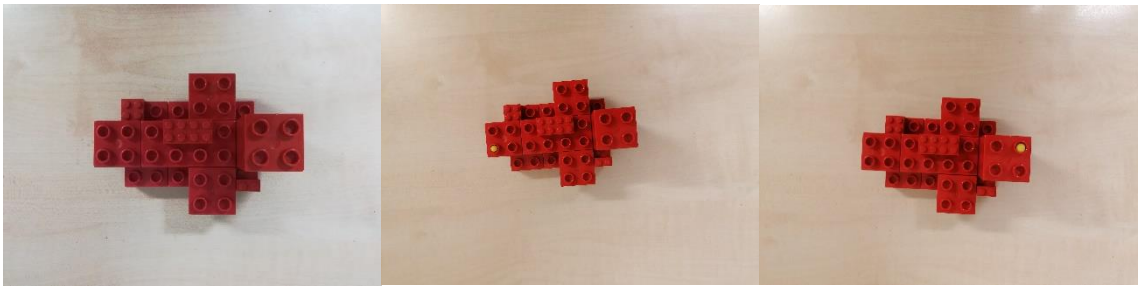
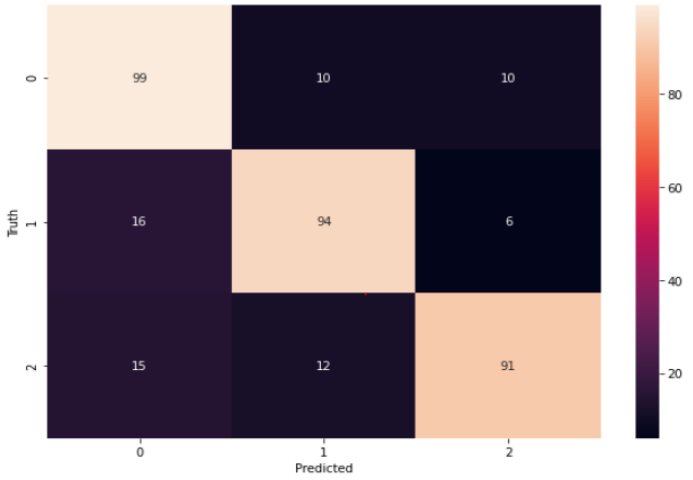
Error Lego



Feladat ennek a 3 fázisnak a megkülönböztetése/felismerése.

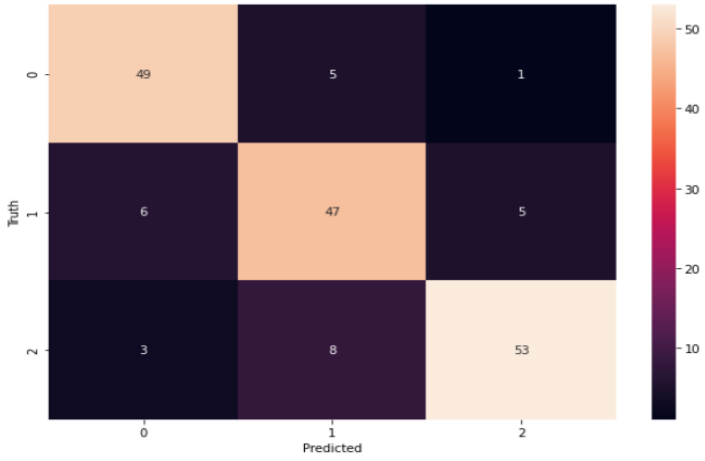
RandomForestClassifier: Először beolvassuk a képeket még nem színesen és megnézzük, hogy teljesít a model.

Először a test_size = 0.2: 0.8-as pontosságot ért el a model.

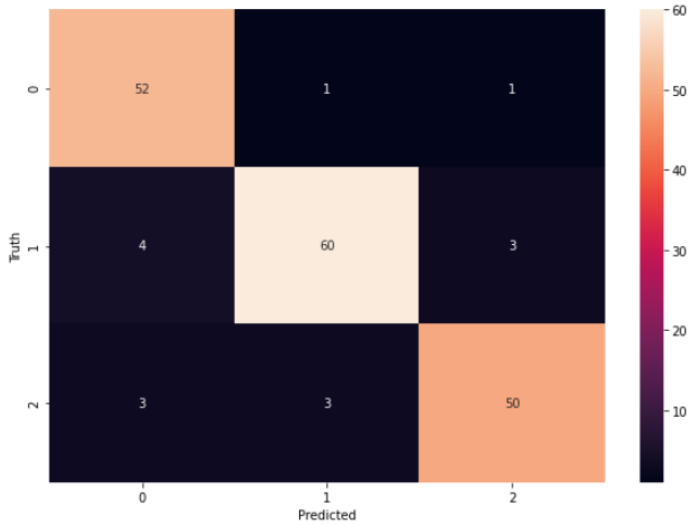


Nem olyan rossz eredmény, de nagyobb pontosság szükséges.

test_size = 01: 0.84 így jobban teljesített a model:



Most színesben olvassuk be a képeket (így más jóval hosszabb a beolvasás**):** úgy gondolom, hogy így pontosabb kellene, hogy legyen a model mivel így talán élesebbek a különbségek.
Hála Istennek igaz volt a sejtés így a model már 0.915 pontosságot ért el.



TensorFlow_Kreas:

Sequential model:

Először egy egyszerű neurális hálót próbáltam ki csak Dense Layer-ek el de ezek nem hoztak jó megoldást ezért további Layer-ekre van szükség a pontosság növeléséhez. Most a pontosság csak 0.64 maximum.

```
Epoch 1/19
50/50 [==============================] - 1s 16ms/step - loss: 863.7296 - accuracy: 0.3060
Epoch 2/19
50/50 [==============================] - 1s 16ms/step - loss: 334.5793 - accuracy: 0.3710
Epoch 3/19
50/50 [==============================] - 1s 16ms/step - loss: 97.5198 - accuracy: 0.4038
Epoch 4/19
50/50 [==============================] - 1s 16ms/step - loss: 211.4770 - accuracy: 0.3729
Epoch 5/19
50/50 [==============================] - 1s 16ms/step - loss: 65.7130 - accuracy: 0.4366
Epoch 6/19
50/50 [==============================] - 1s 17ms/step - loss: 73.7226 - accuracy: 0.4391
Epoch 7/19
50/50 [==============================] - 1s 16ms/step - loss: 54.9884 - accuracy: 0.4486
Epoch 8/19
50/50 [==============================] - 1s 16ms/step - loss: 42.8771 - accuracy: 0.4707
Epoch 9/19
50/50 [==============================] - 1s 16ms/step - loss: 18.7084 - accuracy: 0.5830
Epoch 10/19
50/50 [==============================] - 1s 16ms/step - loss: 62.4507 - accuracy: 0.4341
Epoch 11/19
50/50 [==============================] - 1s 16ms/step - loss: 59.9040 - accuracy: 0.4246
Epoch 12/19
50/50 [==============================] - 1s 16ms/step - loss: 24.5309 - accuracy: 0.5224
Epoch 13/19
50/50 [==============================] - 1s 16ms/step - loss: 21.0005 - accuracy: 0.5344: 0s - loss: 23.2221 - accuracy: 0.
Epoch 14/19
50/50 [==============================] - 1s 16ms/step - loss: 15.0646 - accuracy: 0.5628
Epoch 15/19
50/50 [==============================] - 1s 16ms/step - loss: 15.6249 - accuracy: 0.5584
Epoch 16/19
50/50 [==============================] - 1s 16ms/step - loss: 9.9871 - accuracy: 0.5779
Epoch 17/19
50/50 [==============================] - 1s 16ms/step - loss: 8.4212 - accuracy: 0.5981
Epoch 18/19
50/50 [==============================] - 1s 18ms/step - loss: 5.9526 - accuracy: 0.6461
Epoch 19/19
50/50 [==============================] - 1s 17ms/step - loss: 8.4884 - accuracy: 0.5804
```

Convolutional neural network tensorflow:

Itt kétdimenziós konvolúciós neurális hálót használtam. Meg is látszik az eredményen mivel a végére a model elérte a 1-es pontosságot.

