Szivacsfázisok megkülönböztetése

A téma kezdeti fázisában még nem tudtuk, hogy pontosan mit kell majd felismernie a programnak ezért azt találtuk ki, hogy építünk magunknak saját modellt ameddig az igaziról nem kapunk képeket. Először legóból akartunk építkezni, de mivel még azon a héten nem állt rendelkezésünkre szivacsból építettünk magunknak különböző fázisokat, amin tudunk gyakorolni. Meglepően jó eredmények jöttek ki még a legegyszerűbb modellek használatánál is, de igazából ez várható is volt a képek nagyon különböznek egymástól sok az eltérés könnyű a megkülönböztetés. Ennél a résznél kellett megismerkednem azzal, hogy hogyan lehet beolvasni a képeket mi a különbség a színes és a fekete fehér beolvasás között és hogy hogyan kell átalakítani a beolvasott képek tömbjét, hogy azt utána a tanítás után fel tudjam használni.

Képek beolvasása:

```
In [16]: DATADIR = "E:\Egyetem\Data\szivacs"
         CATEGORIES = ['phase_1','phase_2','phase_3', 'phase_4']

         IMG_SIZE = 100
         training_data = []

         def create_training_data():
             for category in CATEGORIES:
                 path = os.path.join(DATADIR,category)
                 class_num = CATEGORIES.index(category) + 1
                 num_of_pics_each = 0
                 for img in os.listdir(path):
                     img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE) #fekete fehér képek beolvasása/színes képek beolv
                     new_array = cv2.resize(img_array,(IMG_SIZE,IMG_SIZE)) #kép átméretezése
                     training_data.append([new_array,class_num])
                     num_of_pics_each+=1
                 print(f"In [[{category}]] found {num_of_pics_each} pics") #kiírjuk hány kép volt minden kategóriában

         create_training_data()
```

Először is fent meg kell adni a DATADIR változóban, hogy a mappák, amik tartalmazzák a különböző képeket hol találhatók, majd a CATEGORIES tömbben fel kell sorolni a mappák nevét jelen esetben ugye a 4 fázis. A beolvasás során ezeket egyesítjük és így olvassuk be a képeket minden mappából külön-külön. Egy for ciklussal végig megyünk az összes kategórián, egyesítjük az elérési útvonallal és beolvassuk az ott található összes képet. Majd ezt egy tömbben tároljuk, ahol a kép mellett a kategória indexet is megadjuk innen tudjuk, hogy a kép melyik fázisból származik. Itt fekete fehéren olvasom be a képeket, színes beolvasásnál 3 érték keletkezik ugye az rgb-nek megfelelően, de jelen esetben csak 1. Ez a c2.imread függvényben tudjuk megadni, most cv2.IMREAD_GREYSCALE van megadva ezért fekete fehér. Azért, hogy a különböző méretű képek ne zavarjanak meg és hogy csökkentsük az adat mennyiséget minden képet átméretezek jelen esetben 100x100-as ra.

Ezek után „összerázzuk” a tömböt, hogy a különböző kategóriák ne egymás mellett legyenek mert az a tanulás szempontjából káros, mivel ilyenkor azt tanulja meg a modell, hogy mindig 1-es fázis utána azt, hogy mindig 2-es és így tovább.

Jelen esetben szükség van arra hogy flatten-eljük a tömböt mert ez a modell csak 1d-s tömböt fogad el.

```
In [4]: target = []
        data = []

        for features, label in training_data:
            data.append(features.flatten())  #a flatten szükséges a model betanításához mivel ez a model csak 1d-s tömböt fogad el
            target.append(label)
```

Itt RandomForestClassifier modellt használunk.

```
In [6]: model.fit(X_train,Y_train)
Out[6]: RandomForestClassifier()

In [7]: model.score(X_test, Y_test) #pontosság
Out[7]: 1.0
```

És látszik, hogy egy ilyen egyszerű modellnél 100% pontosságot érünk el tehát Hála Istennek nagyon jól teljesített. Ezt meg is tudjuk tekinteni a Confusion mátrixon:

```
In [8]: # nézzük meg az eredményt a confusion mátrixban
        Y_predicted = model.predict(X_test)

In [9]: from sklearn.metrics import confusion_matrix
        cm = confusion_matrix(Y_test,Y_predicted)
        %matplotlib inline
        import matplotlib.pyplot as plt
        import seaborn as sn
        plt.figure(figsize=(10,7))
        sn.heatmap(cm, annot=True)
        plt.xlabel('Predicted')
        plt.ylabel('Truth')
Out[9]: Text(69.0, 0.5, 'Truth')
```