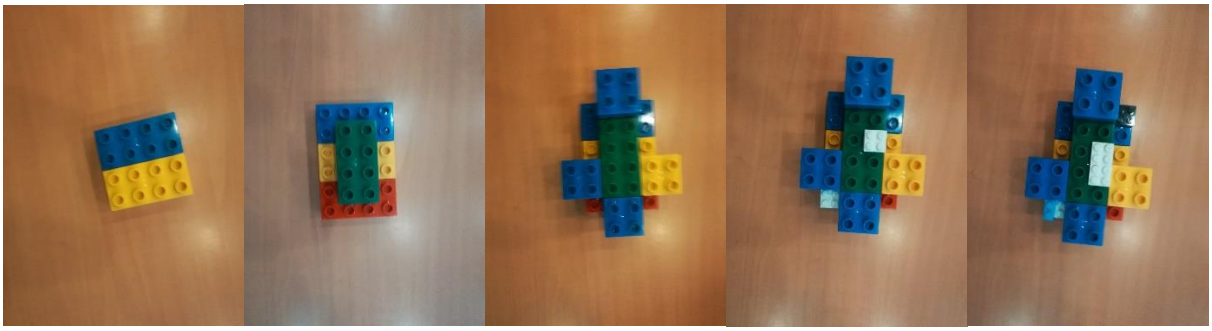
Lego fázis megkülönböztetés

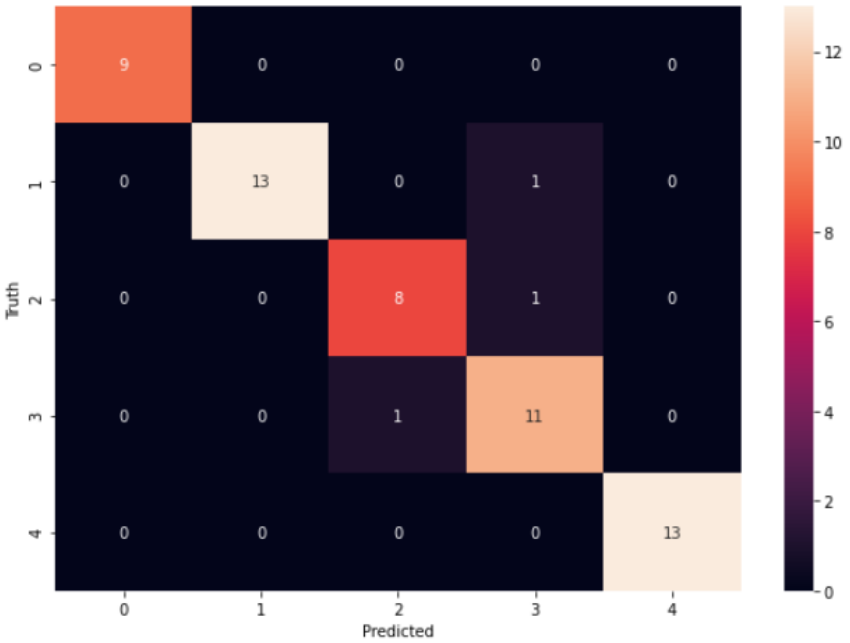


Ezen a héten már sikerült legot szerezni így megépítettük a lego modellünket, ezt az öt fázist különböztetjük meg.

Először a szivacsok megkülönböztetésére készített modellen próbáltam ki és meglepően szép eredményt hozott fekete fehéren is.

```
In [18]: model.score(X_test, Y_test) #pontosság
Out[18]: 0.9473684210526315
```

```
Out[20]: Text(69.0, 0.5, 'Truth')
```

