Témalabor:

Gépi tanulás, MI alkalmazása a gyártóiparban (predictive maintenance)

Mesterházi Marcell (TN0VU7)

Dokumentáció

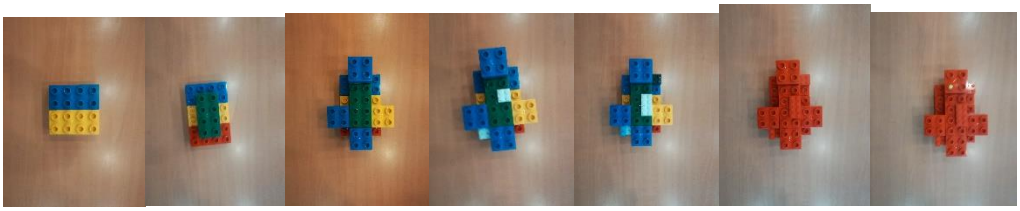
ML_Szivacs_classification:

A megbeszéltek alapján a szivacs képek osztályozását valósítottam meg Jupyterben, egyszerű machine learninggel, random foresttel, ami meglepően ügyesen, 98% pontossággal ((`szivacs_calssification_regi`))

A témalaboron kicsit egyszerűsítettünk a képek tárolásán, a pandas dataframe-mel, ((`szivacs_calssification_temalab_jav`))

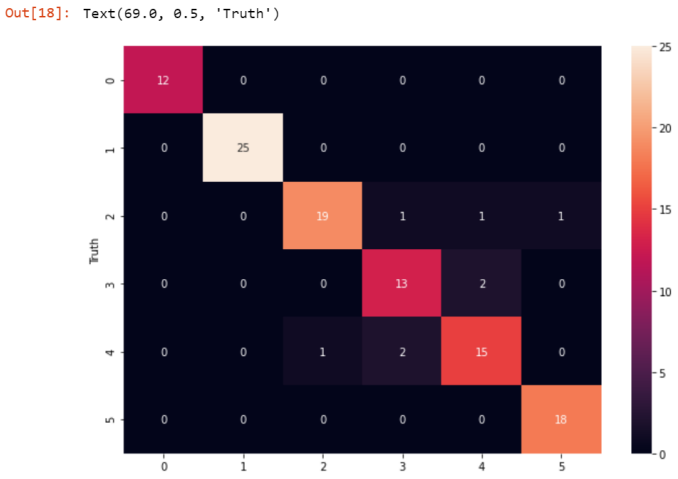
ML_Lego_classification:

Állapotok:



A legokról készül képeket elforgattam, kibalanszoltam az állapotokat (minden állapotról kb 90 kép), most a már megírt szivacson alkalmazott random forestes programmal futtattam a Lego-s adatokra. Először csak az első 6 állapotot osztályoztam.

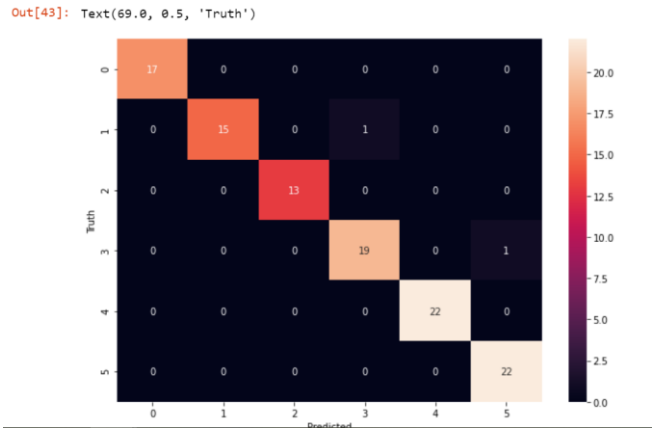
- A pontosság 50x50-es képekkel, random foresttel 92% lett, confusion matrix:



