CNN, 4. gyakorlat

2017. tavasz

Mai feladatok

1. ismerkedés a MatCNN szimulátorral

2. egyszerű objektumklasszifikáció megvalósítása

MatCNN

1. szimulátor letöltése:
   a. Windows/Linux alá: http://users.itk.ppke.hu/~horan/CNN/matCNN.zip
   b. OS X alá: http://users.itk.ppke.hu/~horan/CNN/os_x_matcnn.zip
   c. kicsomagolás a Matlab munkakönyvtárába
2. **`gyak4.zip`** letöltése, kicsomagolása, ebből **`temlib_plus.m`** bemásolása a **`MatCNN`** mappa gyökerébe
3. a Matlab fő szalagján, *Home* alatt: *Set Path*
   a. a felugró ablakban *'Add with Subfolders...'*
   b. válasszuk ki a kicsomagolt **`MatCNN`** mappát
   c. *Save* majd *Close*
   d. géptermi gépekhez megjegyzés: nincs írási jogunk bizonyos mappákhoz, ezért a root hitelesítésre a válasz '*No*', aztán mentsük más helyre a **`pathdef.m`** fájlt (pl a **`Documents`** mappádba vagy a **`MATLAB`** mappádba azon belül)
4. template-gyűjtemény (függvénykönyvtár, pl v3.1): http://cnn-technology.itk.ppke.hu/

gyak4_introduction.m

- 12 template-példa mintamegvalósítással, blokk-kommentben
- template-ek használatának bevezető lépései -- ***egyszer, a szkriptünk elején:***
  - **`close all; clear all;`** → minden bezárása, törlése a munkatérből
  - **`cnn_setenv;`** → CNN környezet inicializálása, létrejön az mCNN struktúra részeivel együtt
  - **`mCNN.TemGroup = 'temlib_plus';`** → ha szeretnénk felülírni az alapértelmezett függvénykönyvtárat (template-katalógust); szövegfájl az egyes template-ek **`A`**, **`B`** és **`I`** (elméleti leírásban **`z`** azaz *bias*) mátrixaival/értékeivel, pl:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% EDGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

EDGE_A = [0 0 0;
          0 2 0;
          0 0 0];
EDGE_B = [-0.25 -0.25 -0.25;
          -0.25  2.0  -0.25;
          -0.25 -0.25 -0.25];
EDGE_I = -1.5;
```

gyak4_introduction.m

- egy template használatának lépései:
  - `in_pic = lbmp2cnn('piclib/BABOON.bmp');` → egy kép betöltése egy változóba
  - `mCNN.INPUT1 = 1.* in_pic;` → a szimulátor bemenetének megadása; elemenként 1-gyel szorzás, értelme: új memóriaterület allokálásának kikényszerítése
  - `init_state = lbmp2cnn('piclib/BABOON.bmp');`
  - `mCNN.STATE = 1.* init_state;` → a szimulátor kezdőállapotának beállítása
  - `mCNN.Boundary = 2;` → határfeltétel beállítása: konstans: [-1 … 1], zeroflux: 2, periódikus: 3
  - `mCNN.TimeStep = 0.1;` → elemi integrálási idő
  - `mCNN.IterNum = 20;` → összes iterációk száma
    megjegyzés: habár a szimuláció összidejét a `TimeStep x IterNum` határozza meg, bizonyos template-ek esetén nem közömbös, hogy milyen finomságú elemi lépésekkel szimulálunk
  - `loadtem('GRADT');` → template-értékek betöltése név alapján, az előzőleg megadott függvénykönyvtárból (`mCNN.TemGroup` -ban rögzített fájl)
  - `runtem;` → szimulátor futtatása
  - `figure;`
  - `cnnshow(mCNN.OUTPUT);` → az `mCNN` struktúra `OUTPUT` mezőjéből kérhető le a kimenet, a `cnnshow` dinamikatartomány-helyes megjelenítést csinál ([0 255] vs [-1 1], bw csere)

Objektumklasszifikáció

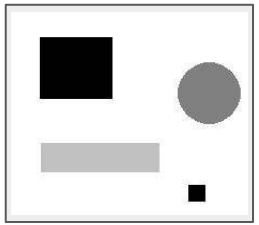
Detektálandó objektumok:

1. kicsi, fekete négyzet (pl. 20x20 vagy annál kisebb)
2. középszürke téglalap (oldalak közt az arány legalább 1.5-szeres)
3. lyukas objektum

Adottak különböző képek a **`gyak4_images`** mappában, egy kép akár több különböző típusú objektumot is tartalmazhat.

A **`LoadGT`** függvény egy oszlopvektorként visszaadja a referenciamegoldást. A használt kódrendszer:

- legkisebb helyiérték -- kicsi, fekete négyzet, legnagyobb helyiérték: lyukas objektum;
- `0` -- nem tartalmaz az adott osztályú objektumból, `1` -- tartalmaz
- pl: `011` → van kis, fekete négyzet és szürke téglalap a képen



Objektumklasszifikáció - segédfüggvények

Egy ilyen tömbprocesszoron mint célarchitekturán általában nem magától értetődő individuális memóriacellák olvasása/írása (ehelyett például teljes kép feltöltése, eredménykép lementése, szélső oszlopok/sorok olvasása, stb.)

- **`CountBlackRow(.)`** → visszaadja a paraméterként adott kép utolsó előtti sorában a fekete pixelek (**`+1`**) számát
- **`CountBlackCol(.)`** → visszaadja a paraméterként adott kép 2. oszlopában a fekete pixelek (**`+1`**) számát
- **`CountBlackImg(.)`** → megmondja, hogy a paraméterként adott kép hány fekete pixelt (**`+1`**) tartalmaz összesen

Objektumklasszifikáció - fő szkript: `gyak4.m`

- 5 kép van a `gyak4_images` mappában
  - a `class` vektorba kódoljuk majd az egyes osztályokba tartozás fennállását, helyiértékek szerint
  - a `LoadGT()` függvény visszatérési értéke alapján majd le tudjuk ellenőrizni, hogy helyes osztályokba soroltuk-e a képeket
- a nagy `for`-ciklus egyesével végigiterál a képeken
  - a ciklusmagon belül három különböző blokkban érdemes az egyes osztályba-tartozásokat vizsgálni, template-sorozatokkal
  - a példa kedvéért a 3. osztály (lyukas objektum) detektáló algoritmusát meghagytuk, ez:
    - binarizálás (`'THRES'`)
    - lyukas objektum feltöltés (`'HOLE'` másnéven holefiller)
    - logikai különbségképzéssel (`'LOGDIF'`): a feltöltött mínusz a simán binarizált kép → ha ennek az eredményén maradt még fekete pixel, akkor volt lyukas objektum az eredeti képen