Sieć neuronowa do zadania konwersji pomiędzy przestrzeniami RGB i HSV

Sprawozdanie z realizacji projektu sieci neuronowej

Michał Nowicki
95883

1 Definicja problemu

Celem projektu było zaprojektowanie, nauczenie i przetestowanie sieci neuronowej realizującej zadanie konwersji z przestrzeni barw RGB do przestrzeni HSV oraz zadanie do niego odwrotne.

Reprezentacja barw RGB jest standardowym sposób reprezentacji kolorów stosowanym w informatyce. Każdy kolor dostępny na ekranie bądź zdjęciu reprezentowany jest jako złożenie 3 kanałów:

- R (red) - kanał czerwony,
- G (green) - kanał zielony,
- B (blue) - kanał niebieski.

Zakres wartości każdego z kanałów jest sprawą umowną. W zadaniach z dziedziny przetwarzania obrazów standardowo zakłada się, że każdy kanał reprezentowany jest przez jedną wartość całkowitoliczbową z zakresu $< 0, 255 >$. Do reprezentacji takiej wartości standardowo stosuje się zmienną 8-bitową bez znaku (uint8). Innym analogicznym sposobem reprezentacji jest znormalizowanie zakresu każdego kanału do wartości zmiennoprzecinkowych z zakresu $< 0, 1 >$.

Przestrzeń barw RGB nie jest jedynym sposobem reprezentacji możliwego koloru. Najlepiej zrozumieć za pomocą sześcianu kolorów. Umieśćmy sześcian w kartezjańskim układzie współrzędnych tak, aby każdy jego bok był równoległy do jednej osi układu. Każdy punkt wewnątrz oraz na bokach sześcianu może przedstawić za pomocą 3 wartości - (x, y, z). Teraz zakładając, że w 3 wierzchołkach sześcianu leżących na osiach poza początkiem układu mamy punkty odpowiadające całkowicie kolorowi czerwonemu, zielonemu i niebieskiemu. Każdy możliwy kolor posiada odpowiadający mu punkt wewnątrz sześcianu. Jednak, każdy punkt w układzie 3-wymiarowym można także przedstawić w inny sposób. Przykładowo punkt może zostać zaprezentowany jako:

- odległość na odcinku pomiędzy początkiem układu, a przeciwległym narożnikiem sześcianu,
- promieniem okręgu prostopadłego do tego odcinka,
- kątem reprezentującym pozycję na obwodzie odcinka.

W ten sposób można przedstawić każdy punkt sześcianu za pomocą jednoznacznej reprezentacji tych 3 wartości. Właśnie taki sposób można rozumieć przestrzeń HSV, gdzie odległość, promień i kąt odpowiadają odpowiednio wartością hue(H), saturation(S), value(V). Dodatkowo, jeśli istnieje jednoznaczna reprezentacja punktu w przestrzeni RGB oraz w przestrzeni HSV to istnieje także przekształcenie łączące obie reprezentacje.

1.1 Transformacja reprezentacji w przestrzeni RGB do przestrzeni HSV

Zakładając, że wartości kanałów RGB są znormalizowane w celu znalezienie przekształcenia do przestrzeni HSV, można zapisać pomocnicze wartości:

$$C_{max} = max\{R, G, B\} \tag{1}$$

$$C_{min} = min\{R, G, B\} \tag{2}$$

$$\Delta = C_{max} - C_{min} \tag{3}$$

Ostatecznie, wartość koloru w przestrzeni HSV można zapisać jako:

$$H = \begin{cases} 60 \times (\frac{G-B}{\Delta})\ mod\ 6 & \text{, gdy } C_{max} = R \\ 60 \times (\frac{B-R}{\Delta}) + 2 & \text{, gdy } C_{max} = G \\ 60 \times (\frac{R-G}{\Delta}) + 4 & \text{, gdy } C_{max} = B \end{cases} \tag{4}$$

$$S = \begin{cases} 0 & \text{, gdy } \Delta = 0 \\ \frac{\Delta}{C_{max}} & \text{, gdy } \Delta \neq 0 \end{cases} \tag{5}$$

$$V = C_{max} \tag{6}$$

1.2 Transformacja reprezentacji w przestrzeni HSV do przestrzeni RGB

Analogicznie do poprzedniej transformacji można zdefiniować pomocnicze zmienne:

