FCT, 4.11d, 50%



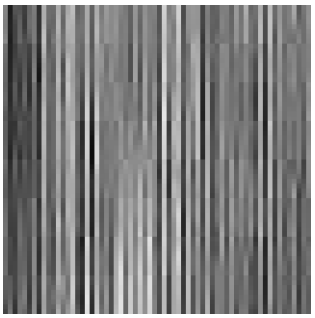
Rys. 4.12(xxiii): KLT, 4.11d, 25%



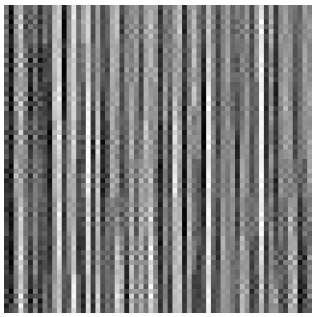
Rys. 4.12(xxiv): FCT, 4.11d, 25%



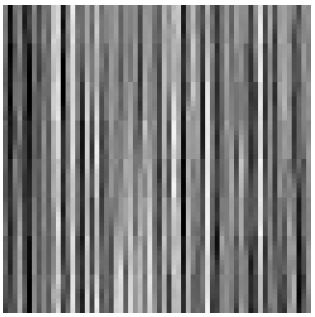
Rys. 4.12(xxv): MLP, 4.11d, 85%



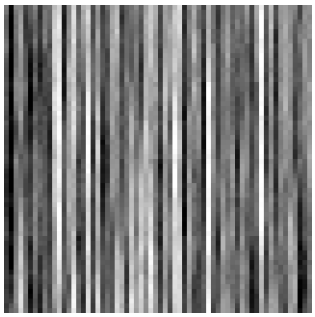
Rys. 4.12(xxvi): NFJT, 4.11d, 85%



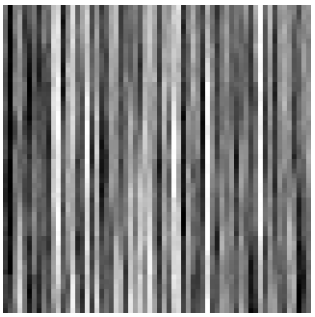
Rys. 4.12(xxvii): MLP, 4.11d, 75%



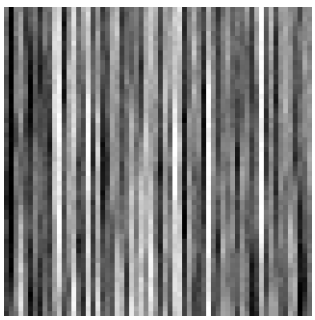
Rys. 4.12(xxviii): NFJT, 4.11d, 75%



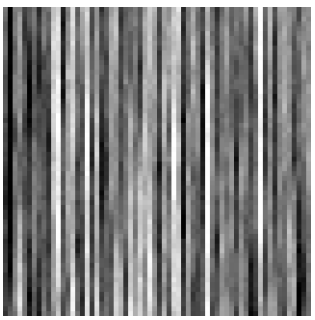
Rys. 4.12(xxix): MLP, 4.11d, 50%



Rys. 4.12(xxx): NFJT, 4.11d, 50%



Rys. 4.12(xxxi): MLP, 4.11d, 25%



Rys. 4.12(xxxii): NFJT, 4.11d, 25%

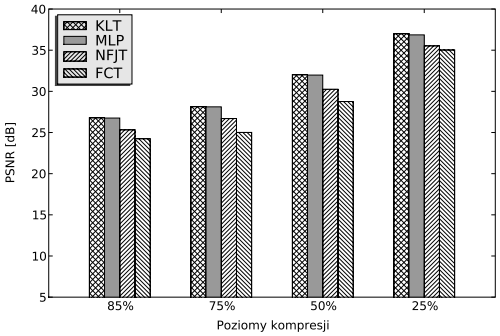
Porównania jakościowo-czasowe dla obrazów 4.14a i 4.14b

Tab. 4.9a: Porównania czasowe badanych przekształceń, obrazy 4.14a i 4.14b

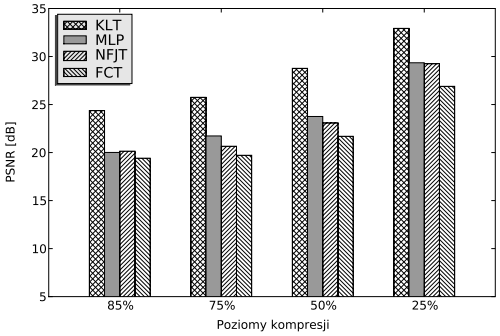
Czasy [s]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
MLP	*3.576*	*0.022*	*5.711*	*0.036*	*11.403*	*0.072*	*17.096*	*0.108*
NFJT	*1.34*	*0.021*	*1.34*	*0.021*	*1.34*	*0.021*	*1.34*	*0.021*
FCT	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*

Tab. 4.9b: Porównania jakościowe badanych przekształceń, obrazy 4.14a i 4.14b

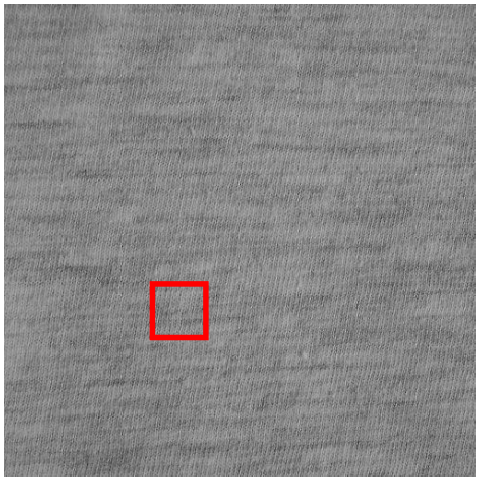
PSNR [dB]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
KLT	*26.796*	*24.374*	*28.146*	*25.753*	*32.025*	*28.770*	*37.009*	*32.931*
MLP	*26.757*	*20.013*	*28.120*	*21.731*	*31.978*	*23.757*	*36.862*	*29.350*
NFJT	*25.329*	*20.146*	*26.698*	*20.658*	*30.262*	*23.084*	*35.528*	*29.255*
FCT	*24.247*	*19.415*	*25.012*	*19.717*	*28.765*	*21.695*	*35.022*	*26.896*



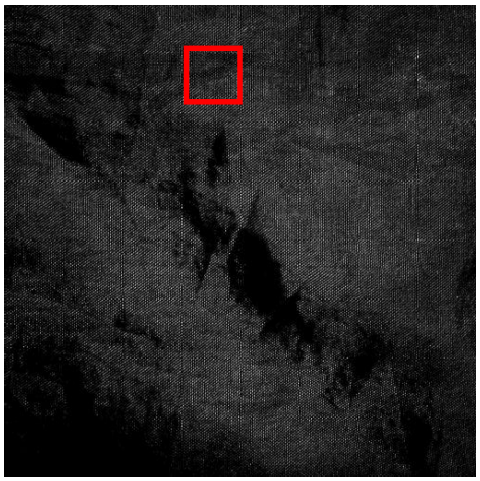
Rys. 4.13a: Wykresy porównań jakościowych dla obrazu uczącego 4.14a



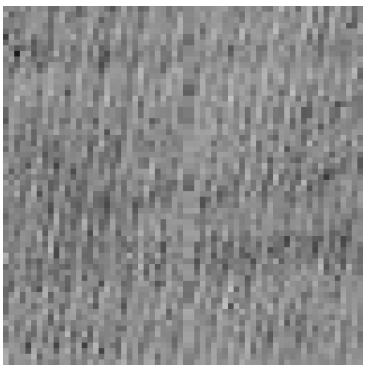
Rys. 4.13b: Wykresy porównań jakościowych dla obrazu testowego 4.14b



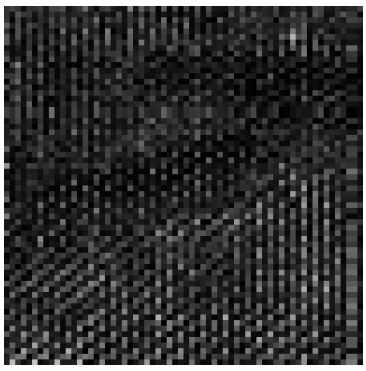
Rys. 4.14a: Obraz uczący (2): 512 ×512 pikseli z fragmentem 64 × 64 piksele



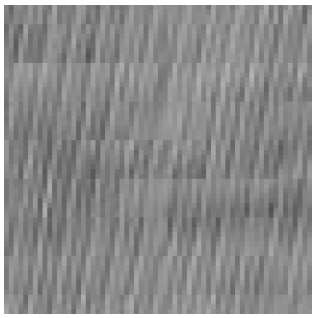
Rys. 4.14b: Obraz testowy (2): 512 ×512 pikseli z fragmentem 64 × 64 piksele