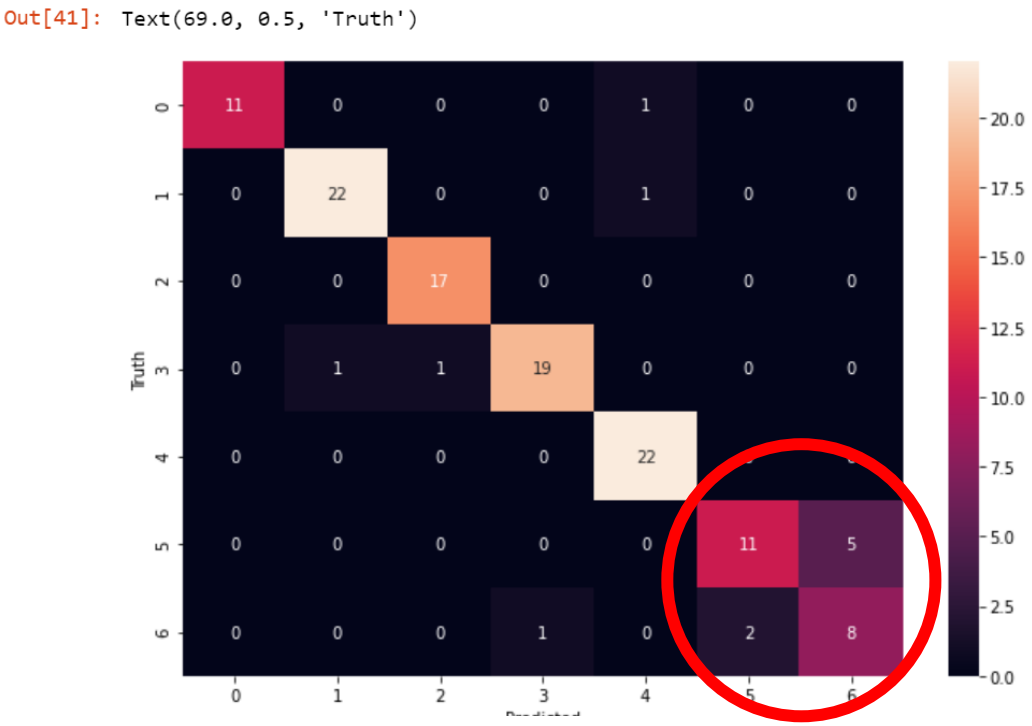

- A pontosság 100x100-as képekkel 93,4% lett(utána már csak 90%), confusion matrix

ML_Lego_classification:

- A pontosság 100x100-as képekkel, logistic regressionnel 98% lett, confusion matrix