Magyarázat: Először is a szükséges importok a programhoz:

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import os
        import cv2
        import random
        from sklearn.model_selection import train_test_split
```

Ezekre szükség van a képek beolvasásához, összekeveréséhez és train és test adatok sztválasztásához.

Következő lépésben beolvassuk a képeket színesben és közben átméretezzük 100X100-as méretűre mert ez is elég jó minőségű ahhoz hogy a model meg tudja különböztetni.

```
DATADIR = "E:\Egyetem\Data\error"
CATEGORIES = ['1','2','3']

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 beolvasás
            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()
```

```
In [[1]] found 598 pics
In [[2]] found 579 pics
In [[3]] found 585 pics
```

Láthatjuk kiírva, hogy az egyes kategóriákról hány kép van. Ezek után összekevertem az adatokat, hogy ne sorban legyen mert akkor a model azt tanulja meg hogy például mindig csak 1-es kategória utána meg azt hogy mindig csak 2-es stb.

```
target = []
data = []

for features, label in training_data:
    data.append(features)
    target.append(label)
```

Itt valósult meg az adataink szétválasztása külön data és target-re. Ezeket szétválasztottam külön teszt és train adatokra.

```
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
```

Utána a két train adatot reshape-eltem mert a modelünknek megfelelő alakú numpy tömbre van szüksége. Utána az X_train tömböt osztottam 255-el, hogy benne csak 0->1-ig legyenek értékek mert így a model könnyebben és jobban tanul belőle. És ezután következett a neurális háló megalkotása.

```
import tensorflow as tf
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(4, activation = tf.nn.softmax))

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

Az első Layer-nél meg kell adni az adat formáját vagyis az imput_shape-et ami itt ugye a kép méret szor a kép méret és utána van egy 3 azért mert színes képekről van szó és így egy pixelnek 3 értéke van: red green blue. Itt ’adam’ optimizert használtam és a loss ’categorical’ mivel három fázisunk van.

```
Epoch 1/20
50/50 [==============================] - 6s 118ms/step - loss: 18.7856 - accuracy: 0.4006
Epoch 2/20
50/50 [==============================] - 6s 118ms/step - loss: 0.7617 - accuracy: 0.6498
Epoch 3/20
50/50 [==============================] - 6s 118ms/step - loss: 0.4783 - accuracy: 0.8063
Epoch 4/20
50/50 [==============================] - 6s 119ms/step - loss: 0.2720 - accuracy: 0.8984
Epoch 5/20
50/50 [==============================] - 6s 118ms/step - loss: 0.1637 - accuracy: 0.9451
Epoch 6/20
50/50 [==============================] - 6s 118ms/step - loss: 0.0654 - accuracy: 0.9811
Epoch 7/20
50/50 [==============================] - 6s 118ms/step - loss: 0.0200 - accuracy: 0.9975
Epoch 8/20
50/50 [==============================] - 6s 118ms/step - loss: 0.0119 - accuracy: 0.9987
Epoch 9/20
50/50 [==============================] - 6s 118ms/step - loss: 0.0041 - accuracy: 1.0000
Epoch 10/20
50/50 [==============================] - 6s 118ms/step - loss: 0.0016 - accuracy: 1.0000
Epoch 11/20
50/50 [==============================] - 6s 119ms/step - loss: 8.2252e-04 - accuracy: 1.0000
Epoch 12/20
50/50 [==============================] - 6s 120ms/step - loss: 6.0725e-04 - accuracy: 1.0000
Epoch 13/20
50/50 [==============================] - 6s 120ms/step - loss: 4.2924e-04 - accuracy: 1.0000
Epoch 14/20
50/50 [==============================] - 6s 120ms/step - loss: 3.1036e-04 - accuracy: 1.0000
Epoch 15/20
50/50 [==============================] - 6s 121ms/step - loss: 2.2279e-04 - accuracy: 1.0000
Epoch 16/20
50/50 [==============================] - 6s 120ms/step - loss: 1.6721e-04 - accuracy: 1.0000
Epoch 17/20
50/50 [==============================] - 6s 119ms/step - loss: 1.2501e-04 - accuracy: 1.0000
Epoch 18/20
50/50 [==============================] - 6s 118ms/step - loss: 9.6798e-05 - accuracy: 1.0000
Epoch 19/20
50/50 [==============================] - 6s 119ms/step - loss: 7.3954e-05 - accuracy: 1.0000
Epoch 20/20
50/50 [==============================] - 6s 118ms/step - loss: 5.8154e-05 - accuracy: 1.0000
```