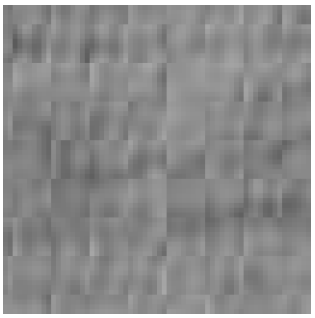
Rys. 4.14c: Fragment obrazu 4.14a



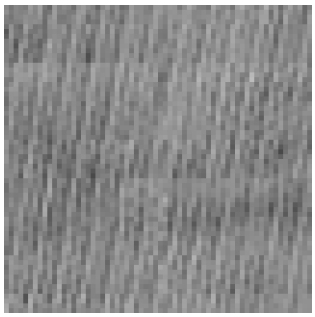
Rys. 4.14d: Fragment obrazu 4.14b



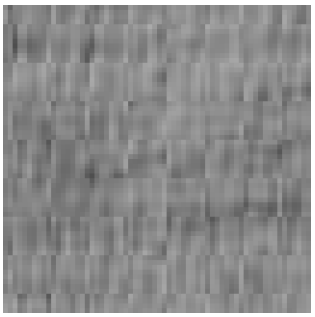
Rys. 4.15(i): KLT, 4.14c, 85%



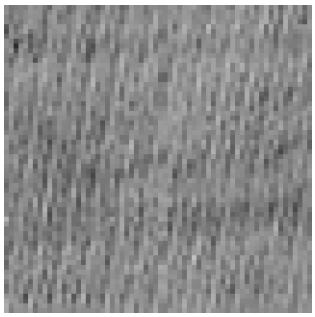
Rys. 4.15(ii): FCT, 4.14c, 85%



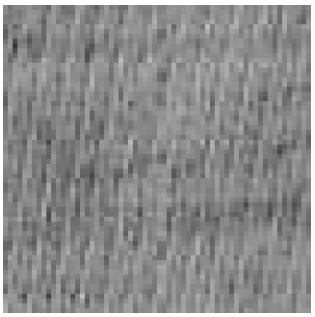
Rys. 4.15(iii): KLT, 4.14c, 75%



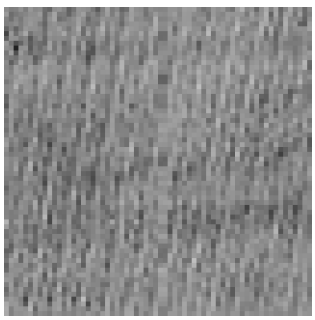
Rys. 4.15(iv): FCT, 4.14c, 75%



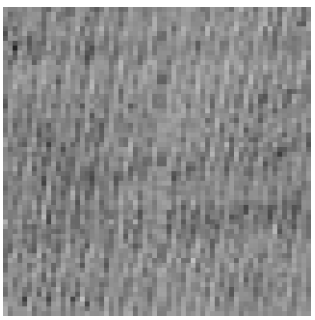
Rys. 4.15(v): KLT, 4.14c, 50%



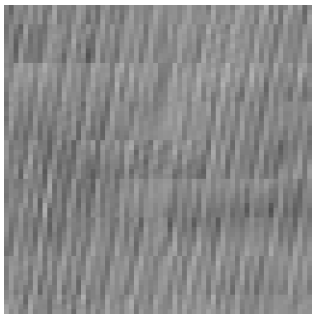
Rys. 4.15(vi): FCT, 4.14c, 50%



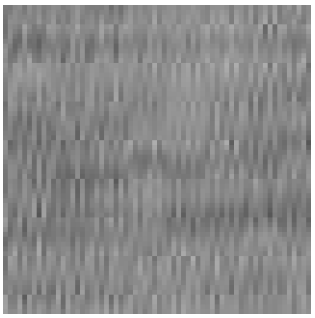
Rys. 4.15(vii): KLT, 4.14c, 25%



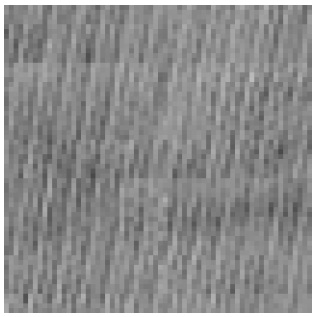
Rys. 4.15(viii): FCT, 4.14c, 25%



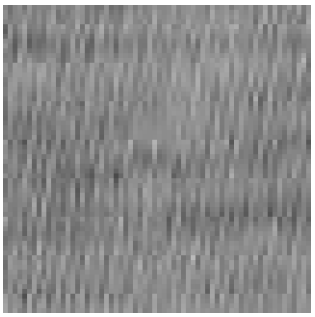
Rys. 4.15(ix): MLP, 4.14c, 85%



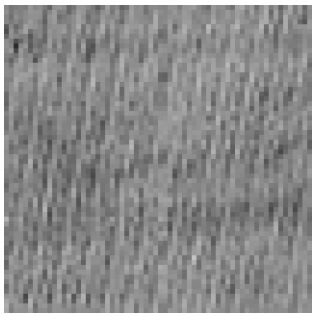
Rys. 4.15(x): NFJT, 4.14c, 85%



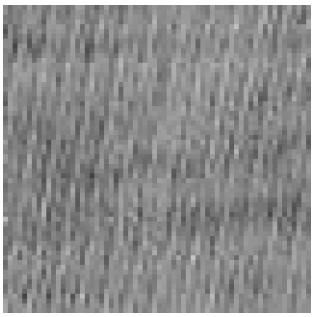
Rys. 4.15(xi): MLP, 4.14c, 75%



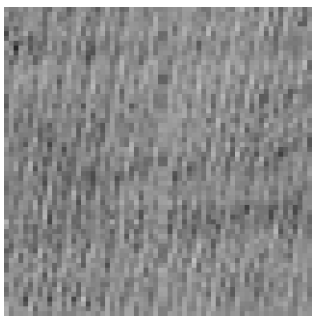
Rys. 4.15(xii): NFJT, 4.14c, 75%



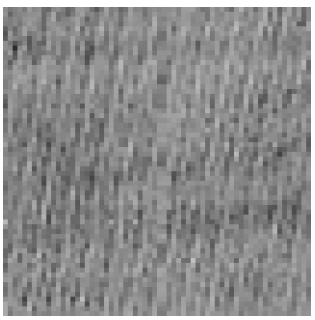
Rys. 4.15(xiii): MLP, 4.14c, 50%



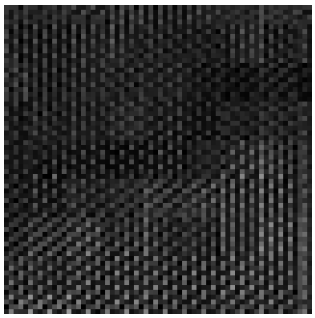
Rys. 4.15(xiv): NFJT, 4.14c, 50%



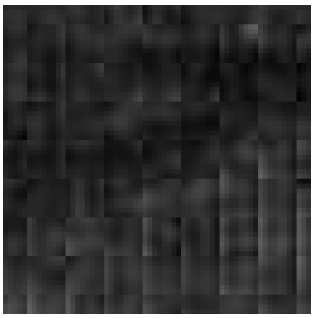
Rys. 4.15(xv): MLP, 4.14c, 25%



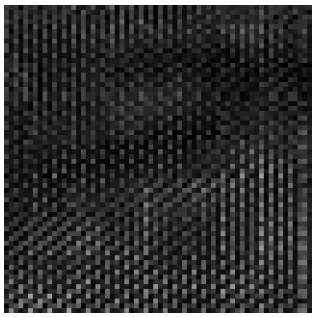
Rys. 4.15(xvi): NFJT, 4.14c, 25%



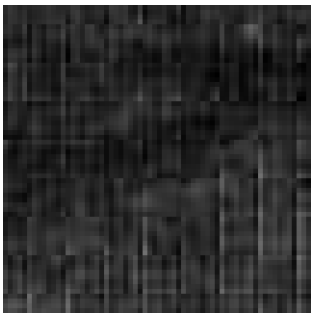
Rys. 4.15(xvii): KLT, 4.14d, 85%



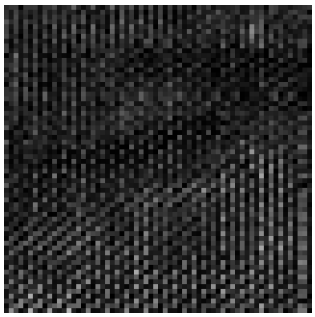
Rys. 4.15(xviii): FCT, 4.14d, 85%



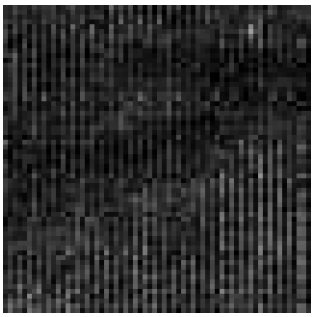
Rys. 4.15(xix): KLT, 4.14d, 75%



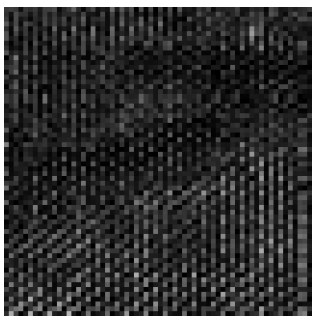
Rys. 4.15(xx): FCT, 4.14d, 75%



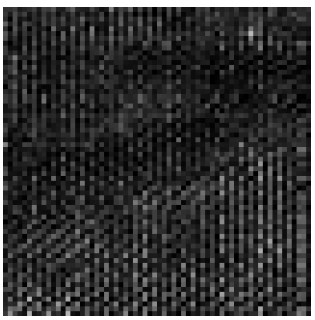
Rys. 4.15(xxi): KLT, 4.14d, 50%



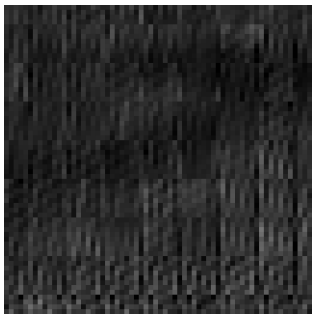
Rys. 4.15(xxii): FCT, 4.14d, 50%



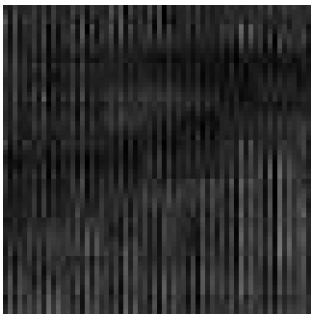
Rys. 4.15(xxiii): KLT, 4.14d, 25%



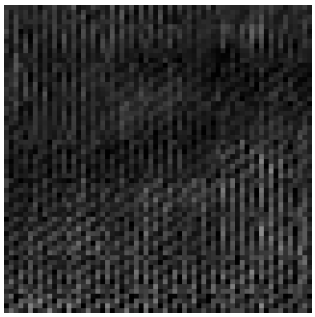
Rys. 4.15(xxiv): FCT, 4.14d, 25%



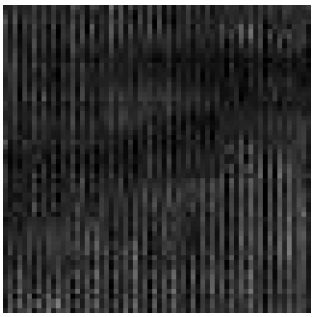
Rys. 4.15(xxv): MLP, 4.14d, 85%



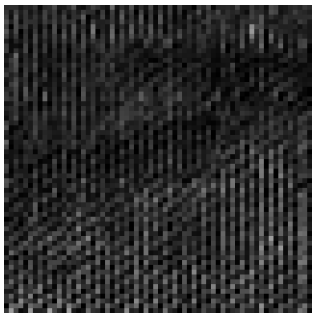
Rys. 4.15(xxvi): NFJT, 4.14d, 85%



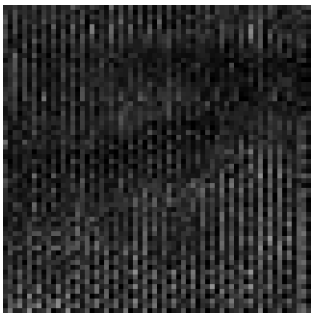
Rys. 4.15(xxvii): MLP, 4.14d, 75%



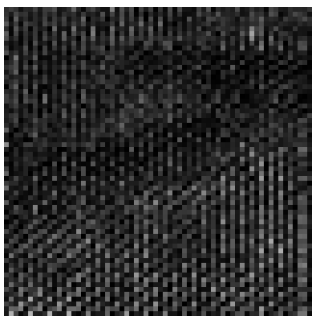
Rys. 4.15(xxviii): NFJT, 4.14d, 75%



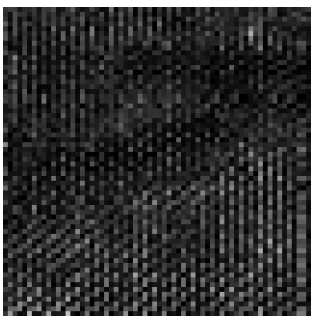
Rys. 4.15(xxix): MLP, 4.14d, 50%



Rys. 4.15(xxx): NFJT, 4.14d, 50%



Rys. 4.15(xxxi): MLP, 4.14d, 25%



Rys. 4.15(xxxii): NFJT, 4.14d, 25%

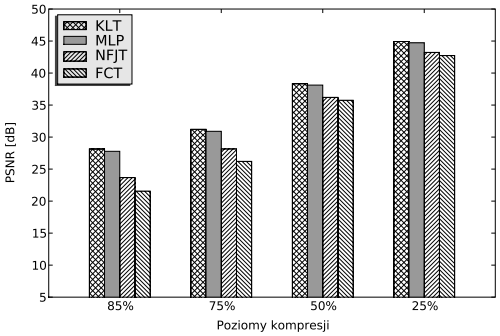
Porównania jakościowo-czasowe dla obrazów 4.17a i 4.17b

Tab. 4.10a: Porównania czasowe badanych przekształceń, obrazy 4.17a i 4.17b

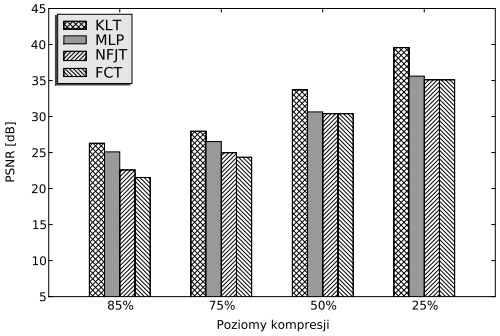
Czasy [s]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
MLP	*7.153*	*0.022*	*5.711*	*0.036*	*68.42*	*0.072*	*136.764*	*0.108*
NFJT	*2.681*	*0.021*	*1.34*	*0.021*	*8.043*	*0.021*	*10.725*	*0.021*
FCT	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*

Tab. 4.10b: Porównania jakościowe badanych przekształceń, obrazy 4.17a i 4.17b

PSNR [dB]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
KLT	*28.138*	*26.310*	*31.224*	*27.986*	*38.339*	*33.730*	*44.929*	*39.588*
MLP	*27.779*	*25.097*	*30.904*	*26.525*	*38.122*	*30.646*	*44.724*	*35.616*
NFJT	*23.678*	*22.581*	*28.151*	*24.985*	*36.205*	*30.4*	*43.227*	*35.108*
FCT	*21.568*	*21.544*	*26.209*	*24.369*	*35.754*	*30.4*	*42.733*	*35.105*

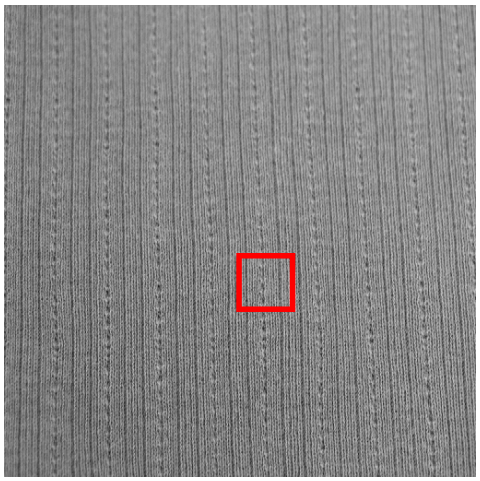


Rys. 4.16a: Wykresy porównań jakościowych dla obrazu uczącego 4.17a

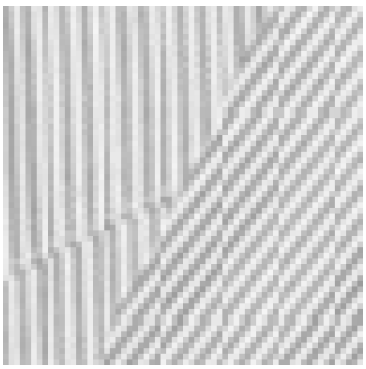


Rys. 4.16b: Wykresy porównań jakościowych dla obrazu testowego 4.17b

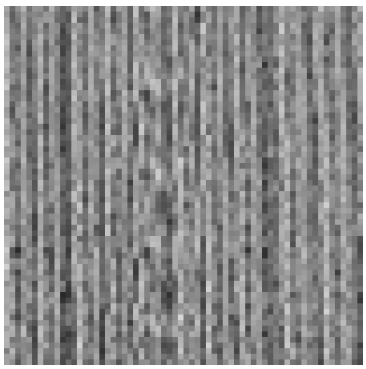
Rys. 4.17a: Obraz uczący (3): 512 ×512 pikseli z fragmentem 64 × 64 piksele



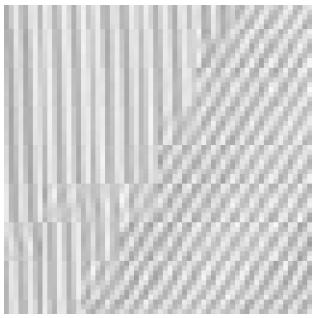
Rys. 4.17b: Obraz testowy (3): 512 ×512 pikseli z fragmentem 64 × 64 piksele



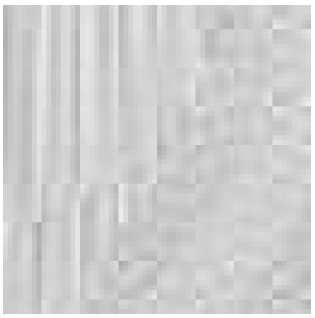
Rys. 4.17c: Fragment obrazu 4.17a



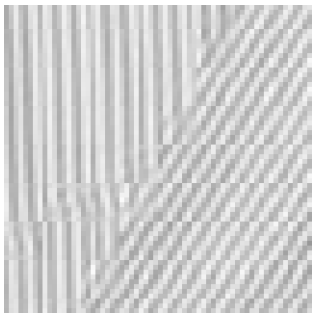
Rys. 4.17d: Fragment obrazu 4.17b



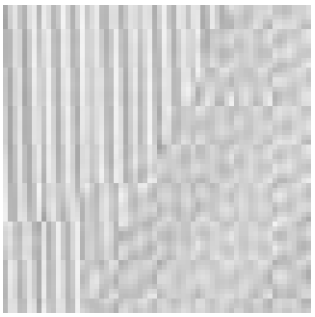
Rys. 4.18(i): KLT, 4.17c, 85%



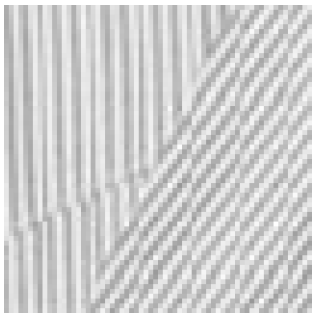
Rys. 4.18(ii): FCT, 4.17c, 85%



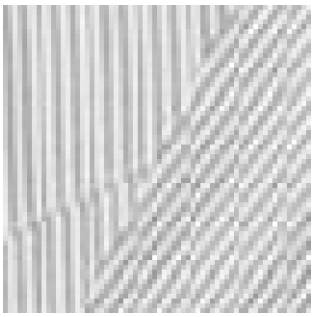
Rys. 4.18(iii): KLT, 4.17c, 75%



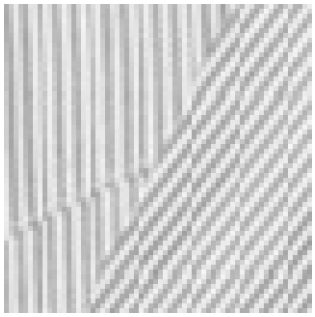
Rys. 4.18(iv): FCT, 4.17c, 75%



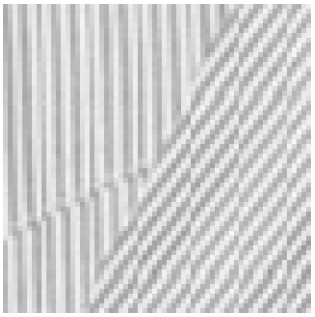
Rys. 4.18(v): KLT, 4.17c, 50%



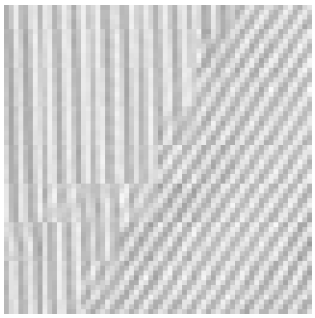
Rys. 4.18(vi): FCT, 4.17c, 50%



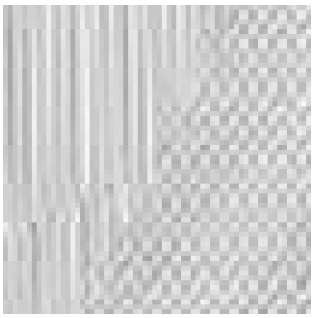
Rys. 4.18(vii): KLT, 4.17c, 25%



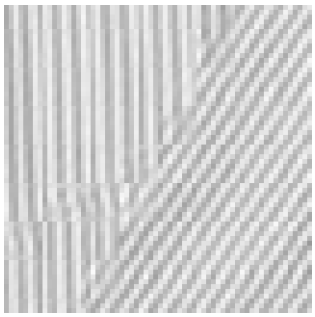
Rys. 4.18(viii): FCT, 4.17c, 25%



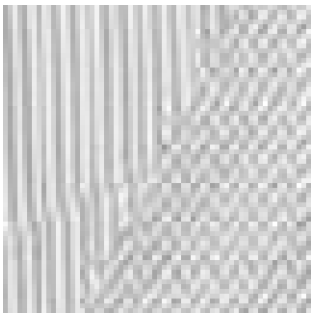
Rys. 4.18(ix): MLP, 4.17c, 85%



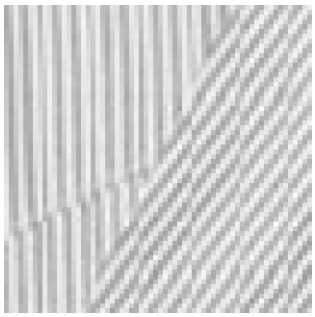
Rys. 4.18(x): NFJT, 4.17c, 85%



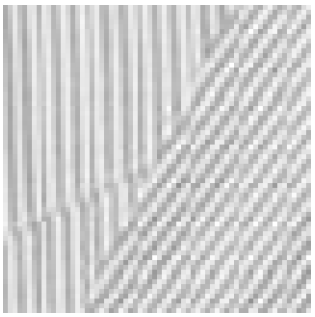
Rys. 4.18(xi): MLP, 4.17c, 75%



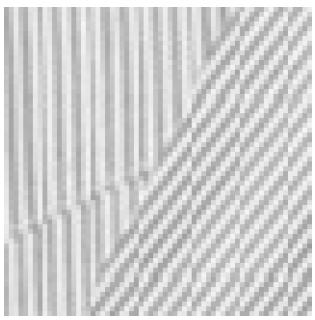
Rys. 4.18(xii): NFJT, 4.17c, 75%



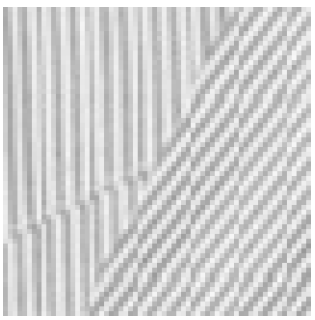
Rys. 4.18(xiii): MLP, 4.17c, 50%



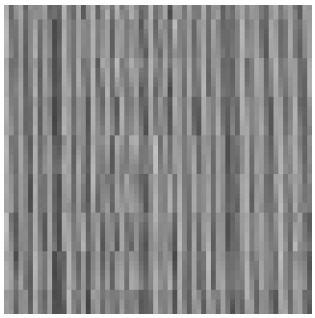
Rys. 4.18(xiv): NFJT, 4.17c, 50%



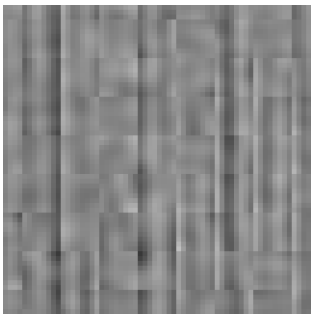
Rys. 4.18(xv): MLP, 4.17c, 25%



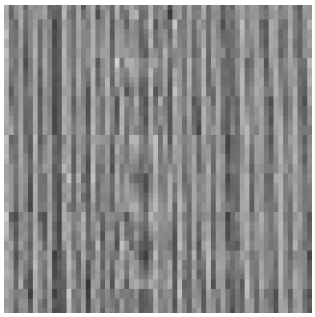
Rys. 4.18(xvi): NFJT, 4.17c, 25%



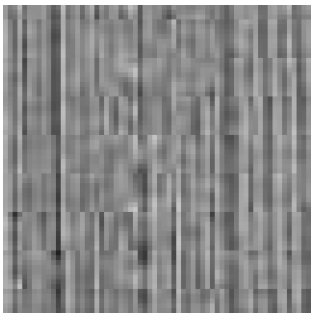
Rys. 4.18(xvii): KLT, 4.17d, 85%



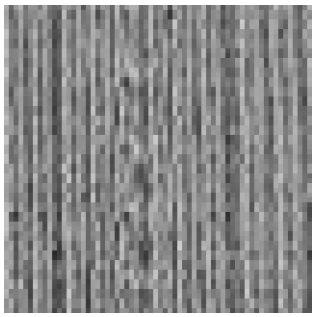
Rys. 4.18(xviii): FCT, 4.17d, 85%



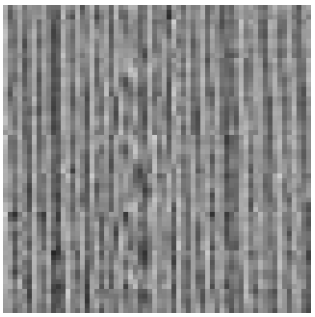
Rys. 4.18(xix): KLT, 4.17d, 75%



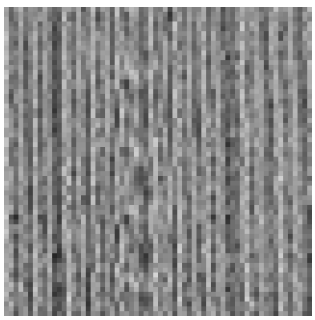
Rys. 4.18(xx): FCT, 4.17d, 75%



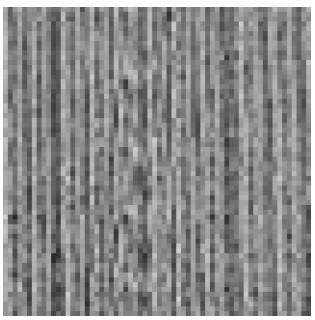
Rys. 4.18(xxi): KLT, 4.17d, 50%



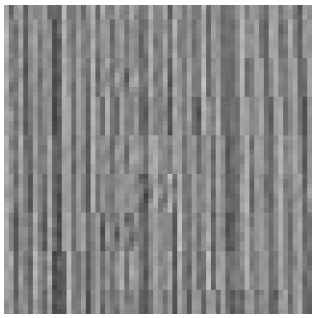
Rys. 4.18(xxii): FCT, 4.17d, 50%



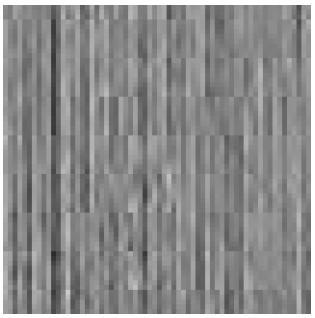
Rys. 4.18(xxiii): KLT, 4.17d, 25%



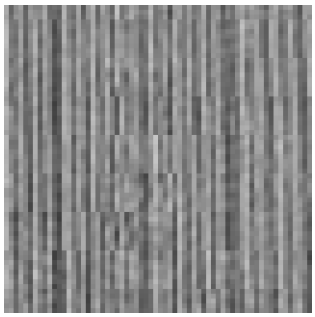
Rys. 4.18(xxiv): FCT, 4.17d, 25%



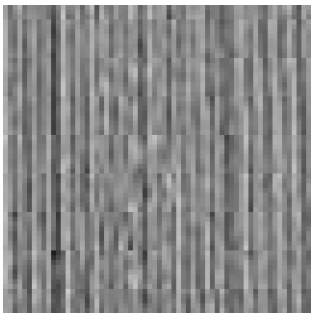
Rys. 4.18(xxv): MLP, 4.17d, 85%



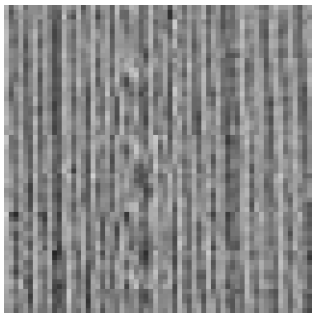
Rys. 4.18(xxvi): NFJT, 4.17d, 85%



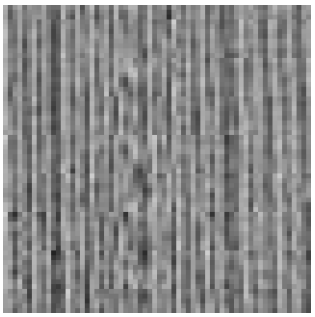
Rys. 4.18(xxvii): MLP, 4.17d, 75%



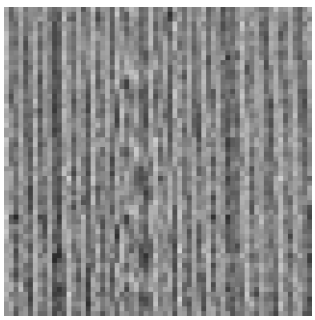
Rys. 4.18(xxviii): NFJT, 4.17d, 75%



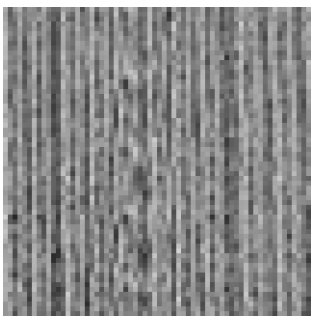
Rys. 4.18(xxix): MLP, 4.17d, 50%



Rys. 4.18(xxx): NFJT, 4.17d, 50%



Rys. 4.18(xxxi): MLP, 4.17d, 25%



Rys. 4.18(xxxii): NFJT, 4.17d, 25%

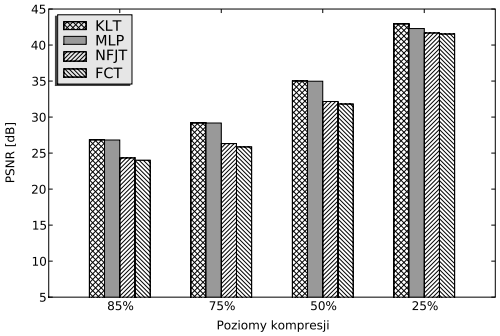
Porównania jakościowo-czasowe dla obrazów 4.20a i 4.20b