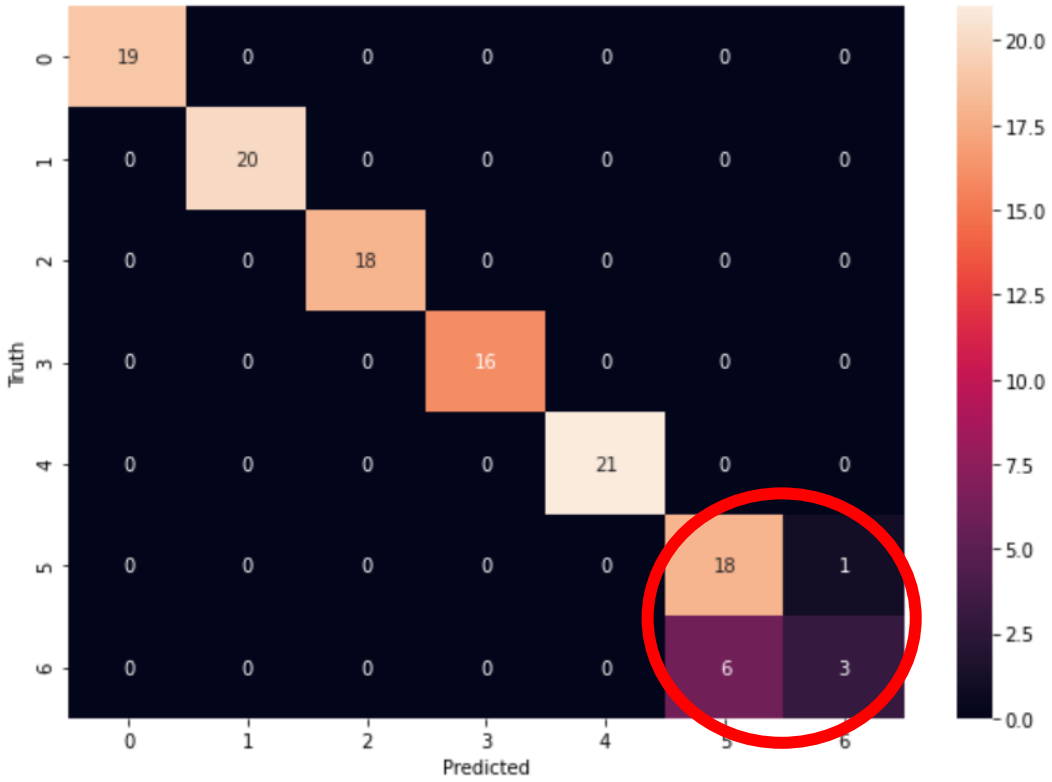

- ha hozzáteszem az phase_6+errors részt, már csak 90%, látszik is, hogy az utolsó két állapotnál pontatlanabb, akkor sem javul, ha megemeljük a képek felbontását, vagy színes képeket olvasunk be:



Deep learning:

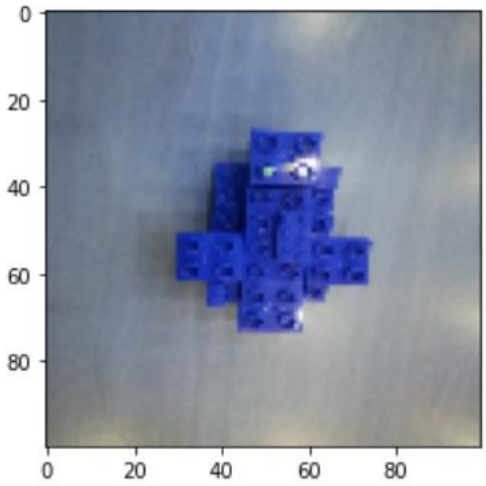
A programot pytorch-chal folytattam, ez 100x100-as fekete-fehér képekkel, 20 batch-al a pontossága hét állapotra tekintve 94%-os volt.
Látható módon, ő is megküzdött az utolsó kettő, nehezebben megkülönböztethető állapottal, de nem annyira, mint machine learning esetén

```
Out[25]: Text(69.0, 0.5, 'Truth')
```



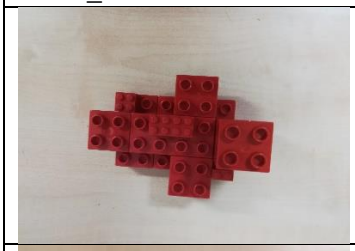
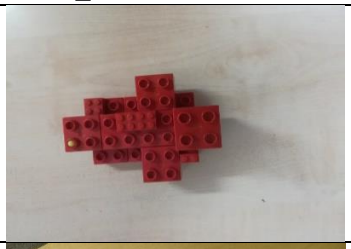
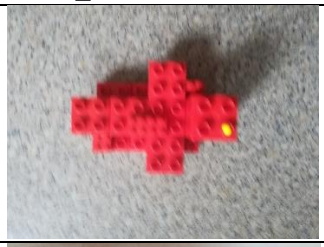
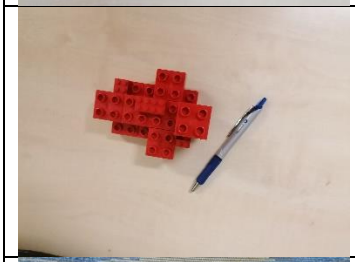
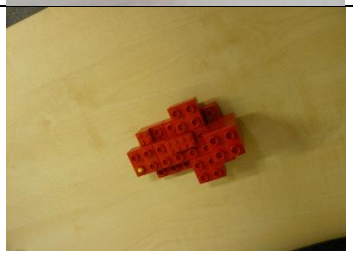
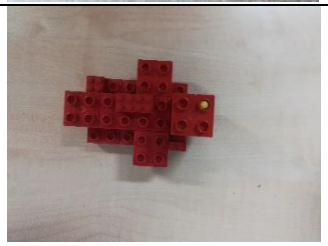
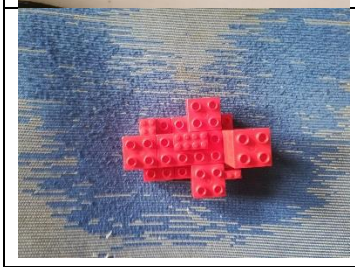
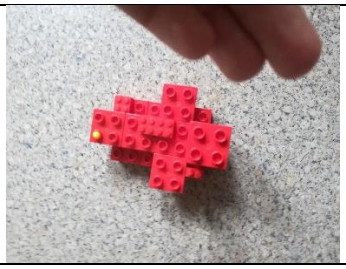
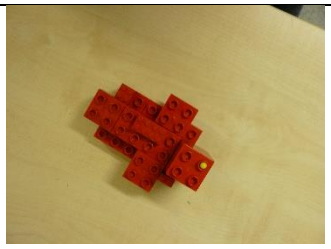