$$C = V \times S \tag{7}$$

$$X = C \times (1 - |\frac{H}{60^\circ}\ mod\ 2 - 1|) \tag{8}$$

$$m = V - C \tag{9}$$

Korzystając ze zmiennych pomocniczych możemy zapisać:

$$(R,G,B) = \begin{cases} (C+m, X+m, m) & \text{, gdy } 0^\circ \leq H < 60^\circ \\ (X+m, C+m, m) & \text{, gdy } 60^\circ \leq H < 120^\circ \\ (m, C+m, X+m) & \text{, gdy } 120^\circ \leq H < 180^\circ \\ (m, X+m, C+m) & \text{, gdy } 180^\circ \leq H < 240^\circ \\ (X+m, m, C+m) & \text{, gdy } 240^\circ \leq H < 300^\circ \\ (C+m, m, X+m) & \text{, gdy } 300^\circ \leq H < 360^\circ \end{cases} \tag{10}$$

2 Sposób rozwiązania problemu

Do rozwiązania problemu wykorzystałem dwie jednokierunkowe sieci feed-forward. Zestawy testowe zostały wygenerowane korzystając z funkcji losującej zaimplementowanej w pakiecie MATLAB oraz wbudowanych funkcji konwersji: $rgb2hsv$ oraz $hsv2rgb$. Ostatecznie stworzyłem zestaw uczący 1000 próbek, z których:

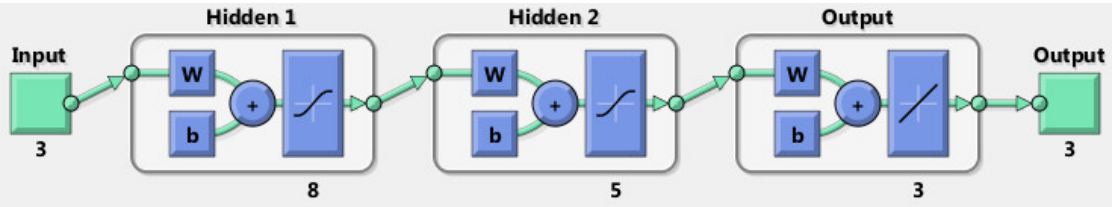
- 70% zostało wykorzystane do nauki sieci,
- 15% zostało wykorzystane do walidacji nauki,
- 15% zostało wykorzystane do przetestowania nauczonej sieci.

2.1 Ogólna struktura sieci realizującej zadania RGB2HSV oraz HSV2RGB

Sieć wybrana do realizacji zadania to sieć jednokierunkowa, wielowarstwowa, gdzie wyjście każdego neuronu warstwy poprzedniej połączone jest z wejściem każdego neuronu warstwy następnej za pomocą wagi. Każda taka sieć składa się z warstw: wejściowej oraz wyjściowej. Dodatkowo każda z sieci można zawierać warstwy ukryte, których obecność umożliwia zwiększenie możliwości sieci neuronowej.

W ramach ćwiczenia do nauki podanej sieci wykorzystałem algorytmu Levenberga-Marquardta, który charakteryzuje się dużymi wymaganiami pamięciowymi, jednak daje bardzo dobre rezultaty.

Ostatecznie zaproponowana sieć jest postaci:



Rysunek 1: Wybrana struktura sieci z dwoma warstwami ukrytymi zawierającymi odpowiednio 8 oraz 5 neuronów

3 Implementacja rozwiązania

Na całość projektu składają się pliki:

- runproject.m - skrypt uruchamiający projekt,
- siecRGB2HSV.m - skrypt inicjalizujący i uczący sieć realizującą zadanie RGB2HSV,
- siecHSV2RGB.m - skrypt inicjalizujący i uczący sieć realizującą zadanie HSV2RGB,
- showRGB2HSV.m - skrypt pokazujący działanie sieci RGB2HSV na wybranym obrazie,
- showHSV2RGb.m - skrypt pokazujący działanie sieci HSV2RGB na wybranym obrazie.

Do stworzenia szkicu pliku sieci neuronowej skorzystałem z menu dostępnego z toolboxu Neural Network. Do realizacji zadania wykorzystano sieć *fitnet* będącą modyfikacją sieci feed-forward przystosowaną do nauki funkcji wejścia-wyjścia zapisanej w danych uczących.

3.1 Dobór struktury sieci

W celu dobrania odpowiedniej struktury sieci zbadano działania sieci dla 5 konfiguracji warstw ukrytych:

- 1 warstwa ukryta z 5 neuronami
- 1 warstwa ukryta z 10 neuronami
- 2 warstwy ukryte z 5 oraz 8 neuronami w odpowiednich warstwach
- 2 warstwy ukryte z 10 oraz 10 neuronami w odpowiednich warstwach
- 3 warstwy ukryte z 5, 8 oraz 5 neuronami w odpowiednich warstwach

Warstwy ukryte sieci	[5]	[10]	[8 5]	[10 10]	[5 8 5]
performance	1.24e-02	2.9e-03	3.7576e-04	5.2558e-04	6.5018e-04
train performance	1.17e-02	3.0e-03	2.5081e-04	3.3942e-04	5.4442e-04
val performance	1.25e-02	3.2e-03	4.3110e-04	9.4445e-04	7.3727e-04
test performance	1.53e-02	2.3e-03	9.0347e-04	9.7547e-04	1.1e-03

Tablica 1: Wyniki uzyskane dla sieci RGB2HSV w zależności od struktury warstwy ukrytej

Warstwy ukryte sieci	[5]	[10]	[8 5]	[10 10]	[5 8 5]
performance	5.5e-03	5.8031e-04	7.2879e-05	2.1859e-05	8.7622e-05
train performance	5.5e-03	5.8791e-04	7.1420e-05	2.0900e-05	8.5433e-05
val performance	5.2e-03	5.3555e-04	7.9391e-05	2.1030e-05	9.4975e-05
test performance	5.7e-03	5.8960e-04	7.3180e-05	2.7163e-05	9.0485e-05

Tablica 2: Wyniki uzyskane dla sieci HSV2RGB w zależności od struktury warstwy ukrytej

Analizując wyniki porównania zebrane w tabelach [1, 2] można zaobserwować, że dla 1 warstwy ukrytej sieci neuronowe RGB2HSV oraz HSV2RGB stosunkowo słabo odtwarzają badane transformacje. Prawdopodobną przyczyną jest fakt, że transformacje realizowana według wzorów [4, 5, 10] są mocno nieliniowe. Dodanie dodatkowej warstwy ukrytej zwiększa możliwość sieci i powoduje poprawę realizacji działania sieci.

Analogiczną sytuację można zaobserwować dla sieci HSV2RGB, gdzie transformacja 10 jest jedynie przedziałami ciągła. Również tutaj dodanie dodatkowej warstwy ukrytej poprawia realizację funkcji i umożliwia skuteczne działanie.

Dla obu sieci można także zaobserwować wolniejsze i słabsze działanie dla sieci posiadającej 3 warstwy ukrytej. Jest to spowodowane rozbudowaną strukturą sieci, której nauka jest już zdecydowanie trudniejsza. Dodatkowo tak rozbudowana sieć potrzebuje większej liczby danych wejściowych (lub iteracji) w celu osiągnięcia wyniku porównywalnego do sieci z 2 warstwami ukrytymi.

3.2 Dobór parametrów nauki

W celu optymalizacji procesu nauki postanowiono zbadać zmianę parametrów nauki na wynik nauki sieci neuronowych. Badano wpływ parametrów:

- epochs - liczba cykli nauki - wartości 100, 1000 oraz 10000.
- goal - cel nauki - wartości 0, 10e-2, 10e-6.

Eksperymenty przeprowadzono dla obu sieci badając wpływ zmiany każdego z parametrów. Wyniki zebrano w tabelach.

Sieć RGB2HSV, goal = 0	epochs = 100	epochs = 1000	epochs = 10000
performance	0.0015	7.5793e-04	6.0089e-04
train performance	9.8379e-04	7.2850e-04	5.5945e-04
val performance	0.0018	8.9602e-04	6.6212e-04
test performance	0.0035	7.5717e-04	7.3306e-04

Tablica 3: Wyniki nauki sieci RGB2HSV dla goal = 0 i różnych wartości epochs

Sieć HSV2RGB, goal = 0	epochs = 100	epochs = 1000	epochs = 10000
performance	4.8661e-04	1.3909e-04	1.5606e-04
train performance	4.2981e-04	1.2711e-04	1.4405e-04
val performance	6.4971e-04	1.6145e-04	1.8000e-04
test performance	5.8857e-04	1.7261e-04	1.8817e-04

Tablica 4: Wyniki nauki sieci HSV2RGB dla goal = 0 i różnych wartości epochs

Analizując wyniki uzyskane w tabelach można zaobserwować, że dla sieci RGB2HSV optymalną wartością jest epochs = 1000, ponieważ dla wartości 100 następuje niedouczenie sieci, natomiast do wartości większej nie następuje poprawa nauczonej wartości. Identyczna sytuacja ma miejsca dla sieci HSV2RGB.