Tab. 4.11a: Porównania czasowe badanych przekształceń, obrazy 4.20a i 4.20b

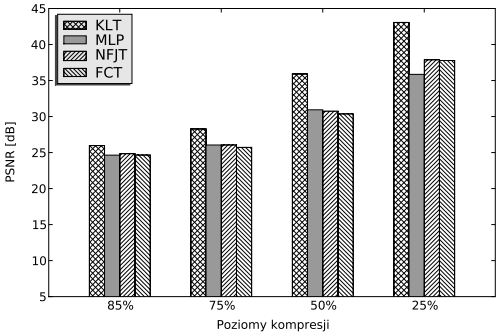
Czasy [s]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
MLP	*21.458*	*0.022*	*34.266*	*0.036*	*68.42*	*0.072*	*102.573*	*0.108*
NFJT	*8.043*	*0.021*	*8.043*	*0.021*	*8.043*	*0.021*	*8.043*	*0.021*
FCT	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*	*0.0*	*0.011*

Tab. 4.11b: Porównania jakościowe badanych przekształceń, obrazy 4.20a i 4.20b

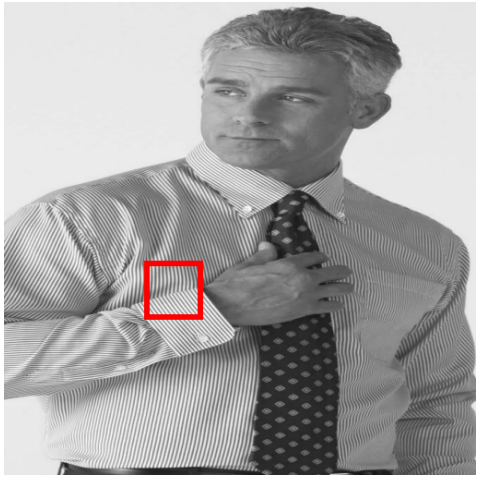
PSNR [dB]	***Wartości poziomów kompresji dla rozważanych obrazów***							
	85%		***75%***		***50%***		***25%***	
	nauka	***test***	***nauka***	***test***	***nauka***	***test***	***nauka***	***test***
KLT	*26.847*	*25.974*	*29.197*	*28.283*	*35.016*	*35.915*	*42.945*	*43.067*
MLP	*26.818*	*24.644*	*29.177*	*26.051*	*34.980*	*30.938*	*42.289*	*35.858*
NFJT	*24.333*	*24.848*	*26.322*	*26.056*	*32.166*	*30.759*	*41.676*	*37.884*
FCT	*24.023*	*24.655*	*25.859*	*25.722*	*31.812*	*30.371*	*41.532*	*37.781*



Rys. 4.19a: Wykresy porównań jakościowych dla obrazu uczącego 4.20a

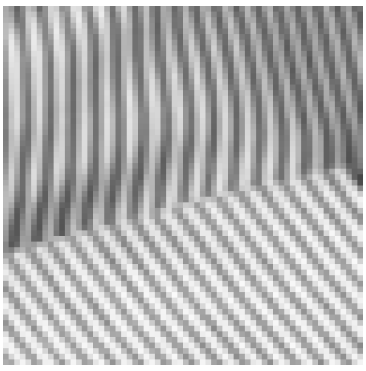


Rys. 4.19b: Wykresy porównań jakościowych dla obrazu testowego 4.20b

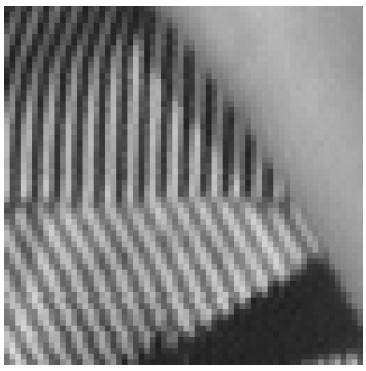


Rys. 4.20a: Obraz uczący (4): 512 ×512 pikseli z fragmentem 64 × 64 piksele

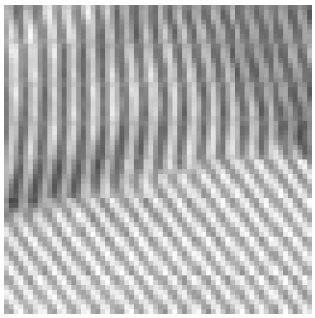
Rys. 4.20b: Obraz testowy (4): 512 ×512 pikseli z fragmentem 64 × 64 piksele