Érdekes volt, hogy amikor a tesztek közül kiírattam a rosszul megtippelt, látszott, hogy néha miért is tévedett a programunk, itt például a visszacsillanó fény megzavarta.

```
Valos:  6 , tippelt:  5
```



# Lego error classification:

Ezeknél a képeknél már sokat változtak a szögek, a fények, a háttér is, különbség ugye a sárga pont helyzete.

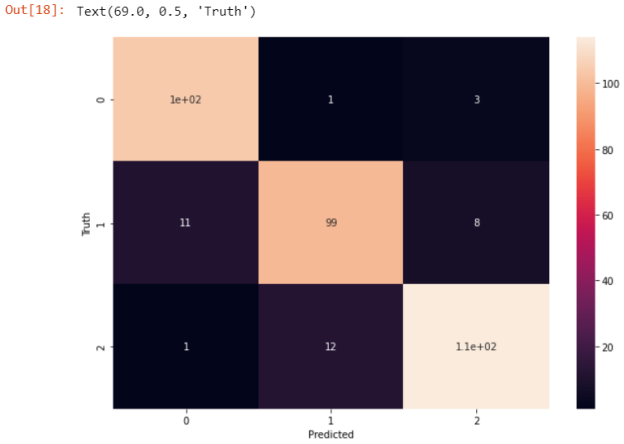
| Phase_1 | Phase_2 | Phase_3 |
|---|---|---|
|  |  |  |
|  |  |  |
|  |  |  |

Meglepő módon a **Randomforest** tartja a versenyt a Deep Learning programokkal, többszöri futtatás után is körülbelül **90%**-os eredményt kapunk.

```
In [19]: model.fit(X_train,Y_train)
Out[19]: RandomForestClassifier()

In [20]: model.score(X_test, Y_test) #pontosság
Out[20]: 0.9121813031161473
```

((további futtatások után ez az eredmény inkább átlagosan 86% körül volt))



A **Pytorch**-os Neural Network 54 epoch után **90%** pontosságú lett:

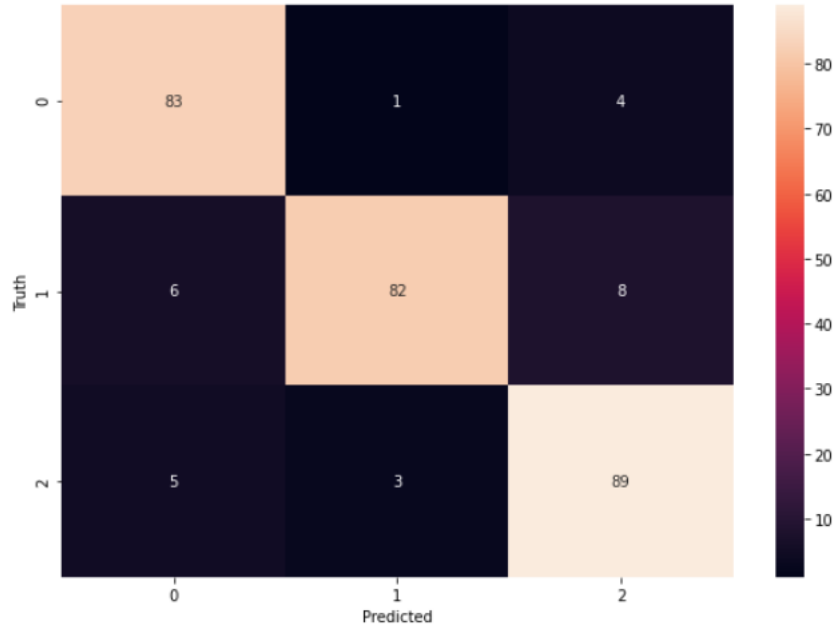
```
Epoch: 51. Loss: 2.0176346879452467e-05
100%|██████████████████████████████████████████████████████████████████████████| 23/23 [01:25<00:00,  3.71s/it]
  0%|                                                                            | 0/23 [00:00<?, ?it/s]
Epoch: 52. Loss: 1.8091088804794708e-06
100%|██████████████████████████████████████████████████████████████████████████| 23/23 [01:23<00:00,  3.64s/it]
  0%|                                                                            | 0/23 [00:00<?, ?it/s]
```

```
print("Accuracy: ", round(correct/total, 3))
```