Hála Istennek elég szép eredményeket produkált a model már a kilencedik epoch-nál láthattuk, hogy eléri az 1-es pontosságot, ami a 100%. És utána sem romlott el.

RMSprop optimizer:

Most kipróbáltam egy másik optimizert ami széntén jó eredményeket hozott de 20 epoch alatt maximum 0.98 volt a pontossága szóval kicsit rosszabbúl teljesített mint az Adam.

```
Epoch 1/20
50/50 [==============================] - 6s 126ms/step - loss: 39.0716 - accuracy: 0.4625
Epoch 2/20
50/50 [==============================] - 6s 127ms/step - loss: 1.5421 - accuracy: 0.4808
Epoch 3/20
50/50 [==============================] - 7s 133ms/step - loss: 1.9273 - accuracy: 0.5047
Epoch 4/20
50/50 [==============================] - 6s 126ms/step - loss: 2.9550 - accuracy: 0.5356
Epoch 5/20
50/50 [==============================] - 6s 126ms/step - loss: 1.9399 - accuracy: 0.5981
Epoch 6/20
50/50 [==============================] - 6s 126ms/step - loss: 0.6997 - accuracy: 0.7476
Epoch 7/20
50/50 [==============================] - 6s 126ms/step - loss: 1.6592 - accuracy: 0.6568
Epoch 8/20
50/50 [==============================] - 6s 125ms/step - loss: 0.7790 - accuracy: 0.7968
Epoch 9/20
50/50 [==============================] - 6s 124ms/step - loss: 1.3139 - accuracy: 0.8517
Epoch 10/20
50/50 [==============================] - 7s 134ms/step - loss: 0.1642 - accuracy: 0.9495
Epoch 11/20
50/50 [==============================] - 7s 137ms/step - loss: 1.1178 - accuracy: 0.8688
Epoch 12/20
50/50 [==============================] - 6s 128ms/step - loss: 0.0738 - accuracy: 0.9760
Epoch 13/20
50/50 [==============================] - 6s 126ms/step - loss: 0.9465 - accuracy: 0.9514
Epoch 14/20
50/50 [==============================] - 6s 129ms/step - loss: 0.3118 - accuracy: 0.9634
Epoch 15/20
50/50 [==============================] - 6s 126ms/step - loss: 0.0930 - accuracy: 0.9779
Epoch 16/20
50/50 [==============================] - 6s 126ms/step - loss: 0.0840 - accuracy: 0.9785
Epoch 17/20
50/50 [==============================] - 6s 126ms/step - loss: 0.4295 - accuracy: 0.9603
Epoch 18/20
50/50 [==============================] - 7s 132ms/step - loss: 0.0078 - accuracy: 0.9981
Epoch 19/20
50/50 [==============================] - 6s 128ms/step - loss: 0.1649 - accuracy: 0.9798
Epoch 20/20
50/50 [==============================] - 6s 126ms/step - loss: 0.2731 - accuracy: 0.9804
```