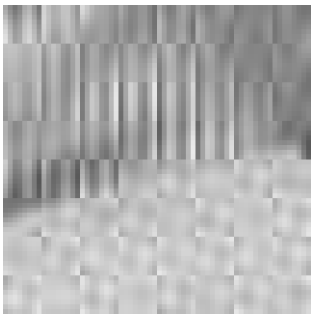
Rys. 4.20c: Fragment obrazu 4.20a



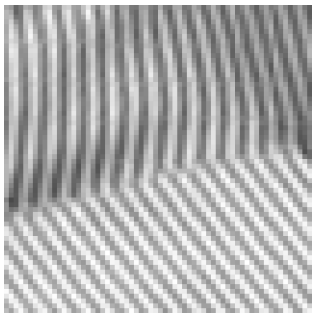
Rys. 4.20d: Fragment obrazu 4.20b



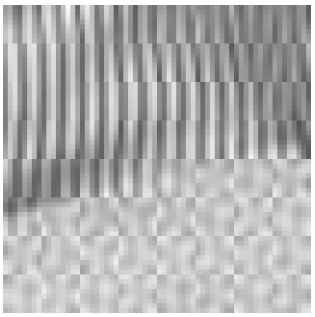
Rys. 4.21(i): KLT, 4.20c, 85%



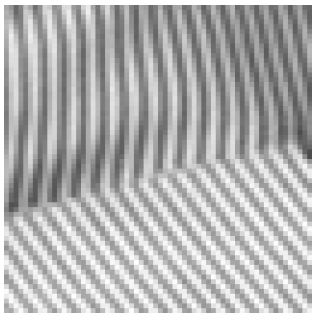
Rys. 4.21(ii): FCT, 4.20c, 85%



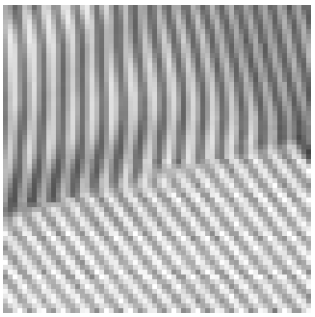
Rys. 4.21(iii): KLT, 4.20c, 75%



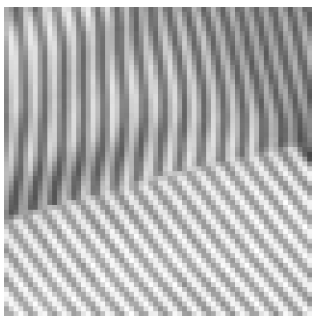
Rys. 4.21(iv): FCT, 4.20c, 75%



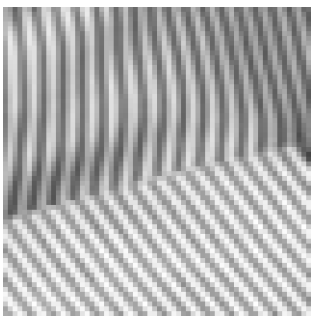
Rys. 4.21(v): KLT, 4.20c, 50%



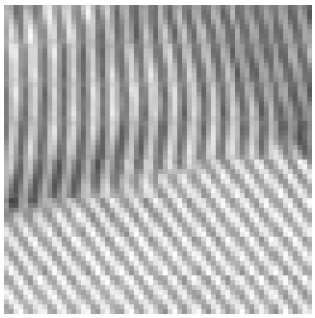
Rys. 4.21(vi): FCT, 4.20c, 50%



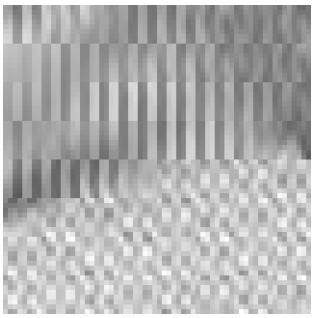
Rys. 4.21(vii): KLT, 4.20c, 25%



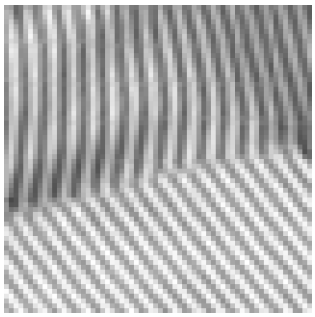
Rys. 4.21(viii): FCT, 4.20c, 25%



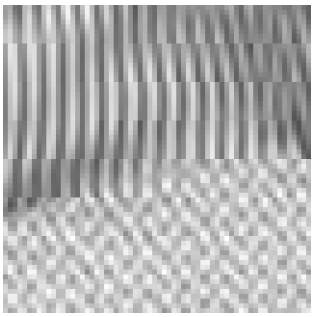
Rys. 4.21(ix): MLP, 4.20c, 85%



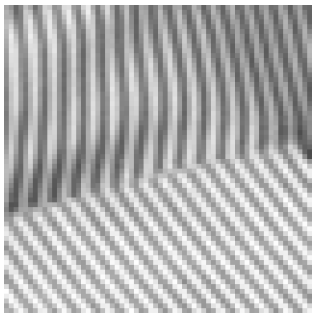
Rys. 4.21(x): NFJT, 4.20c, 85%



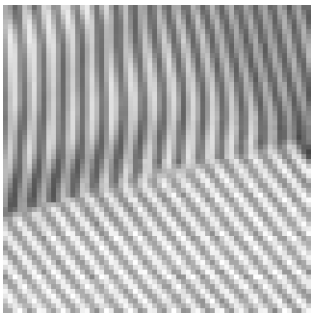
Rys. 4.21(xi): MLP, 4.20c, 75%



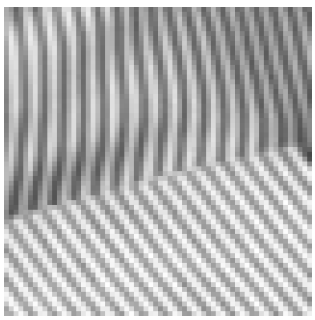
Rys. 4.21(xii): NFJT, 4.20c, 75%



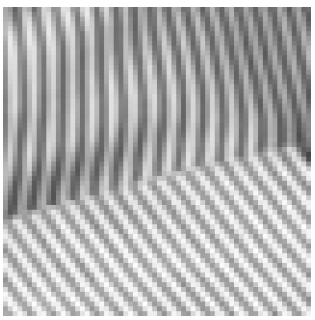
Rys. 4.21(xiii): MLP, 4.20c, 50%



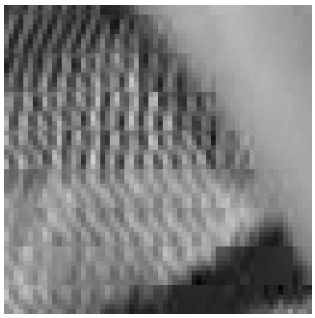
Rys. 4.21(xiv): NFJT, 4.20c, 50%



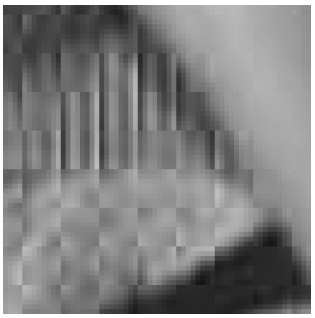
Rys. 4.21(xv): MLP, 4.20c, 25%



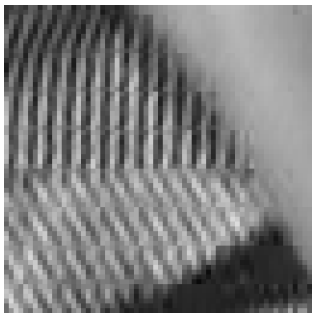
Rys. 4.21(xvi): NFJT, 4.20c, 25%



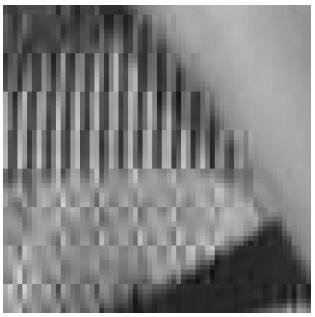
Rys. 4.21(xvii): KLT, 4.20d, 85%



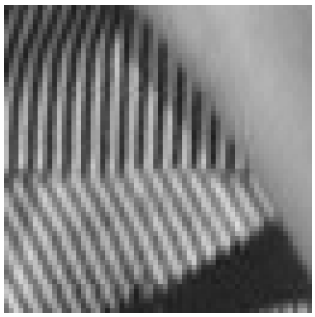
Rys. 4.21(xviii): FCT, 4.20d, 85%



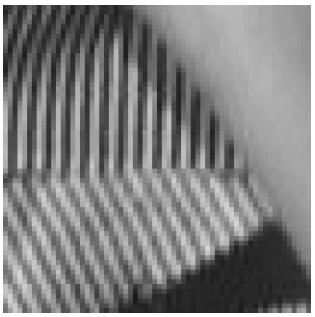
Rys. 4.21(xix): KLT, 4.20d, 75%



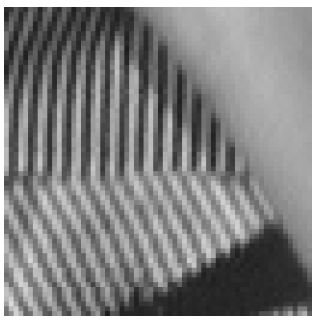
Rys. 4.21(xx): FCT, 4.20d, 75%



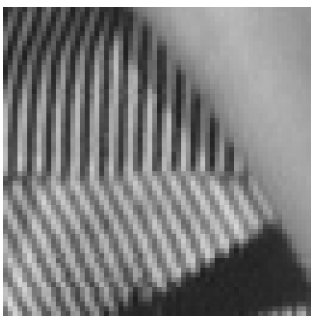
Rys. 4.21(xxi): KLT, 4.20d, 50%



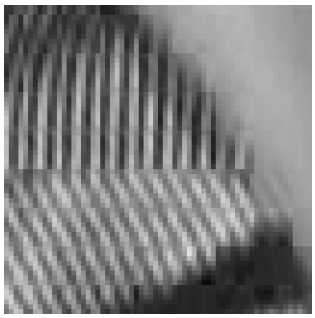
Rys. 4.21(xxii): FCT, 4.20d, 50%



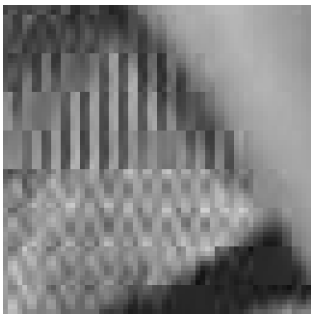
Rys. 4.21(xxiii): KLT, 4.20d, 25%



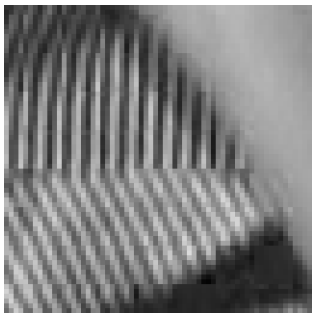
Rys. 4.21(xxiv): FCT, 4.20d, 25%



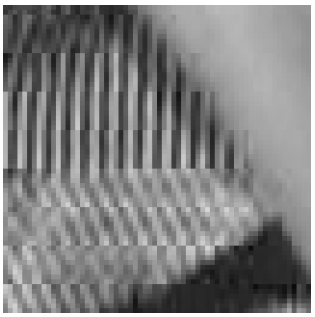
Rys. 4.21(xxv): MLP, 4.20d, 85%



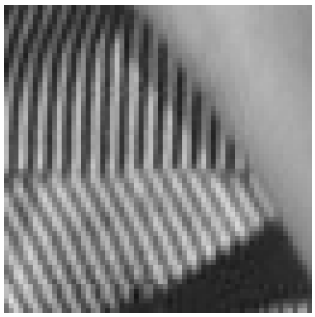
Rys. 4.21(xxvi): NFJT, 4.20d, 85%



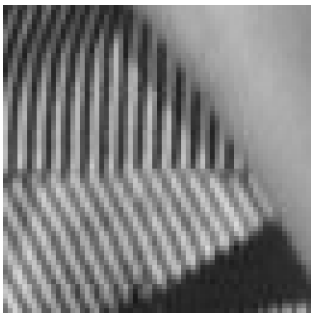
Rys. 4.21(xxvii): MLP, 4.20d, 75%



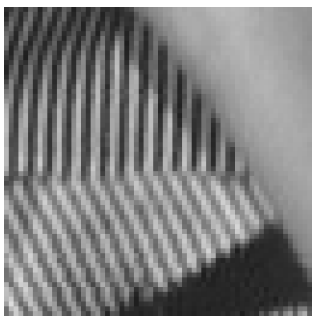
Rys. 4.21(xxviii): NFJT, 4.20d, 75%



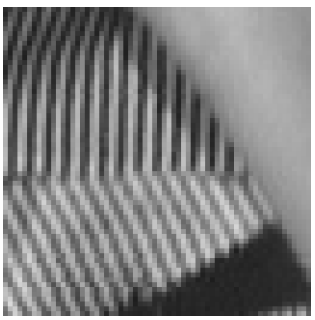
Rys. 4.21(xxix): MLP, 4.20d, 50%



Rys. 4.21(xxx): NFJT, 4.20d, 50%



Rys. 4.21(xxxi): MLP, 4.20d, 25%



Rys. 4.21(xxxii): NFJT, 4.20d, 25%

4.4. Wnioski

W niniejszej części opracowania przedstawiona zostanie dyskusja dotycząca wyników badań eksperymentalnych nad jakościowo - efektywościowymi charakterystykami zaprezentowanej w niniejszej monografii architektury neuronowej do kompresji obrazów uzyskanych podczas realizacji dwóch eksperymentów głównych opisanych wyżej w podrozdziałach 4.3.1 oraz 4.3.2.

Warto najpierw poddać analizie wyniki eksperymentu pierwszego. Jak pokazują tabele porównań jakościowych 4.2b – 4.5b i odpowiadające im rysunki 4.4 – 4.7, dla metod kompresji o standardowych parametrach, tzn. wymiarach przekształceń dyskretnych równych 8×8 punktom oraz przy kwantyzacji dokładnej odpowiadającej sekwencji ZIG-ZAG, w przypadku sygnałów dwuwymiarowych o charakterystyce probabilistycznej opisanej w punkcie 4.3.1 wszystkie rozważane porównania plasują metodę kompresji z wykorzystaniem proponowanej architektury neuronowej `NFJT2D` pomiędzy metodą standardową, z użyciem transformaty kosinusowej `FCT2D`, która wypada zdecydowanie najgorzej z jakościowego punktu widzenia, a metodą adaptacyjną `MLP2D` z wykorzystaniem perceptronu liniowego, której wyniki zbliżają się do wartości optymalnych osiąganych przez dyskretną transformatę Karhunena-Loève'go `KLT2D`.

Warto jednak zauważyć, że o ile różnica jakościowa pomiędzy metodami adaptacyjnymi `NFJT2D` oraz `MLP2D` jest stosunkowo niewielka, o tyle standardowa procedura kompresji stosowana w metodzie JPEG wykorzystująca przekształcenie `FCT2D` wydaje się być zupełnie nieprzystosowaną do kompresji sygnałów o dobranej w eksperymencie pierwszym charakterystyce statystycznej. Zwróćmy chociażby uwagę na kilka z najbardziej skrajnych przykładów opisanej wyżej relacji pomiędzy porównywanymi metodami. Na przykład dla dość typowych z punktu widzenia użytkownika systemu kompresji parametrach testowanych metod, tzn. przy wysokich modułach korelacji wierszowej/kolumnowej sygnałów wejściowych $|\rho_r| = |\rho_c| = 0.9$ oraz stosunkowo dużym współczynniku kompresji $R = 75\%$ widać w tabeli 4.3b i na odpowiadającym jej rysunku 4.5, że różnica pomiędzy metodami adaptacyjnymi `MLP2D` i `NFJT2D` jest stosunkowo niewielka i wynosi około 3.2 dB, natomiast analogiczna relacja między proponowaną metodą adaptacyjną `NFJT2D` a standardową procedurą kompresji z użyciem przekształcenia `FCT2D` jest już bardzo duża i wynosi około 10.4 dB na korzyść proponowanej architektury neuronowej.

Podobnie przedstawia się sytuacja w przypadku testu jakościowego, którego wyniki zawarte są w tabeli 4.4b i zaprezentowane na rysunku 4.6. Tutaj przy takich samych jak poprzednio parametrach sygnału wejściowego i współczynniku kompresji $R = 50\%$ różnica jakościowa pomiędzy metodami adaptacyjnymi `MLP2D` i `NFJT2D` wynosi około 5.6 dB na korzyść tej pierwszej, zaś porównanie wyników metod `NFJT2D` oraz `FCT2D` ujawnia fakt istotnej przewagi proponowanej architektury nad metodą standardową, gdyż analogiczna relacja pomiędzy wymienionymi wyżej metodami wynosi w tym przypadku 8.7 dB na korzyść architektury `NFJT2D`. Tego typu relacje, choć nie w tak skrajnych proporcjach, zachowane są we wszystkich porównaniach jakościowych eksperymentu `(1)`. Warto w tym miejscu dodać, że zgodnie z ogólnie

przyjętym przez środowisko ekspertów dziedziny kompresji obrazów wnioskiem,[1] dolna granica różnicy jakościowej rekonstruowanych w procesie dekodowania skompresowanych wcześniej obrazów przy której zmiany w dekodowanym obrazie w stosunku do oryginału zaczynają być zauważalne przez człowieka określona jest na poziomie około 0.5 dB. Wniosek ten jeszcze bardziej zdaje się uwypuklać przewagę jakościową algorytmu adaptacyjnego NFJT2D nad jego standardowym odpowiednikiem FCT2D stanowiącym główny komponent metody kompresji JPEG, gdyż w obydwu rozważanych wyżej przypadkach testów jakościowych wspomniana granica została przekroczona ponad 20–krotnie w pierwszym przykładzie i około 18–krotnie w drugim z analizowanych wyżej przykładów na korzyść metody adaptacyjnej NFJT2D.

Zajmijmy się teraz kwestią wyników efektywnościowych porównywanych ze sobą metod kompresji dla eksperymentu (1). Tutaj relacja pomiędzy dwoma skrajnymi jakościowo przekształceniami MLP2D oraz FCT2D odwraca się na korzyść metody standardowej FCT2D z jednoczesnym zachowaniem centralnej pozycji proponowanej architektury neuronowej do kompresji obrazów NFJT2D względem wyników czasowych uzyskiwanych w przeprowadzonych testach, zebranych w tabelach 4.2a 4.5a. Relacja taka utrzymuje się na przestrzeni wszystkich testów efektywnościowych przeprowadzonych podczas eksperymentu (1) zarówno w przypadku czasów procesów optymalizacyjnych dla przekształceń adaptacyjnych (w tym także w przypadku metody FCT2D dla której przyjęto umowny, zerowy czas procesu optymalizacji) jak i efektywności etapów operacyjnych rozważanych metod kompresji, obejmujących czasy procesów kompresji i dekodowania sygnałów wejściowych podczas ich późniejszego funkcjonowania. Podobnie jak wcześniej prześledźmy dwa przykłady będące wymownymi ilustracjami przedstawionej wyżej tezy.

Z wyników zawartych w tabeli 4.4a dla poziomu kompresji $R = 50\%$ można wnioskować, że czas treningu sieci NFJT2D i wynosi 6.55 sek. i jest około 8.5 raza mniejszy niż odpowiadający mu czas optymalizacji standardowej sieci MLP2D stosowanej w adaptacyjnej kompresji obrazów, który kształtuje się na poziome około 55.7 sekund. Czasy kodowania/dekodowania wszystkich trzech analizowanych metod kompresji są w tym przypadku równe wartościom 17.6, 5.3 oraz 2.5 sekund, odpowiednio dla algorytmów MLP2D, NFJT2D oraz FCT2D. Oznacza to, że czas procesu kompresji i dekodowania sygnału dla proponowanej architektury adaptacyjnej NFJT2D jest w tym przypadku około 3 razy krótszy niż dla sieci MLP2D i z drugiej strony około 2 razy dłuższy w stosunku do efektywności standardowej metody FCT2D. Relacja ta pogłębia się na korzyść proponowanej architektury neuronowej w przypadku testu z poziomem kompresji równym 25%, którego wyniki przedstawione są w tabeli 4.5a. Tutaj czas treningu sieci NFJT2D jest około 13 razy krótszy od odpowiadającego mu czasu dla jej standardowego odpowiednika MLP2D. Efektywności procesów kodowania i dekompresji przetwarzanych danych obrazowych również świadczą jeszcze dobitniej na korzyść metody NFJT2D względem standardowej sieci neuronowej MLP2D, gdyż w przypadku tym czas odpowiedzi pierwszej z wymienionych przed chwilą sieci jest około 5 razy krótszy od analogicznego czasu raportowanego dla drugiej z nich. Tutaj NFJT2D jest znów około 2 razy wolniejszy w procesie kompresji/dekodowania sygnałów wejściowych od rozwiązania FCT2D stosowanego w metodzie JPEG.

