Jakub Janecki, gr 2

Dokumentacja projektu – Podstawy Sztucznej Inteligencji

**1.Opis problemu:**

Zagadnieniem które wybrałem jako swój projekt końcowy jest przerysowywanie danych obrazów, a następnie obliczanie podobieństwa między plikiem wyuczonym, a innym przyłożonym na wejście sieci. Projekt oparłem na sieciach WTM, względnie na WTA w celach testowych. Docelowo moim założeniem była prosta daktyloskopia, porównanie odcisków palców na podstawie podobieństwa ich skanów. Przy implementacji projektu nie korzystałem z żadnych frameworków.

**2. Opisy podproblemów wskazując wykorzystane metody:**

1. Klasteryzacja obszaru na mniejsze fragmenty matrycy pikseli

- w celu zmniejszenia złożoności problemu przeprowadzam podział obrazu na odpowiednio mniejsze klastry, przykładowo obraz 400x400 na 16 klastrów, każdy po 100x100 pikseli. Obiekt klastra zawiera współrzędne punktu 0 tego fragmentu w układzie globalnym, do tego tablice koloru każdego piksela

1. Uczenie sieci neuronowej

Uczenie przeprowadzam metodą WTM, polega ona na wyborze neuronu o najbardziej zbliżonym zestawie wag do wektora wejściowego. Zwycięzki neuron, oraz odpowiednia liczba jego sąsiadów, uczona jest danego zestawu wejściowego, czyli w praktyce po n iteracjach upodabnia swoje wagi do danych wejściowych. W moim przypadku, danymi wejściowymi był dany klaster, którego następnie jeden z neuronów uczył się odwzorować. Wynikiem uczenia była mapa, obrazująca który neuron odwzorowywuje który klaster. W przypadku testów z WTA, uczenia sąsiadów nie było, co nieznacznie spowalnia proces. Rozważałem modyfikacje z wykluczaniem ponownej nauki neuronu, w celu jak najlepszego odwzorowania każdego klastra. Metoda ta zwróciła wręcz identyczną skuteczność porównania (100%) przy każdym uruchomieniu programu, lecz nie wykorzystuje ona pełnego potencjału sieci neuronowych, tylko ich umiejętność do obliczania podobieństwa i adaptacji. Metoda z ponownym uczeniem neuronu, wykazuje jak dotąd dużo mniejszą skuteczność.

1. Wizualizacja wyników

Wizualizacja wyników przedstawia aktualny stan wag każdego neuronu, są one mapowane na odpowiednie współrzędne, i docelowo mają stanowić kolor danego piksela

1. Porównanie obrazów

Do porównania obrazów wykorzystałem ceche sieci neuonowych WTA, czyli obliczanie podobieństwa przyłożonych danych do wyuczonego wzorca. Wyczam sieć odcisku palca danej osoby, a następnie przykładam na wejście skany innych jej odcisków, oraz różnych innych osób

1. Normalizacja

Wartości kolorów znormalizowane do przedziału -1 do 1

1. Liczenie wartości sieci

Na podstawie mapy klastrów na odwzorowywujące je neurony, na wejście sieci, przykładam każdy klaster w odpowiednie miejsce. Odpowiednie neurony są już wyuczone jak dany fragment obrazu powinien wyglądać.

1. Zatrzymanie uczenia

Uczenie prowadzone jest do momentu, w którym różnica podobieństw wag do wzorca jest mniejsza od zadanego progu błędu. W tym momencie na sieć przekazywane jest kolejny fragment obrazu

*Wykres różnicy podobieństw wag do wektora wejścia, względem iteracji*

**3.Dane wejściowe:**

1. Przykładowe dane przedstawiające kolory pikseli matrycy:

-5855578

-6250336

-10592674

-10592674

-10592674

-8224126

-8618884

-7039852

-6645094

-5855578

-1513240

-1513240

-1513240

-1513240

-1513240

-1513240

1. Przykładowe dane po normalizacji:

0 = {Double@1586} "-0.9737901176470588"

1 = {Double@1587} "0.9999998823529412"

2 = {Double@1588} "0.9999998823529412"

3 = {Double@1589} "0.9999998823529412"

4 = {Double@1590} "0.9999998823529412"

5 = {Double@1591} "0.9999998823529412"

6 = {Double@1592} "0.9999998823529412"

7 = {Double@1593} "0.9999998823529412"

8 = {Double@1594} "0.9999998823529412"

9 = {Double@1595} "0.9999998823529412"

10 = {Double@1596} "0.9999998823529412"

11 = {Double@1597} "0.9999998823529412"

12 = {Double@1598} "0.9999998823529412"

13 = {Double@1599} "0.9999998823529412"

1. Obrazy wejściowe:

Obrazy o nazwie „Osoba1\_1”, „Osoba1\_2”, Osoba1\_3” są skanami odcisków palców tej samej osoby. Pozostałe przedstawiają inne odciski.

