Kraków, 30.05.2012

**Sztuczne Sieci Neuronowe**

**- Kompresja obrazów**

**Joanna Misztal**

1. Zastosowana metoda – sieć Kohonena stworzona za pomocą biblioteki som w języku R.
2. Kompresowane zdjęcia miały rozmiar ponad 20 MB, były zapisane w formacie .TIFF i miały wymiary 4320x2667.
3. Spostrzeżenie: po odczytaniu i ponownym zapisaniu obrazu przez R jego rozmiar zmniejszał się do około 1,5 MB.
4. Wyniki dla przykładowych zdjęć przy wykorzystaniu różnych parametrów kompresji:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Zdjęcie | Rozmiar[KB]  (rozmiar po zapisaniu przez R) | Po kompresji i zapisaniu[KB] | Parametry SOM | Jakość | czas | Wielkość kwadratu wykorzystanego do kompresji |
| bc1.tiff | 28 789(1533) | 963 | 10x10,  5 iteracji | D:\UJ\sieci neuronowe\kompresja\afterCompression1,4x4.jpg | Ok. 8min | 4x4 |
| bc1.tiff | 28 789(1533) | 329 | 10x10,  5 iteracji | D:\UJ\sieci neuronowe\kompresja\afterCompression1,8x8.jpg | Ok. 2min | 8x8 |
| bc1.tiff | 28 789(1533) | 348 | 8x8,  5 iteracji |  | Ok. 1 min | 8x8 |
| bc1.tiff | 28 789(1533) | 346 | 8x8, 10 iteracji |  | Ok. 2 min | 8x8 |
| bc1.tiff | 28 789(1533) | 365 | 12x12, 5 iteracji | D:\UJ\sieci neuronowe\kompresja\bc1afterCompression.jpg | Ok. 3 min | 8x8 |
| bc2.tiff | 22523(1620) | 312 | 12x12, 5 iteracji | D:\UJ\sieci neuronowe\kompresja\afterCompression.jpg | Ok. 3 min | 8x8 |
| bc2.tiff | 22523(1620) | 309 | 10x10, 5 iteracji | D:\UJ\sieci neuronowe\kompresja\afterCompression2,8x8.jpg | Ok. 2 min | 8x8 |
| bc2.tiff | 22523(1620) | 904 | 15x15, 5 iteracji | D:\UJ\sieci neuronowe\kompresja\afterCompression.jpg | Ok. 10 min | 4x4 |

1. **Wnioski:**

W zależności od parametrów sieci SOM oraz wielkości kwadratów, które są przetwarzane przez sieć, jakość i stopień kompresji zmieniają się. Jakość przy podziale na kwadraty o wymiarze 4x4 piksele jest nieco lepsza niż 8x8, jednak znacznie wzrasta czas kompresji i wielkość pliku po zapisaniu. W tabeli podane zostały wielkości plików po odczycie i zapisie ich przez R(w nawiasie) i w ten sam sposób po zapisie po kompresji. Dla podziału 8x8 stopień kompresji to około 20%, natomiast dla 4x4 to już 60% (stosunek obrazu po i przed kompresją dla obrazów zapisywanych przez R).

Na jakość kompresji nie ma dużego wpływu maksymalna liczba iteracji dla sieci. Jest za to pewna różnica w zależności od wymiarów sieci – dla większych wymiarów wyniki są lepsze. Zdarza się jednak (jak w tabeli przykład 5), że wektory zostaną błędnie dopasowane i następuje zmiana kolorów na obrazku.

Średni błąd dla klas liczony jako suma błędów pomnożona przez promień sąsiedztwa dla znormalizowanych danych dla podziału na kwadraty 4x4 i SOM o wymiarach 15x15 wynosi 1.14, natomiast przy podziale 8x8 i SOM 10x10 jest to 2.55.

Teoretycznie dane, które wystarczyłyby do odtworzenia obrazu to wektor liczb całkowitym o długości równej liczbie kwadratów, na które został podzielony obraz oraz macierz wag sieci składająca się z tylu wektorów, ile jest klastrów w sieci, każdy wektor to tyle liczb typu double ile jest pikseli w kwadracie (tu 16 lub 64). Właściwie można by przesłać tylko te wektory, do których zostały dopasowane jakieś wektory z obrazu.

Zatem dla podziału na kwadraty 8x8 przy sieci o wymiarach 10x10 dla zdjęcia 4320x2667 wystarczyłoby przesłać 179 820 liczb całkowitych oraz co najwyżej 6400 liczb typu double.

1. Kod programu:

*bc1<-readTiff("bc1.tiff")*

*bc1<-imgRGB2Grey(bc1)*

*nrOfClasses = 100*

*xSize = dim(bc1)[1]*

*ySize = dim(bc1)[2]*

*d = 8*

*xDim = floor(xSize/d)*

*yDim = floor(ySize/d)*

*xyDim = xDim\*yDim*

*tmp<-matrix(0,xDim,yDim)*

*for (i in 1:yDim)*

*for(j in 1:xDim)*

*tmp[j,i] = i+yDim\*(j-1)*

*mx<-matrix(bc1,xSize,ySize)*

*mx<-mx/256*

*splitMatrix<-array(0, dim=c(d,d,xyDim))*

*n=0*

*for(i in 1:xDim)*

*for(j in 1:yDim)*

*{*

*n=n+1*

*splitMatrix[,,n]<-mx[(d\*(i-1)+1):(d\*(i-1)+d),(d\*(j-1)+1):(d\*(j-1)+d)]*

*}*

*splitVectors<-matrix(,64,xyDim)*

*for(i in 1:xyDim)*

*splitVectors[,i] <- c(splitMatrix[,,i])*

*som<-som(t(splitVectors),xdim=floor(nrOfClasses^(1/2)),ydim=floor(nrOfClasses^(1/2)),topol="rect",neigh="gaussian",rlen=5)*

*compressedClasses<-matrix(,xyDim,1)*

*for(i in 1:xyDim)*

*compressedClasses[i]<-som$visual$x[i]+(som$visual$y[i]\*nrOfClasses^(1/2))+1*

*compressedVectors<-matrix(,xyDim,64)*

*for (i in 1:xyDim){*

*compressedVectors[i,]<-som$code[compressedClasses[i],]*

*}*

*compressedMatrix<-array(0, dim=c(d,d,xyDim))*

*for (i in 1:xyDim)*

*compressedMatrix[,,i] = matrix(compressedVectors[i,],ncol=d,nrow=d)*

*cMx<-matrix(0,xDim\*d,yDim\*d)*

*for (i in 1:yDim)*

*for(j in 1:xDim)*

*cMx[(d\*(j-1)+1):(d\*(j-1)+d),(d\*(i-1)+1):(d\*(i-1)+d)]<-compressedMatrix[,,tmp[j,i]]*

*cMx<-cMx\*256*

*img<-imagedata(cMx)*