[1] patrz chociażby uwagi zawarte w książce [112]

Kończąc opis eksperymentu (1) warto jeszcze na moment zwrócić uwagę na ciekawy przypadek pary badań eksperymentalnych dla przekształceń 16×16–punktowych, których wyniki jakościowe zawarte są w tabelach 4.6b oraz 4.7b i uwidocznione odpowiednio na rysunkach 4.8 oraz 4.9, zaś dane z ich testów efektywnościowych zebrane są w tabelach 4.6a oraz 4.7a. Badania te sugestywnie uwidaczniają przypuszczalny koniec możliwości efektywnego treningu autokodera perceptronowego MLP2D za pomocą metody wstecznej propagacji błędu. Jest to szczególnie widoczne na rysunku 4.9 dla poziomu kompresji $R = 25\%$, gdzie wykres jakościowy metody MLP2D załamuje się wyraźnie w kierunku niższych wartości miary PSNR dla odtwarzanych obrazów na poziomie modułu współczynników autokorelacji wierszowej/kolumnowej sygnału wejściowego równym 0.9. Efekt ten, choć nie tak wyraźny, zaczyna być już widoczny w przypadku badania przekształcenia 16×16–punktowego przy poziomie kompresji równym 50%, co pokazują dane z tabeli 4.6b oraz wykresy charakterystyk efektywnościowych przedstawione na rysunku 4.8. Pomimo szeregu powtórzeń obydwu badań z różnymi ustawieniami parametrów procesu nauki perceptron MLP2D przez cały czas wykazywał analogiczne zachowanie. Zasugerowany wyżej wniosek o kresie możliwości efektywnego treningu autokodera MLP2D na poziomie wymiaru optymalizowanych przekształceń równym 16×16 punktom wynika z doświadczeń autora wyniesionych z prac nad optymalizacją przekształceń jedno oraz dwuwymiarowych zarówno metodami szybkimi jak i standardowymi, których wyniki nie są jednak częścią niniejszej publikacji.

Kończąc ten wątek warto jeszcze podkreślić, iż na podstawie wspomnianych przed chwilą doświadczeń autor czuje się uprawniony do stwierdzenia, iż o ile skuteczność procesu treningowego autokodera perceptronowego spada drastycznie już dla przedstawionych tu przekształceń 16×16–punktowych, o tyle sieć neuronowa NFJT2D jest w stanie bez większego trudu zrealizować z powodzeniem skuteczny trening dla przekształceń 32×32–punktowych, a nawet przekształceń o wyższych wymiarach, choć w przypadku tych ostatnich porces treningowy jest już stosunkowo powolny. Wnioski te zostały potwierdzone przez autora w sposób empiryczny na podstawie wspomnianych wyżej eksperymentów pomocniczych.

Ostatnią kwestią na którą warto zwrócić uwagę przy omawianiu rozpatrywanych tu przykładów przekształceń 16×16–punktowych jest skala przyrostu stosunku czasów treningu i późniejszego wyznaczania odpowiedzi sieci MLP2D oraz NFJT2D względem odpowiedniej relacji dla wymienionych architektur neuronowych realizujących przekształcenia 8×8–punktowe. Weźmy znów pod uwagę przypadek skrajny, z danymi eksperymentalnymi zawartymi w tabeli 4.7a, przekształcenia 16×16 punktowego przy stopniu kompresji $R = 25\%$. Proces treningu autokodera NFJT2D jest tutaj niemalże 30 razy krótszy od czasu optymalizacji sieci standardowej MLP2D, co stanowi prawie trzykrotny przyrost tej relacji na korzyść architektury NFJT2D względem odpowiedniego stosunku czasów dla sieci 8×8–punktowych równego, zgodnie z wcześniejszymi spostrzeżeniami, 13–krotności. Podobnie, stosunek czasów wyznaczania odpowiedzi zoptymalizowanych wcześniej sieci, zgodnie z danymi zawartymi w tabeli 4.7a, wynosi w przypadku przekształceń 16×16–punktowych 15–krotności, podczas gdy analogiczna relacja dla sieci 8×8–punktowych kształtuje się na poziomie równym 5–krotności, co oznacza że i w tym przypadku przyrost rozważanych wartości jest

trzykrotny również z korzyścią dla autokodera NFJT2D. Powyższe spostrzeżenia, wraz z podanymi wcześniej uwagami na temat skuteczności procesów treningowych rozpatrywanych architektur neuronowych do kompresji obrazów mogą być interpretowane jako empiryczna manifestacja teoretycznej różnicy rzędów złożoności obliczeniowych proponowanej architektury sieciowej NFJT2D i jej standardowego odpowiednika w postaci autokodera perceptronowego MLP2D na rzecz pierwszej z wymienionych sieci. Powyższe spostrzeżenia i wnioski kończą analizę wyników eksperymentu (1).

W odróżnieniu od teoretycznego charakteru eksperymentu pierwszego, eksperyment drugi miał na celu weryfikację efektywności ogólnej proponowanej architektury neuronowej w warunkach jak najbardziej przypominających te spotykane najczęściej w praktyce zadania kompresji adaptacyjnej, w której przetwarzaniu podlegają obrazy rzeczywiste, będące cyfrowymi reprezentacjami zdjęć wybranych, realnych obiektów czy postaci. Niemniej ważnym celem uzupełniającym eksperymentu (2) było ewentualne potwierdzenie praktyczne dość satysfakcjonujących rezultatów teoretycznych symulacji procesu kompresji dla obrazów modelowych, uzyskanych w eksperymencie (1) i opisanych w poprzednim paragrafie. Postawione w ten sposób cele oraz chęć zapewnienia przejrzystości i spójności całości badań eksperymentalnych nad proponowanym rozwiązaniem zdeterminowały sposób doboru obrazów badawczych w ramach eksperymentu (2) zawężając ich rodzaj do obrazów rzeczywistych, których charakterystyki statystyczne są podobne do odpowiednich charakterystyk obrazów będących przedmiotem eksperymentu pierwszego. Mimo tego, choć jak zaznaczono przed chwilą doświadczenie (2) poświęcone było niemalże w całości obrazom rzeczywistym, dla uwypuklenia i sugestywnego przedstawienia charakteru najbardziej znaczących cech proponowanej architektury neuronowej do adaptacyjnej kompresji obrazów warto było również posłużyć się przykładami obrazów wygenerowanych sztucznie o charakterystykach probabilistycznych[1] zbliżonych do pozostałych, rozważanych w dalszej części eksperymentu (2) obrazów rzeczywistych.

Przejdźmy najpierw do analizy relacji między charakterystykami jakościowymi dla rozważanych w niniejszej pracy czterech metod kompresji dla wybranych obrazów eksperymentalnych, które widnieją na rysunkach 4.10a, 4.10b, 4.13a, 4.13b, 4.16a, 4.16b oraz 4.19a i 4.19b. Wspomniane przed chwilą wyniki testów jakościowych dla wymienionych wyżej obrazów eksperymentalnych zebrane są w tabelach 4.8b - 4.11b. Najbardziej istotne różnice jakościowe pomiędzy porównywanymi metodami kompresji można zauważyć analizując dane zgromadzone w tabeli 4.8b, raportujące rezultaty badań jakościowych dla wspomnianych wcześniej przykładowych obrazów 4.11a i 4.11b wygenerowanych w sposób sztuczny, przypominających swą teksturą fakturę tkaniny czy drewna. Najwidoczniejsze różnice dotyczą wyników jakościowych symulacji procesu kompresji i dekodowania dla obrazu 4.11a używanego wcześniej w roli źródła danych optymalizacyjnych dla testowanych metod. Na przykład dla poziomu kompresji $R = 75\%$ obraz treningowy 4.9a odtwarzany jest w procesie kodowania i dekompresji przy użyciu zoptymalizowanej wcześniej sieci neuronowej NFJT2D z miarą PSNR o wartości o około 10 dB większej od wartości wspomnianego wskaźnika dla rozważanego obrazu kompresowanego i odtwarzanego

[1] dokładny opis charakterystyk probabilistycznych obrazów 4.11a i 4.11b znajduje się w punkcie 4.3.2 niniejszego rozdziału

za pomocą standardowej techniki FCT2D. Z drugiej strony w przypadku tym jakość przetworzonego przy pomocy autokodera NFJT2D obrazu uczącego nie odbiega znacznie od tej notowanej dla obrazu odtworzonego za pomocą standardowej architektury neuronowej MLP2D, mieszcząc się w granicy miary PSNR na poziomie około 2.5 dB. Podobna sytuacja ma miejsce dla współczynnika kompresji $R = 25\%$ gdzie różnica jakościowa miar PSNR dla metod NFJT2D i FCT2D wynosi ponownie około 10 dB na korzyść pierwszej z wymienionych metod, zaś analogiczna relacja jakościowa dla wspomnianego poziomu kompresji w przypadku sieci neuronowych NFJT2D i jej standardowego odpowiednika MLP2D waha się w granicach tylko około 2 dB na korzyść drugiej z nich. W analizowanym przykładzie wyniki jakościowe kompresji i dekodowania dla obrazu testowego 4.11b są nieco gorsze dla wszystkich rozpatrywanych metod, przy czym nadal sieć NFJT2D odtwarza rozważny obraz ze średnią przewagą dla wszystkich rozważanych poziomów kompresji wynoszącą około 1.0 dB, co jak wspomniano wcześniej stanowi dwukrotność granicy miary PSNR, przy której człowiek jako użytkownik systemu kompresji zauważa pozytywną poprawę jakości przetwarzanego obrazu - nad standardowym przekształceniem kosinusowym FCT2D, notując jednocześnie bardzo korzystne rezultaty porównawcze względem autokodera MLP2D, gdzie, co istotne, w dwóch przypadkach dla współczynników kompresji równych 50% i 25% wynik jakościowy sieci NFJT2D jest o ponad 1.0 db korzystniejszy od tego raportowanego dla standardowej architektury neuronowej MLP2D.

Podobne liczbowe charakterystyki jakościowe, choć już nie w tak skrajnych proporcjach, dotyczą także pozostałych, uwidocznionych dalej, trzech par obrazów treningowych i testowych, przedstawiających obiekty rzeczywiste. Umieszczone niżej dla każdej z par obrazów uczącego i testowego wizualne porównania ich kompresowanych i odtwarzanych fragmentów potwierdzają przewagę proponowanej metody kompresji NFJT2D nad jej standardowym odpowiednikiem FCT2D, szczególnie w przypadku wysokich współczynników kompresji i jednocześnie pozwalają stwierdzić, że kompresja adaptacyjna z użyciem sieci NFJT2D nie wiele ustępuje metodzie wykorzystującej standardowy autokoder perceptronowy.

Skoncentrujmy jeszcze uwagę na charakterystykach czasowych porównywanych w niniejszym rozdziale metod kompresji zebranych w tabelach 4.8a – 4.11a, które również zdają się potwierdzać ogólną przewagę proponowanego algorytmu adaptacyjnego kompresji obrazów z użyciem architektury sieciowej NFJT2D nad pozostałymi rozpatrywanymi w niniejszej pracy metodami kompresji w sytuacji zbliżonej do rzeczywistych warunków funkcjonowania procedur obliczeniowych adaptacyjnej kompresji obrazów. Choć proponowana architektura neuronowa NFJT2D z oczywistych względów nie jest w stanie dorównać efektywnością czasową procesu optymalizacyjnego i późniejszego etapu funkcjonalnego szybkiemu, dedykowanemu specjalnie rozważanemu zadaniu kompresji algorytmowi FCT2D, płacąc w ten sposób cenę za swą elastyczność i uniwersalność w reprezentowaniu różnorakich efektywnych przekształceń dyskretnych służących kompresji obrazów, to już porównanie jej charakterystyk czasowych z ich odpowiednikami dla architektury MLP2D nie pozostawia wątpliwości o przewadze względem kryterium efektywnościowego proponowanego podejścia nad wymienioną przed chwilą standardową metodą adaptacyjną.

Odnosząc się do pierwszej z poruszanych wyżej kwestii proces optymalizacyjny metody FCT2D obliczania dyskretnego dwuwymiarowego przekształcenia kosinusowego nie wymaga od operatora systemu kompresji adaptacyjnej obrazów wysiłku kalkulacyjnego z uwagi na ustaloną postać obliczeniową tej metody, niezależną od charakteru kompresowanego i dekodowanego sygnału wejściowego. Fakt ten jest umownie odzwierciedlony we wszystkich podanych niżej tabelach porównań efektywnościowych poprzez umieszczenie w nich wartości zerowej dla opisu charakterystyk czasowych procesów optymalizacyjnych metody FCT2D. W przypadku relacji pomiędzy efektywnościami czasowymi algorytmów FCT2D i NFJT2D dla etapów funkcjonalnych kodowania i odtwarzania skompresowanych wcześniej danych obrazowych we wszystkich rozpatrywanych porównaniach i niezależnie od stopnia kompresji, co wynika bezpośrednio z konstrukcji metod FCT2D i NFJT2D – przewaga czasowa pierwszego z wymienionych algorytmów jest w przybliżeniu dwukrotna, a zatem obydwie wymienione metody, z dokładnością do stałej, wykazują złożoność efektywnościową tego samego rzędu dla rozważanego procesu.