**4. Konfiguracja:**

1) Sieć:

- ADELINE

- 5 x 1 neuronów

Próg błędu:

- 0.0000001

Stopień kompresji:

- 1.0

Ilość klastrów:

- 2x2

2) Sieć:

- ADELINE

- 150 x 1 neuronów

Próg błędu:

- 0.0000001

Stopień kompresji:

- 1.0

Ilość klastrów:

- 16x16

**5. Wyniki Uczenia, i obliczanie podobieństwa:**

Sieć została wyuczona obrazu „Osoba1\_1”

1)

Klaster 0 - neuron: 2

Klaster 1 - neuron: 3

Klaster 2 - neuron: 1

Klaster 3 - neuron: 3

Ilosc epok: 4

Czas uczenia: (min): 0.019555022883333332

Obraz nr:f1: 34.13530396747855

Obraz nr:f2: 180.7133763694069

Obraz nr:f3: -105.02812440533083

Obraz nr:fS2: 189.7169342086369

Obraz nr:fS3: 29.03463729755029 -> zwycięzca, najmniejsza różnica podobieństwa

2)

Ilosc epok: 256

Czas uczenia: (min): 0.1770297722

Obraz nr:Osoba1\_2: -430.7893113979311

Obraz nr:Osoba2: 403.8083771364165

Obraz nr:Osoba3: 574.550124365708

Obraz nr:Osoba4: 555.5184276379898

Obraz nr:Osoba1\_3: 147. 52075168374733 -> zwycięzca, najmniejsza różnica podobieństwa

6.Wnioski:

Metoda CWTM dla tego zastosowania osiąga wysoką skuteczność pomimo tego, że sprowadza sieć do pracy „na sztywno”, WTM i WTA natomiast, osiągają skuteczność na poziomie 40-50%. Moim wnioskiem z tego projektu jest konkluzja, że proste porównywanie matrycy pikseli na potrzeby daktyloskopii jest zwyczajnie niewystarczające. Można było by wyuczać sieć na kilku obrazach naraz, co mogło by podnieść skuteczność takiego porównania.

Wybrane fragmenty kodu:

Funkcja wybierająca neuron najlepiej odwzorowywujący dany klaster

public static int szukajZwyciezcyWTM(Warstwa aktualna, Klaster klaster){  
 Double odleglosci[]=new Double[aktualna.ilosc\_neuronow];  
 for(int j=0;j<aktualna.ilosc\_neuronow;j++){ //dla kazdego nauronu  
 AbstractNeuron aktualny\_neuron=aktualna.neurony[j];  
 Double suma=0.0;  
 for (int k = 0; k < aktualny\_neuron.wejscia.length; k++) { //dla kazdego wejscia neuronu  
  
 Double w=aktualny\_neuron.wagi[k];  
 Double x=aktualny\_neuron.wejscia[k];  
  
 suma+=Math.*pow*(w-x,2);  
 }  
 Double odleglosc=Math.*sqrt*(suma);  
 odleglosci[j]=odleglosc;  
 }  
 int min=0;  
  
 for(int i=0;i<aktualna.ilosc\_neuronow;i++) {  
 //if(zwyciezcy.get(i)==null) { //cwtm  
 if (odleglosci[i] < odleglosci[min]) {  
 min = i;  
 }  
 //}  
 }  
  
 System.*out*.println(odleglosci[min]);  
  
 if(odleglosci[min]<=0.0000001)  
 {  
 klaster.id\_neuron=min;  
 klaster.odwzorowano=true;  
 System.*out*.println(min);  
 return min;  
 }  
  
 return min;  
}

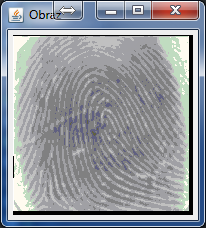
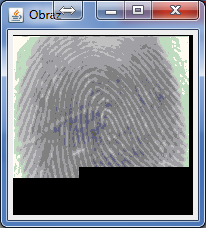
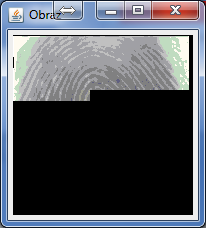
Epoka