Sieć RGB2HSV, epochs = 1000	goal = 0	goal = 0.000001	epochs = 0.01
performance	7.5793e-04	8.4823e-04	0.0092
train performance	7.2850e-04	7.3610e-04	0.0087
val performance	8.9602e-04	7.8010e-04	0.0090
test performance	7.5717e-04	0.0014	0.0113

Tablica 5: Wyniki nauki sieci RGB2HSV dla epochs = 1000 i różnych wartości goal

Sieć HSV2RGB, epochs = 1000	goal = 0	goal = 0.000001	epochs = 0.01
performance	1.3909e-04	1.2416e-04	0.0099
train performance	1.2711e-04	1.1849e-04	0.0095
val performance	1.6145e-04	1.4275e-04	0.0121
test performance	1.7261e-04	1.3202e-04	0.0095

Tablica 6: Wyniki nauki sieci HSV2RGB dla epochs = 1000 i różnych wartości goal

Analizując wyniki wpływu celu nauki można zauważyć, że dla obu sieci największa badana wartość spowodowała słabe wyniki badanych sieci. Pomiędzy wartością 0, a 1.0e-6 nie zaobserwowano większych różnic w naukach sieci.

Ostatecznie zdecydowano się zostawić parametry nauki:

- epochs = 1000,
- goal = 0.

3.3 Porównanie czasu działania z wbudowaną funkcją MATLABa

Analizując wzory użycie do konwertowania pomiędzy przestrzeniami można zauważyć, że są to dość skomplikowane, nieliniowe wzory. Z drugiej strony sieć neuronowa składa

się jedynie z operacji ważonej sumy wejść oraz wyliczania funkcji aktywacji. Dlatego postanowiono zbadać czasy działania obu rozwiązań na zestawie danych zawierającym $512 \times 512 = 262144$ pixeli. Uzyskano:

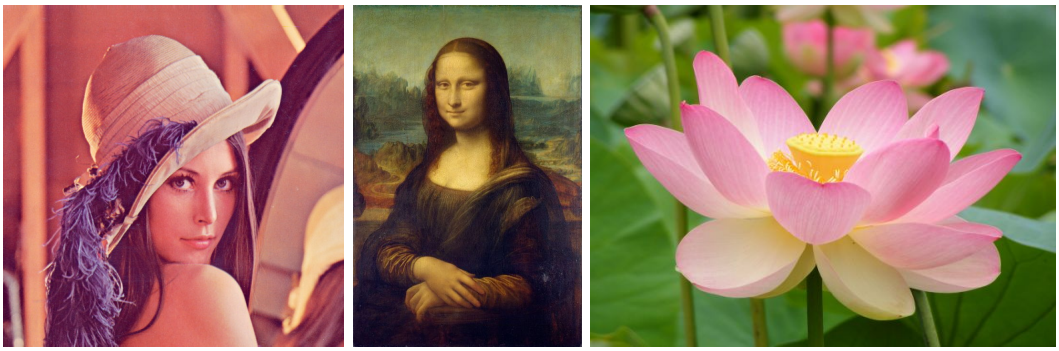
- Sieć neuronowa: 0,476180 [s].
- Wbudowana funkcja MATLABa: 0,076601 [s].

Okazało się, że realizacja obliczeń z siecią neuronową jest około 6-krotnie wolniejsza od realizacji obliczeń za pomocą wbudowanej funkcji MATLABa. Jednak w przypadku nieznania analitycznej postaci odtwarzanej funkcji może to być jedna z lepszych metod szybkiego i skutecznego odtworzenia działania funkcji.

3.4 Prezentacja działania na przykładowych obrazach

W celu prezentacji działania obu nauczonych sieci wykorzystano 3 obrazy:

- obraz *lena.jpg* powszechnie stosowany z przetwarzaniu obrazów do różnych testów,
- obraz *monalisa.jpg* przedstawiający słynny obraz Leonardo da Vinci,
- obraz *flower.jpg* przedstawiający kwiat.

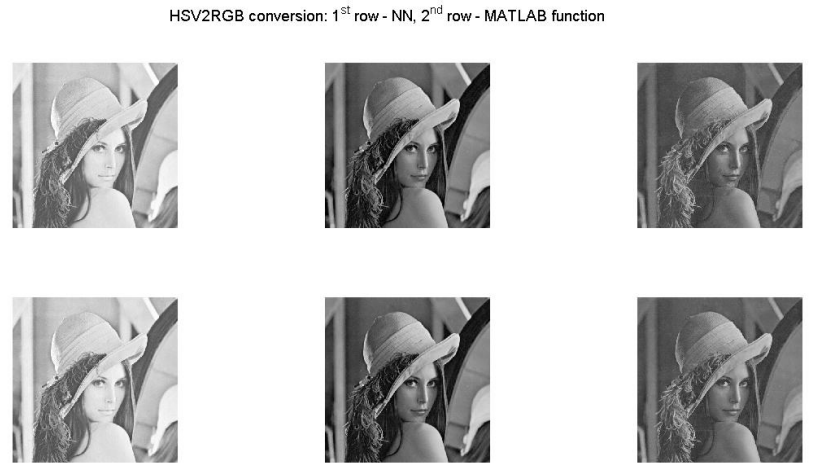


Rysunek 2: Rysunki testowe: lena.jpg, monalisa.jpg oraz flower.jpg

W następnych podsekcjach przedstawiono wyniki konwersji uzyskane poprzez zastosowanie sieci neuronowej. Dla każdej konwersji, w pierwszym wierszu przedstawiono wyniki uzyskane z zastosowaniem sieci neuronowej. W drugim wierszu przedstawiono wynik oczekiwany powstały poprzez zastosowanie wbudowanej funkcji w pakiecie MATLAB.

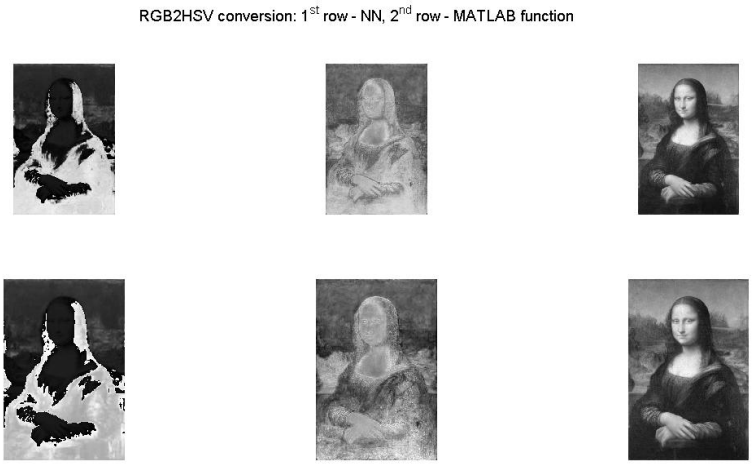
3.4.1 Wyniki uzyskane dla obrazu *lena.jpg*

Rysunek 3: Wyniki konwersji z przestrzeni RGB do przestrzeni HSV. W kolumnach przedstawiono odpowiednio wartości kanałów: H, S oraz V.

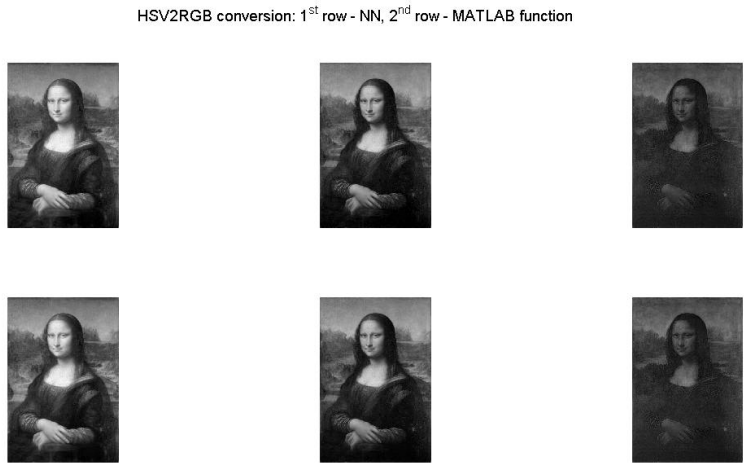


Rysunek 4: Wyniki konwersji z przestrzeni HSV do przestrzeni RGB. W kolumnach przedstawiono odpowiednio wartości kanałów: R, G oraz B.

3.4.2 Wyniki uzyskane dla obrazu *monalisa.jpg*

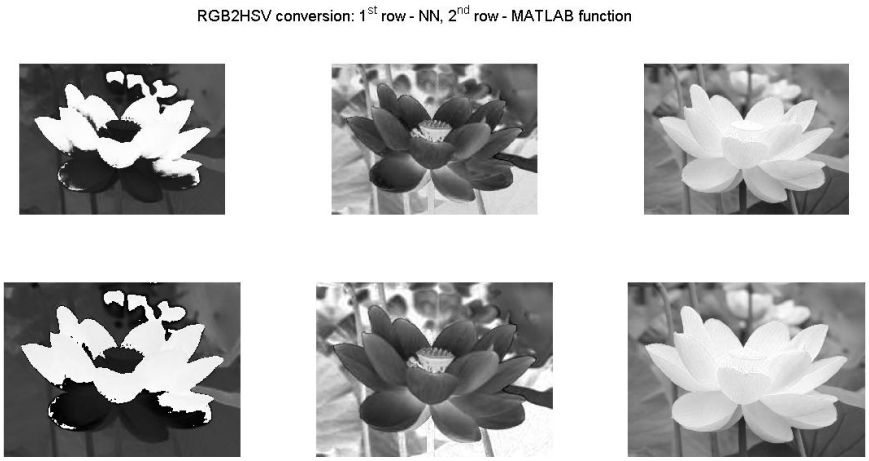


Rysunek 5: Wyniki konwersji z przestrzeni RGB do przestrzeni HSV. W kolumnach przedstawiono odpowiednio wartości kanałów: H, S oraz V.



Rysunek 6: Wyniki konwersji z przestrzeni HSV do przestrzeni RGB. W kolumnach przedstawiono odpowiednio wartości kanałów: R, G oraz B.

3.4.3 Wyniki uzyskane dla obrazu *flower.jpg*



Rysunek 7: Wyniki konwersji z przestrzeni RGB do przestrzeni HSV. W kolumnach przedstawiono odpowiednio wartości kanałów: H, S oraz V.

Rysunek 8: Wyniki konwersji z przestrzeni HSV do przestrzeni RGB. W kolumnach przedstawiono odpowiednio wartości kanałów: R, G oraz B.

4 Wnioski

Sieć fitnet w realizacji zadania transformacji pomiędzy przestrzeniami barw spisuje się bardzo dobrze, choć wymaga ona trochę wysiłku w celu odpowiedniego dobrania struktury oraz parametrów nauki. Dodatkowym wynikiem realizacji projektu jest fakt, że do jakościowo zadowalającej konwersji pomiędzy przestrzeniami RGB i HSV konieczne jest zastosowanie sieci neuronowej z dwoma warstwami ukrytymi. W czasie realizacji tego projektu okazało się, że sieć fitnet o takiej strukturze może skutecznie odtwarzać działania nawet mocno nieliniowych funkcji.

5 Uwagi końcowe

Najwięcej problemów podczas realizacji projektu sprawiło dynamiczne typowanie zmiennych w MATLABie połączony z faktem błędnej informacji w helpie do funkcji rgb2hsv. W informacji do funkcji rgb2hsv można znaleźć informację, że oczekiwane wejście ma zawierać wartości z przedziału $< 0, 1 >$ na każdym kanale. Dopiero analiza kodu funkcji MATLABa pokazała, że konwersja realizowana jest wewnątrz funkcji i zależy od typu zmiennej. Rozwiązaniem okazało się silne typowanie zmiennych (wymuszenie odpowiednich typów) i ostrożne konwertowanie pomiędzy znormalizowaną reprezentacją kanału kolorów, a reprezentacją w postaci 8-bitowej zmiennej całkowitej uint8.