Z drugiej strony, jak wspomniano wcześniej, porównania czasowe metod adaptacyjnych, standardowej MLP2D i szybkiej NFJT2D ujawniają w przypadku procesów treningowych zdecydowaną, około trzykrotną przewagę szybkościową proponowanej architektury neuronowej nad siecią MLP2D przy wysokich współczynnikach kompresji, aż do jej ponad szesnastokrotnego poziomu dla współczynników kompresji o wartościach niższych, co może być zaobserwowane chociażby na przykładzie danych umieszczonych w tabelach 4.8a czy 4.9a. Jeśli zaś idzie o czasy realizacji etapów funkcjonalnych wyznaczania odpowiedzi zoptymalizowanych wcześniej sieci dla obydwu rozważanych architektur neuronowych w celu realizacji procesów kodowania i odtwarzania skompresowanych danych obrazowych to przy najwyższych rozpatrywanych wartościach współczynników kompresji czasy te zrównują się, natomiast dla współczynników o wartościach niższych przewaga autokodera NFJT2D nad jego standardowym odpowiednikiem, siecią MLP2D, rośnie nawet pięciokrotnie, co widać we wszystkich podanych tabelach porównań czasowych.

Tab. 4.12: Charakterystyki jakościowo-wydajnościowe rozpatrywanych metod kompresji obrazów

Metoda	*Charakterystyka jakościowo-wydajnościowa*		
	Koszt optymalizacji	*Koszt funkcjonalny*	*Jakość praktyczna*
KLT2D	*bardzo wysoki*	$\mathcal{O}(N^4)$	*maksymalna*
MLP2D	*wysoki*	$\mathcal{O}(N^4)$	*bardzo dobra*
NFJT2D	*niski*	$\mathcal{O}(N^2 log_2 N^2)$	*dobra*
FCT2D	*brak*	$\mathcal{O}(N^2 log_2 N^2)$	*wystarczająca*

Na podstawie wszystkich przeprowadzonych rozważań można w syntetyczny i zwięzły zarazem sposób scharakteryzować wszystkie cztery porównywane ze sobą w niniejszej, eksperymentalnej, części opracowania metody kompresji adaptacyjnej. Porównanie takie w postaci zbiorczego zestawienia najistotniejszych charakterystyk

wspomnianych metod obliczeniowych prezentuje tabela 4.12. Zgodnie z wprowadzonymi wcześniej oznaczeniami w tabeli tej N jest horyzontalnym i wertykalnym zarazem wymiarem pojedynczego fragmentu obrazu kompresowanego i/lub odtwarzanego za pomocą porównywanych ze sobą metod kompresji. Na mocy przedstawionych wyżej porównań można wysnuć wniosek, że neuronowe realizacje szybkich zunifikowanych algorytmów obliczania przekształceń dyskretnych charakteryzują się lepszą o rząd wielkości złożonością obliczeniową od stosowanych obecnie metod neuronowych adaptacyjnej kompresji obrazów, jednocześnie przewyższając znacznie pod względem charakterystyk jakościowych metody standardowe kompresji obrazów, wykorzystujące szybkie, dedykowane algorytmy wyznaczania dyskretnych przekształceń ortogonalnych, których złożoność obliczeniowa jest równa rzędem złożoności prezentowanych algorytmów.

5. PODSUMOWANIE

W niniejszym opracowaniu przedstawiono szybkie zunifikowane algorytmy wyznaczania przekształceń dyskretnych oraz ich neuronowe realizacje, których efektywność została zweryfikowana eksperymentalnie w przykładowym problemie kompresji obrazów. Wyniki badań potwierdziły efektywność skonstruowanych w ten sposób metod adaptacyjnego doboru dyskretnych przekształceń kodujących/dekodujących dla zadania stratnej kompresji wybranych klas obrazów rzeczywistych i doprowadziły do uzyskania schematów kompresji o niskiej złożoności obliczeniowej i lepszej od stosowanych obecnie metod standardowych charakterystyce jakościowej.

Wynik praktyczny niniejszego opracowania polega na eksperymentalnym potwierdzeniu skuteczności neuronowych realizacji szybkich zunifikowanych algorytmów wyznaczania przekształceń dyskretnych w wybranym zagadnieniu optymalizacyjnym, w tym przypadku w problemie kompresji obrazów, co może prowadzić do wniosku, że metody te mogą z podobną skutecznością zostać zastosowane w innych zadaniach optymalizacyjnych.

Warto na koniec dodać, że zdaniem autora, kierunki dalszych badań nad efektywnymi metodami adaptacji szybkich przekształceń zunifikowanych w wybranych zadaniach optymalizacyjnych z pewnością mogą znaleźć szerokie zastosowania, chociażby w realizacjach splotowych sieci neuronowych, gdzie zarówno efektywność obliczeniowa jak i pojemnościowa odgrywają niebagatelną rolę.

BIBLIOGRAFIA

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, p. 633, Oct. 1948.

[2] J. J. Y. Huang i P. M. Schultheiss, "Block quantization of correlated gaussian random variables," *IEEE Trans. Commun. Syst.*, vol. CS-11, pp. 289–296, Sept. 1963.

[3] J. Max, "Quantizing for minimum distortion," *Trans. IRE Inf. Theory*, vol. IT-6, pp. 7, 12, 1960.

[4] ITU, *Information Technology — Digital Compression and Coding of Continuous-Tone Still Images -– Requirements and Guidelines*. International Telecommunication Union, 1991.

[5] A. Tomczyk, D. Puchała, K. Stokfiszewski, i P. S. Szczepaniak, "System wspomagający diagnostykę obrazową," *Diagnostyka procesów i systemów, Exit, Warsaw*, pp. 133–140, 2007.

[6] P. Lipinski, D. Puchała, A. Wosiak, i L. Byczkowska-Lipińska, "Transformer monitoringsystem taking advantage of hybrid wavelet fourier transform," *ISEF 2007 - XIII International Symposium on Electromagnetic Fields in Mechatronics, Electrical and Electronic Engineering, Prague, Czech Republic*, pp. 126–129, Sept. 2007.

[7] H. L. Blitzer i J. Jacobia, *Forensic Digital Imaging and Photography*. 525 B Street, Suite 1900, San Diego, California, 92101-4495, USA: Academic Press, 2002.

[8] C. Champod, C. J. Lennard, P. Margot, i M. Stoilovic, *Fingerprints and Other Ridge Skin Impressions*. 2000 N.W Corporate Blvd., Boca Raton, Florida 33431, USA: CRC Press LLC, 2004.

[9] NEMA, *Digital Imaging and Communications in Medicine (DICOM), Part II*. 1300 N. 17th Street Rosslyn, Virginia 22209 USA: National Electrical Manufacturers Association, 2007.

[10] A. A. Markov, *Extension of the law of large numbers to dependent variables*. Moscow: Izdatel'stvo Akademii Nauk SSSR, 1951.

[11] K. Karhunen, *Über lineare Methoden in der Wahrscheinlichkeitsrechnung*, vol. A. I. Math.–Phys, pp. 1–79. Ann. Acad. Sci. Fennicae., 1947.

[12] M. Loève, *Probability theory*, vol. 46. Springer-Verlag, 1978.

[13] G. E. P. Box i G. M. Jenkins, *Time series analysis. Forecasting and control*. San Francisco: Holden–Day, 1970.

[14] J. E. Besag, "Spatial interaction and statistical analysis of lattice systems," *Journal of Roylal Statistical Society*, vol. 36, pp. 192–236, 1974.

[15] A. K. Jain i G. R. Cross, "Markov random field texture models," *IEEE Trans. Pat. An. Mach. Intel.*, vol. PAMI-5, pp. 25–39, Jan. 1983.

[16] R. Chellappa, "Two-dimensional discrete gaussian markov random field models for image processing," *in Progress in Pattern Recognition 2. L. N. Kanal and A. Rosenfeld Eds. Elsevier Science Publishers B. V.*, pp. 79–112, 1985.

[17] T. S. Huang, *Digital Picture Processing*, vol. 6. New York–Heidelberg–Berlin: Springer–Verlag, April 1975.

[18] U. N. Ahmed i K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1975.

[19] J. Cooley i J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comput.*, vol. 19, p. 297–301, 1965.

[20] N. Ahmed, T. Natarajan, i K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, pp. 90–93, Jan. 1974.

[21] M. D. Flincker i N. Ahmed, "A derivation for the discrete cosine transform," *Proceedings of the IEEE*, vol. 70, pp. 1132–1134, Sep. 1982.

[22] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Inform. Theory*, vol. 28, pp. 129–137, Mar. 1982.

[23] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[24] M. Akay, "Time frequency and wavelets in biomedical signal processing," *IEEE Press Series in Biomedical Engineering*, p. 737, 1998.

[25] H. S. Malvar, *Signal Processing with Lapped Transforms*. Boston, MA: Artech House, 1992.

[26] ITU-ISO, *Information Technology — JPEG 2000 Image Coding System*. International Telecommunication Union and International Organization for Standardization, Mar. 2000.

[27] M. N. Yatsymirskii i P. S. Szczepaniak, "Neural realization of fast linear filters," *In: Proc of the 4^{th} EURASIP - IEEE Region 8 International Symposium on Video/Image Processing and Multimedia Comm.*, pp. 153–157, 2002.

[28] P. S. Szczepaniak, *Obliczenia inteligentne, szybkie przekształcenia i klasyfikatory*. Warszawa: Akademicka Oficyna Wydawnicza EXIT, 2004.

[29] S. Osowski, *Sieci neuronowe w ujęciu algorytmicznym*. Warszawa: Wydawnictwa Naukowo-Techniczne, 1996.

[30] J. Hertz, A. Krogh, i R. G. Palmer, *Wstęp do obliczeń neuronowych*. Warszawa: Wydawnictwa Naukowo-Techniczne, 1995.

[31] M. Jacymirski, D. Puchała, i P. S. Szczepaniak, "Neural realization of dft and dht based on u-transform structure," *Modelling and Information Technologies, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine*, no. 21, pp. 167–173, 2003.

[32] K. Stokfiszewski i P. S. Szczepaniak, "Image compression with a fast transform based architecture neural networks," *Proc. of the 7^{th} Conference on Computer Methods and Systems*, pp. 100–110, 2009.

[33] D. Puchala i K. Stokfiszewski, "Convolutional neural network for image compression with application to JPEG standard," *Proc. 2021 Data Compression Conference (DCC)*, p. 361, 2021.

[34] D. Puchala, B. Stasiak, K. Stokfiszewski, i M. Yatsymirskyy, "Image filtering with fast parametrized biorthogonal transforms implemented on a new GUI research aid system," *Journal of Applied Computer Science*, vol. 21, no. 2, pp. 97–115, 2013.

[35] D. Puchala i K. Stokfiszewski, "Sparse neural networks with topologies inspired by butterfly structures," in *2021 Signal Processing Symposium (SPSympo)*, pp. 226–231, IEEE, 2021.

[36] D. Puchala i K. Stokfiszewski, "Parametrized orthogonal transforms for data encryption," *Computational Problems of Electrical Engineering Journal*, vol. 3, no. 1, pp. 93–97, 2013.

[37] D. Puchala i K. Stokfiszewski, "Highly effective GPU realization of discrete wavelet transform for big-data problems," in *Computational Science – ICCS 2021*, pp. 213–227, Springer International Publishing, 2021.

[38] K. Stokfiszewski, D. Puchala, i M. Yatsymirskyy, "Effectiveness of gpu realizations of parallel prefix-sums computation algorithms," *IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, pp. 435–438, 2018.

[39] D. Puchala, K. Stokfiszewski, K. Wieloch, i M. Yatsymirskyy, "Comparative study of massively parallel gpu realizations of wavelet transform computation with lattice structure and matrix-based approach," *IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 88–93, 2018.

[40] K. Stokfiszewski, K. Wieloch, i M. Yatsymirskyy, "The fast fourier transform partitioning scheme for gpu's computation effectiveness improvement," *Advances in Intelligent Systems and Computing*, vol. 689, pp. 511–522, 2018.

[41] K. Wieloch, K. Stokfiszewski, i M. Yatsymirskyy, "Effectiveness of partitioning strategies of fast fourier transform in gpu implementations," *Proc. 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, pp. 322–325, 2017.

[42] D. Puchala, K. Stokfiszewski, B. Szczepaniak, i M. Yatsymirskyy, "Effectiveness of fast fourier transform implementations on gpu and cpu," *Przegląd Elektrotechniczny*, vol. 92, no. 7, pp. 69–71, 2016.

[43] K. Stokfiszewski i M. Yatsymirskyy, "Effectiveness of lattice factorization of two-channel orthogonal filter banks," *Signal Processing - Algorithms, Architectures, Arrangements, and Applications Conference Proceedings, SPA*, vol. 2015-April, pp. 275–279, 2015.

[44] K. Stokfiszewski, *Kompresja obrazów za pomocą sieci neuronowych realizujących szybkie przekształcenia dyskretne.* Rozprawa doktorska, Wydział Fizyki Technicznej Informatyki i Matematyki Stosowanej Politechniki Łódzkiej, ul. Wólczańska 215, 90-924 Łódź, Polska, Czerwiec 2010.

[45] R. Chellappa i A. K. Jain, *Markov Random Fields: Theory and Application.* Boston: Academic Press, 1993.

[46] N. Barlam i J. Moura, "Noncasual gauss markov random fields: Parameter structure and estimation," *IEEE Trans. on Information Theory*, vol. 39, pp. 1333–1355, Jul. 1993.

[47] H. Derin i P. A. Kelly, "Discrete-index markov-type random processes," *Proceedings of the IEEE*, vol. 77, pp. 1485–1510, Oct. 1989.

[48] N. Ahuja i Schachter, "Image models," *Computing Surveys*, vol. 13, pp. 373–397, Dec. 1981.

[49] A. K. Jain, "Advances in mathematical models for image processing," *Proceedings of the IEEE*, vol. 69, pp. 502–534, May 1981.

[50] A. Skodras, C. Christopoulos, i T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Processing Mag.*, vol. 18, pp. 36–58, 2001.

[51] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *Signal Processing, IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 46, pp. 1043–1053, Apr. 1998.

[52] I. Daubechies, A. Cohen, i J. Feauveau, "Biorthogonal bases of compactly supported wavelets," *AT&T Bell Labs. Tech. Rep. 20878*, May 1990.

[53] K. R. Rao i J. J. Huang, *Techniques and Standards for Image, Video and Audio Coding.* Englewood Cliffs, NI: Prentice-Hall, 1996.

[54] M. Vetterli i C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. Signal Processing*, vol. 40, pp. 2207–2231, Sept. 1992.

[55] N. S. Jayant i P. Noll, *Digital Coding of Waveforms.* Engelwood Cliffs, NJ: Prentice Hall, 1984.

[56] B. Widrow, I. Kollar, i M. Liu, "Statistical theory of quantization," *IEEE Trans. on Instrumentation and Measurement*, vol. 45, no. 6, pp. 353–361, 1995.

[57] K. I. Diamantaras i M. G. Strintzis, "Noisy pca theory and application in filter bank codec design," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3857–3860, Apr. 1997.

[58] K. I. Diamantaras i M. G. Strintzis, "Optimal transform coding in the presence

of quantization noise," *IEEE Trans. on Image Processing*, vol. 8, pp. 1508–1515, Nov. 1999.

[59] D. Puchala, K. Stokfiszewski, i M. Yatsymirskyy, "Image statistics preserving encrypt-then-compress scheme dedicated for jpeg compression standard," *Entropy*, vol. 23, no. 4, pp. 21–30, 2021.

[60] D. Puchala, K. Stokfiszewski, i M. Yatsymirskyy, "Encryption before compression coding scheme for jpeg image compression standard," *Proc. 2020 Data Compression Conference (DCC)*, pp. 21–30, 2020.

[61] D. Puchała i M. Yatsymirskyy, "Porównanie technik stratnej kompresji obrazów realizowanej przy użyciu dyskretnych przekształceń ortogonalnych," *Inżynieria Biomedyczna, Acta Bio-Optica et Informatica Medica*, vol. 12, no. 3, pp. 174–178, 2006.

[62] W. Skarbek, *Algorytmy i standardy kompresji*. Warszawa: Akademicka Oficyna Wydawnicza PLJ, 1998.

[63] R. Tadeusiewicz, *Sieci neuronowe*. Warszawa: Akademicka Oficyna Wydawnicza RM, 1993.

[64] F. L. Luo i R. Unbehauen, *Applied Neural Networks for Signal Processing*. Cambridge: Cambridge University Press, 1998.

[65] M. Jacymirski, K. Stokfiszewski, i P. S. Szczepaniak, "Image compression using fast transforms realized through neural network learning," *Modelling of Computer Science Technologies (in Ukrainian), Ukrainian National Academy of Sciences*, vol. 23, pp. 95–99, Apr. 2003.

[66] B. Stasiak i M. N. Yatsymirskii, "Fast orthogonal neural networks," *Rutkowski et al. (Eds.): ICAISC 2006, LNAI 4029.*, pp. 142–149, 2006.

[67] K. Stokfiszewski i P. S. Szczepaniak, "An adaptive fast transform based image compression," *Proc. of the 9^{th} Conference on Artifficial Intelligence and Soft Computing*, pp. 874–885, Jun. 2008.

[68] M. M. Jacymirski i D. Puchała, "Fast time-decimated algorithms of one-dimensional discrete cosine and sine transforms of type ii and type iv," *Modelling and Information Technologies, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine*, no. 27, pp. 137–148, 2004.

[69] M. M. Jacymirski i D. Puchała, "Fast time-decimated algorithms of discrete two-dimensional cosine and sine transforms of type ii and type iv," *Modelling and Information Technologies, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine*, no. 30, pp. 138–149, 2005.

[70] M. M. Jacymirski i D. Puchała, "Szybkie adaptacyjne algorytmy jednowymiarowych przekształceń kosinusowych drugiego oraz czwartego rodzaju," *Polish and International PD Forum on Conference on Computer Science, Selected Problems of Computer Science*, pp. 722–730, 2005.

[71] D. Puchała i M. Yatsymirskyy, "Fast adaptive algorithms of one-dimensional discrete sine transforms of type two and type four," *Proceedings of System Modelling Control Conference*, pp. 250–260, 2005.

[72] D. Puchała i M. Yatsymirskyy, "Fast adaptive procedures for two-dimensional cosine transforms of type two and type four," *Information Technologies and Systems, Ukraine*, vol. 8, no. 1, pp. 29–36, 2005.

[73] D. Puchała i M. Yatsymirskyy, "Fast adaptive algorithm for fourier transform," *Proceedings of International Conference on Signals and Electronic Systems, Łódź*, pp. 183–185, 2006.

[74] D. Puchała i M. M. Yatsymirskyy, "Fast adaptive algorithm for two-dimensional fourier transform," *Przegląd Elektrotechniczny*, no. 10, pp. 43–46, 2007.

[75] P. Lipiński, "Prediction of oil temperature for substation distribution transformers using wavelet neural networks," *Przegląd Elektrotechniczny*, vol. 12, pp. 202–204, 2008.

[76] B. Stasiak i M. Yatsymirskyy, "Recursive learning of fast orthogonal neural networks," *Proc. of the International Conf. on Signals and Electronic Systems*, pp. 653–656, 2006.

[77] B. Stasiak i M. Yatsymirskyy, "Fast orthogonal neural networks for 2d signal processing," *Przegląd Elektrotechniczny*, vol. II, pp. 167–170, 2007.

[78] B. Stasiak i M. Yatsymirskyy, "On feature extraction capabilities of fast orthogonal neural networks," *Lecture Notes in Computer Science, Springer Berlin Heidelberg*, pp. 27–36, 2007.

[79] J. Stolarek i M. Yatsymirskyy, "Fast neural network for synthesis and implementation of orthogonal wavelet transform," *Image Processing & Communications Challenges, EXIT Warsaw*, pp. 87–94, 2009.

[80] P. S. Szczepaniak, *Computational Intelligence and Applications*. Springer-Verlag, Heidelberg, New York, 1999.

[81] P. S. Szczepaniak, "Neural networks and fuzzy logic in medicine - survey of applications," *Techniki Informatyczne w Medycynie - TIM'99 (Information Technology in Medicine)*, pp. W2:15–24, 1999.

[82] P. J. Lisboa, E. Ifeachor, i P. S. Szczepaniak, *Artificial Neural Networks in Biomedicine*. Springer-Verlag, London, 2000.

[83] P. Nowak, P. S. Szczepaniak, J. Filutowicz, i P. C. Ojha, "Modelling of urea concentration in serum after hemodialysis," *Journal of Medical Informatics and Technologies*, vol. 2, pp. SN:10–14, 2001.

[84] R. Zajdel, E. Kącki, P. S. Szczepaniak, i M. Kurzyński, *Kompendium Informatyki Medycznej*. a-medica Press, Bielsko-Biała, 2003.

[85] M. Yatsymirskyy, "The fast ortogonal trigonometric transform algorithms," *Lviv Academic Express*, p. 219, 1997.

[86] M. M. Yatsymirskyy i R. I. Liskevytch, "Lattice structures for fourier, hartley, cosine and sine transformations (in ukrainian)," *Modelling and Information Technologies, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine*, no. 2, pp. 173–181, 1999.

[87] K. Stokfiszewski, "Gauss-markov random field simulation for image compression tests," *Image Processing & Communications Challenges, EXIT Warsaw*, pp. 144–152, 2009.

[88] D. Puchała, *Rozprawa doktorska Szybkie algorytmy adaptacyjne przekształceń trygonometrycznych*. ul. Wólczańska 215, Łódź, Polska: Instytut Informatyki Politechniki Łódzkiej, 2008.

[89] EPT, *The FFT Demystified*. 21 Leaveden Road, Watford Hertfordshire, WD24 5EB United Kingdom: Engineering Productivity Tools Ltd., 1999.

[90] H. J. Nassbaumer, *Fast Fourier Transforma and Convolution Algorithms*. Springer Ser. in Information Sciences, Springer-Verlag Berlin Heidelberg, 1981.

[91] J. W. Brewer, "Kronecker products and matrix calculus in system theory," *IEEE Trans. on Circuits and Systems*, vol. CAS-25, pp. 772–781, Sept. 1978.

[92] R. Bracewell, "The fast hartley transform," *Proc. of the IEEE*, vol. 72, pp. 1010–1018, Aug. 1984.

[93] A. B. Watson i A. Poirson, "Separable two-dimensional discrete Hartley transform," *Journal of the Optical Society of America A*, vol. 3, pp. 2001–2004, Dec. 1986.

[94] K. I. Diamantaras i M. G. Strintzis, "Optimal linear compression under unreliable representation and robust pca neural models," *IEEE Trans. on Neural Nets*, vol. 10, pp. 1186–1195, Sept. 1999.

[95] J. Jiang, "Image compression with neural networks - a survey," *Elsevier/Signal Processing: Image Communication*, vol. 14, pp. 737–760, 1999.

[96] M. Kurzyński, *Rozpoznawanie obiektów, metody statystyczne*. Wybrzeże Wyspiańskiego 27, 50-370, Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, 1997.

[97] W. S. McCulloch i W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[98] B. Widrow i M. E. Hoff, "Adaptive switching circuits," *IRE WESCON Convention Record, New York*, vol. 4, pp. 96–104, 1960.

[99] F. Rosenblatt, *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Washington, USA: Spartan Books, 1962.

[100] W. A. Jabar, H. B. El-Aawar, i H. S. Al-Mawsoof, "Op-amp structures of fixed weight applications of bp neural network," *Proceedings of the IEEE International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*, pp. 70–75, Feb. 2004.

[101] S. P. Eberhardt, "Analog hardware for delta-backpropagation neural networks," *Patent National Aeronautics and Space Administration. Pasadena Office, CA*, Mar. 1992.

[102] Y. Arima, K. Mashiko, K. Okada, T. Yamada, i A. Maeda, "A self-learning neural network chip with 125 neurons and 10 k self-organization synapses," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 607–611, 1991.

[103] O. Barkan, W. R. Smith, i G. Persky, "Design of coupling resistor networks for neural network hardware," *IEEE Transactions on Circuits and Systems*, vol. 37, pp. 756–765, 1990.

[104] K. R. Rao i P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[105] M. Kurzyński, "The optimal feature extraction procedure for statistical pattern recognition," in *Computational Science and Its Applications - ICCSA 2006*, (Berlin, Heidelberg), pp. 1210–1215, Springer-Verlag, 2006.

[106] M. Kurzyński i A. Rewak, "The ga-based bayes-optimal feature extraction procedure applied to the supervised pattern recognition," in *ICAISC '08: Proceedings of the 9th international conference on Artificial Intelligence and Soft Computing*, (Berlin, Heidelberg), pp. 620–631, Springer-Verlag, 2008.

[107] M. Kurzyński i A. Rewak, "The bayes-optimal feature extraction procedure for pattern recognition using genetic algorithm," in *Artificial Neural Networks – ICANN 2006*, (Berlin, Heidelberg), pp. 21–30, Springer, 2006.

[108] P. Abrahamsen, *A Review of Random Fields and Correlation Functions*. Box 14 Blindern, N-0314 Oslo, Norway: Norwegian Computing Center, Apr. 1997.

[109] W. A. Gardner, *Introduction to Random Processes with Applications to Signals and Systems*. USA: McGrow-Hill, Inc., 1989.

[110] M. Iosifescu, *Skończone procesy Markowa i ich zastosowania*. Warszawa: Państwowe Wydawnictwo Naukowe, 1988.

[111] K. R. Rao i P. C. Yip, *The Transform and Data Compression Handbook*. 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431: CRC Press LLC, 2001.

[112] D. Salomon, *A Guide to Data Compression Methods*. New York: Springer-Verlag, 2002.

[113] R. G. Lyons, *Wprowadzenie do cyfrowego przetwarzania sygnałów*. Warszawa: Wydawnictwa Komunikacji i Łączności, 2000.

[114] Intel, *Intel 64 and IA-32 Architectures Optimization Reference Manual*. 2009.

[115] M. L. Schmit, *Procesory Pentium: narzędzia optymalizacji*. ul. Postępu 18, 02-676 Warszawa, Polska: Mikom, 1997.

[116] "The USC-SIPI Image Database." `https://sipi.usc.edu/database/database.php`.

SPIS RYSUNKÓW

wszystkie rysunki zawarte w monografii są opracowaniem własnym poza Rys. 4.20a i Rys. 4.20b (wraz z ich fragmentami), które zostały pozyskane z otwartej bazy obrazów [116]

SPIS TABEL

wszystkie tabele zawarte w monografii stanowią opracowanie własne

WYKAZ ALGORYTMÓW