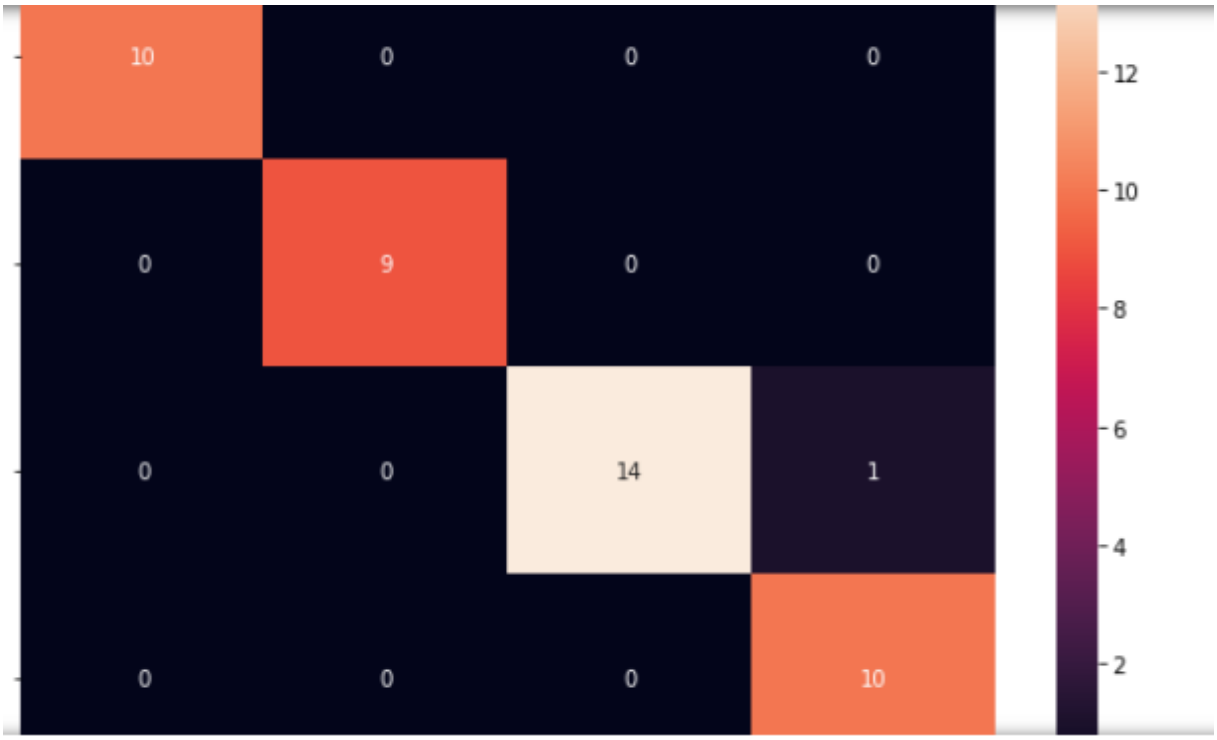


Ezután kipróbáltam, hogy milyen lenne, ha színesbe olvasnám be ehhez el kellet végezni egy két változtatást is.

```
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_COLOR) #fekete fehér képek beolvasása/színes képek beolvasá
            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()
```

Ahogy vártuk Hála Istennek mégjobban teljesített:

```
In [26]: model.score(X_test, Y_test) #pontosság
Out[26]: 0.9772727272727273
```



Ezek után tovább szerettem volna haladni a témában mert ez az egyszerű modell nem igazán ideális kép felismerésre, ha bonyolultabb képeket kell használni akkor szinte biztos, hogy nem fog ilyen jól teljesít, sőt még talán elfogadhatóan sem. Ezért elkezdtem megismerkedni a neurális hálók témakörével. Először kipróbáltam, hogy hogy teljesít egy egyszerű neurális háló csak pár layer-el. Utána megismerkedtem a konvolúciós neurális hálókkal is de azt már egy következő dokumentációban rögzítettem.

Egyszerű neurális háló:

Tensorflow\keras segítségével építettem meg a hálót:

```
In [*]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(data,target, test_size=0.1)

        X_train = np.array(X_train).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
        y_train = np.array(y_train)
        X_tarin = X_train/255
```

```
In [*]: X_test = np.array(X_test).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
        X_test = X_test/255
```

Szükséges a tömbök átformázása a tanításhoz a 3-as szám azt jelenti, hogy színes.

```
In [11]: import tensorflow as tf
         from tensorflow.keras.optimizers import RMSprop

         model = tf.keras.models.Sequential()

         model.add(tf.keras.layers.Flatten())

         model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))
         model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))
         model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))
         model.add(tf.keras.layers.Dense(6, activation = tf.nn.softmax))

         model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
         model.fit(X_train, y_train, epochs = 20)
```

Ezt a hálót építettem csupán dense layer-ek vannak benne. És ennek meg is volt a hatása.

```
Epoch 1/20
16/16 [==============================] - 0s 15ms/step - loss: 1513.7406 - accuracy: 0.2235
Epoch 2/20
16/16 [==============================] - 0s 15ms/step - loss: 407.5438 - accuracy: 0.3059
Epoch 3/20
16/16 [==============================] - 0s 15ms/step - loss: 218.1035 - accuracy: 0.3137
Epoch 4/20
16/16 [==============================] - 0s 15ms/step - loss: 101.2317 - accuracy: 0.5020
Epoch 5/20
16/16 [==============================] - 0s 15ms/step - loss: 76.0202 - accuracy: 0.5784
Epoch 6/20
16/16 [==============================] - 0s 15ms/step - loss: 50.7759 - accuracy: 0.6588
Epoch 7/20
16/16 [==============================] - 0s 15ms/step - loss: 42.7626 - accuracy: 0.6608
Epoch 8/20
16/16 [==============================] - 0s 14ms/step - loss: 46.4260 - accuracy: 0.6098
Epoch 9/20
16/16 [==============================] - 0s 14ms/step - loss: 33.3016 - accuracy: 0.7667
Epoch 10/20
16/16 [==============================] - 0s 15ms/step - loss: 8.7465 - accuracy: 0.8431
Epoch 11/20
16/16 [==============================] - 0s 14ms/step - loss: 15.9539 - accuracy: 0.7922
Epoch 12/20
16/16 [==============================] - 0s 14ms/step - loss: 17.0282 - accuracy: 0.8000
Epoch 13/20
16/16 [==============================] - 0s 14ms/step - loss: 50.1754 - accuracy: 0.6627
Epoch 14/20
16/16 [==============================] - 0s 15ms/step - loss: 11.4322 - accuracy: 0.8686
Epoch 15/20
16/16 [==============================] - 0s 14ms/step - loss: 8.1430 - accuracy: 0.8392
Epoch 16/20
16/16 [==============================] - 0s 14ms/step - loss: 1.3859 - accuracy: 0.9608
Epoch 17/20
16/16 [==============================] - 0s 14ms/step - loss: 0.8170 - accuracy: 0.9706
Epoch 18/20
16/16 [==============================] - 0s 15ms/step - loss: 7.2933 - accuracy: 0.8882
Epoch 19/20
16/16 [==============================] - 0s 14ms/step - loss: 11.7964 - accuracy: 0.8412
Epoch 20/20
16/16 [==============================] - 0s 14ms/step - loss: 2.4119 - accuracy: 0.9431
```

]: <tensorflow.python.keras.callbacks.History at 0x147533e6a60>

Az eredményen jól látszik hogy annyire nem rossz de nagyon ugrál a pontosság ami nem sok jót jelent ezért tovább mentem a konvolúciós neurális hálók irányába ami Hála Istennek nagyon jól teljesített.

Konvalúciós neurális háló:

Nagyon szép eredményt adott hamar elérte a pontosság a 100%ot és nem is csökkent a tanítás végéig sem. Itt max-pooling módszert is alkalmaztam, amit szintén a másik dokumentációban dokumentáltam részletesebben.

In [10]:
```
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(16,(3,3),activation ='relu', input_shape =(IMG_SIZE,IMG_SIZE,3)))
model.add(tf.keras.layers.MaxPool2D(2,2))
model.add(tf.keras.layers.Conv2D(32,(3,3),activation ='relu'))
model.add(tf.keras.layers.MaxPool2D(2,2))
model.add(tf.keras.layers.Conv2D(64,(3,3),activation ='relu'))
model.add(tf.keras.layers.MaxPool2D(2,2))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(512, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(6, activation = tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs = 20)
```

```
Epoch 1/20
16/16 [==============================] - 2s 107ms/step - loss: 85.1656 - accuracy: 0.3373
Epoch 2/20
16/16 [==============================] - 2s 105ms/step - loss: 0.6648 - accuracy: 0.7961
Epoch 3/20
16/16 [==============================] - 2s 104ms/step - loss: 0.0568 - accuracy: 0.9882
Epoch 4/20
16/16 [==============================] - 2s 103ms/step - loss: 0.0058 - accuracy: 1.0000
Epoch 5/20
16/16 [==============================] - 2s 105ms/step - loss: 0.0011 - accuracy: 1.0000
Epoch 6/20
16/16 [==============================] - 2s 104ms/step - loss: 3.6941e-04 - accuracy: 1.0000
Epoch 7/20
16/16 [==============================] - 2s 104ms/step - loss: 2.2556e-04 - accuracy: 1.0000
Epoch 8/20
16/16 [==============================] - 2s 107ms/step - loss: 1.6787e-04 - accuracy: 1.0000
Epoch 9/20
16/16 [==============================] - 2s 111ms/step - loss: 1.3043e-04 - accuracy: 1.0000
Epoch 10/20
16/16 [==============================] - 2s 106ms/step - loss: 1.0305e-04 - accuracy: 1.0000
Epoch 11/20
16/16 [==============================] - 2s 106ms/step - loss: 8.4211e-05 - accuracy: 1.0000
Epoch 12/20
16/16 [==============================] - 2s 109ms/step - loss: 7.2490e-05 - accuracy: 1.0000
Epoch 13/20
16/16 [==============================] - 2s 108ms/step - loss: 5.9661e-05 - accuracy: 1.0000
Epoch 14/20
16/16 [==============================] - 2s 110ms/step - loss: 5.2445e-05 - accuracy: 1.0000
Epoch 15/20
16/16 [==============================] - 2s 107ms/step - loss: 4.4213e-05 - accuracy: 1.0000
Epoch 16/20
16/16 [==============================] - 2s 105ms/step - loss: 3.7209e-05 - accuracy: 1.0000
Epoch 17/20
16/16 [==============================] - 2s 104ms/step - loss: 3.1690e-05 - accuracy: 1.0000
Epoch 18/20
16/16 [==============================] - 2s 106ms/step - loss: 2.7400e-05 - accuracy: 1.0000
Epoch 19/20
16/16 [==============================] - 2s 105ms/step - loss: 2.3531e-05 - accuracy: 1.0000
Epoch 20/20
16/16 [==============================] - 2s 105ms/step - loss: 2.0694e-05 - accuracy: 1.0000
```

Out[10]: <tensorflow.python.keras.callbacks.History at 0x1474ef71b80>