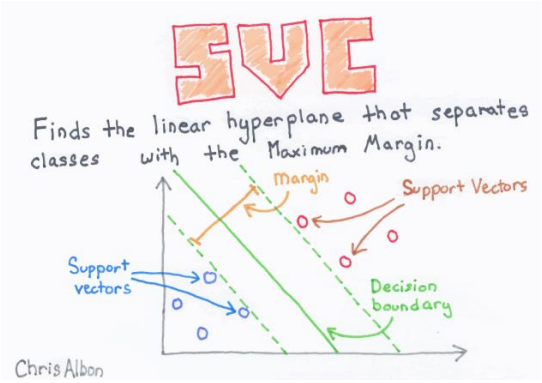
```
100%|██████████████████████████████████████████████████████████████████████████| 281/281 [00:09<00:00, 29.78it/s]
Accuracy:  0.904
```

conf. matrix:

```
Out[15]: Text(69.0, 0.5, 'Truth')
```



A következő hétre különböző más megoldásokat is megpróbáltam:




**SVC** - Support Vector Classifier kipróbálása:

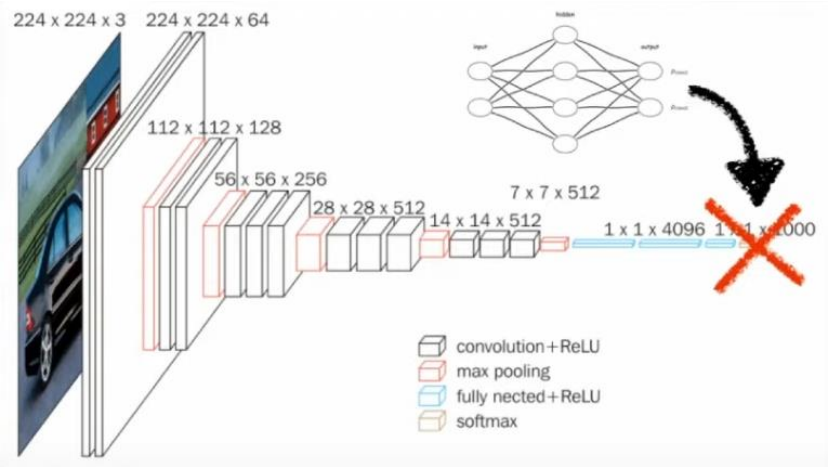
```
In [28]: from sklearn.svm import SVC
         model = SVC(C=1, kernel='poly', gamma = 'auto')
         model.fit(X_train,Y_train)

Out[28]: SVC(C=1, gamma='auto', kernel='poly')

In [29]: model.score(X_test, Y_test) #pontosság

Out[29]: 0.7903682719546742
```

**Pre-trained Neural Network** használata, aminek a lényege az, hogy egy előre, rengeteg képpel betanított neurális hálót használunk fel, amelynek az utolsó layerét változtatjuk a sajátunkra, és ezen a hálón tanítjuk a képek alapján.




```
#importáljuk a már kész NN-t

from torchvision.models import squeezenet1_0

model = squeezenet1_0(pretrained=True)
print(model)
```

```
#módosítjuk az utolsó rétegét

n_classes = 3

model.num_classes = n_classes
model.classifier[1] = nn.Conv2d(512, n_classes, kernel_size=(1,1), stride=(1,1))
```

A neurális háló 10 epoch után **93%**-os pontosságú lett.

```
The testing set accuracy of the network is: 93 %
```

20 epoch után sem lett pontosabb, 92%-os pontossággal dolgozott:

```
The testing set accuracy of the network is: 92 %
```

Nem biztos, hogy ez a legjobb megoldás, meglátom még a másik fajta, VGG16 előretanított neurhálót, de ennek a 20 epochnak a futtatása is 1,5 óra volt Collab-os szuper GPU-val.