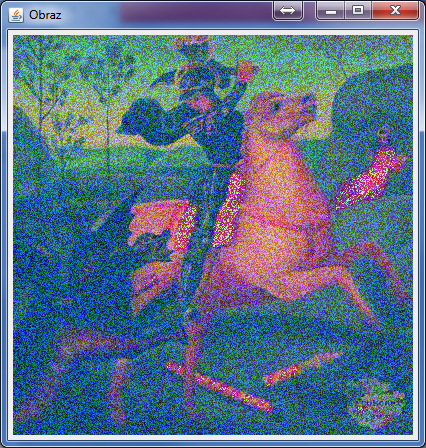
public Pair Epoka(Klaster k, BufferedImage gui\_image, Okno o){  
 int it=input.length;//Dane.inputTest.length  
  
 Pair wynik=null;  
  
  
 do {  
 wynik = Run(k, gui\_image, o);  
 wynik.getL();  
 if (k.odwzorowano) {  
 *zwyciezcy*.put( k.id,(Integer) wynik.getL());  
 System.*out*.println("Klaster " + k.id + " - neuron: " + (Integer) wynik.getL());  
 return wynik;  
 }  
  
 }while(!k.odwzorowano);  
  
 System.*out*.println("Klaster "+k.id+" - neuron: "+(Integer) wynik.getL());  
 *zwyciezcy*.put(k.id,(Integer) wynik.getL());  
 return wynik;  
  
}

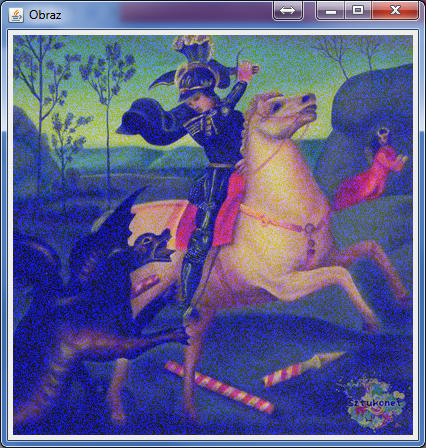
Korekcja wag

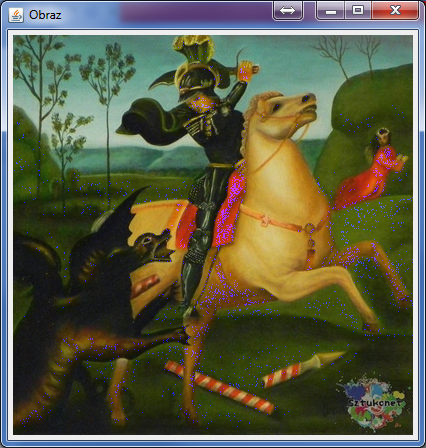
private Double korygujWagi( Warstwa aktualna,int id,Klaster k, BufferedImage gui\_image,Okno o){  
 Double blad=0.0;  
 int[] wsp=getWspolrzedne(id);  
 int wx=wsp[0];  
 int wy=wsp[1];  
 Double korekta\_sum=0.0;  
 for(int j=0;j<aktualna.neurony.length;j++){  
 AbstractNeuron neuron=aktualna.neurony[j];  
 int dx=( j % ((int)Math.*sqrt*(input.length)) ) - wx;  
 int dy=( j / ((int)Math.*sqrt*(input.length)) ) - wy;  
 double factor = Math.*exp*( -(double) ( dx \* dx + dy \* dy ) / ((lambda\*lambda)\*2) );  
  
 for(int i=0;i<neuron.wagi.length;i++){  
 Double e=factor\*(neuron.wejscia[i]-neuron.wagi[i]);  
 blad+=Math.*abs*(e);  
 Double korekta=neuron.learningRate\*e;  
 neuron.wagi[i]+=korekta;  
 korekta\_sum+=korekta;  
 int[] wsp2=this.getWspolrzedne(i);  
 int gx=wsp2[0]+k.x\_start;  
 int gy=wsp2[1]+k.y\_start;  
 double waga=network.warstwy[0].neurony[id].wagi[i];  
 waga=Siec.*denormalizujKolor*(waga);  
 int color=(int)waga;  
 gui\_image.setRGB(gy,gx,color);  
 o.repaint();  
 }  
 }  
 return blad;  
}

Proces odwzorowywania obrazu:









Obraz prawidłowo przerysowany

: