Project: Detecting the types of Uno cards.

Student: Sela Shapira M00956836.

this notebook includes the following codes:

1 - Code for detecting the color of a given card.

2 - Code for detecting the number of a given card.

3 - Code for detecting the type of the card (both color and number).

4 - Code for detecting the type of the card using a camera (openCV).

Code for detecting the color of a given card.

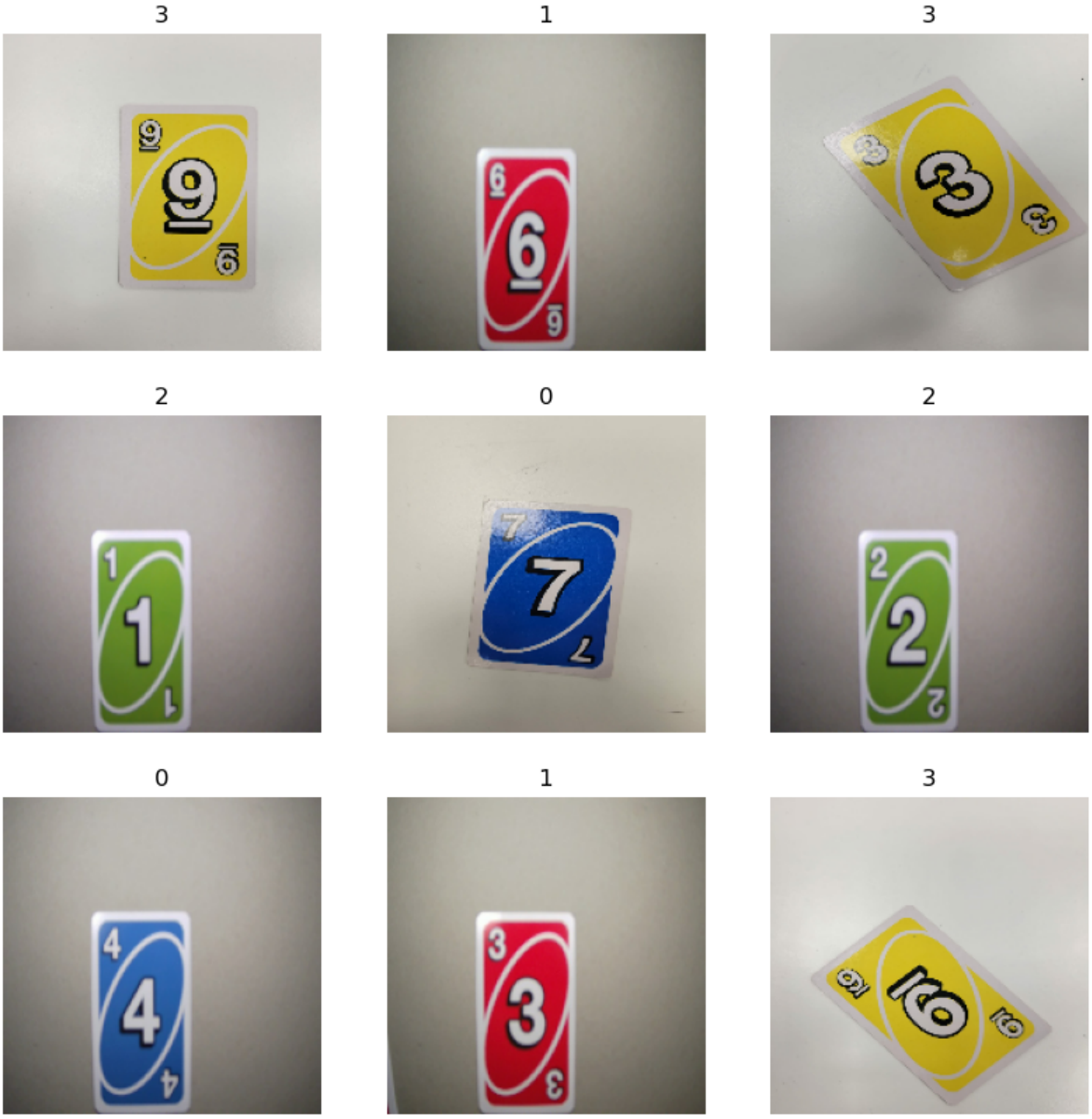
In [2]:
```
import os
import numpy as np
import keras
from keras import layers
from tensorflow import data as tf_data
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import Sequential, datasets
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
!pip install pydot
```

```
WARNING:tensorflow:From C:\Users\selas\anaconda3\Lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

Requirement already satisfied: pydot in c:\users\selas\anaconda3\lib\site-packages
(1.4.2)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\selas\anaconda3\lib\si
te-packages (from pydot) (3.0.9)
```

In [14]:
```
image_size = (180, 180)
batch_size = 128

train_ds, val_ds = keras.utils.image_dataset_from_directory(
    "University/Dataset",
    validation_split=0.2,
    subset="both",
    seed=13,
    image_size=image_size,
    batch_size=batch_size,
)
```

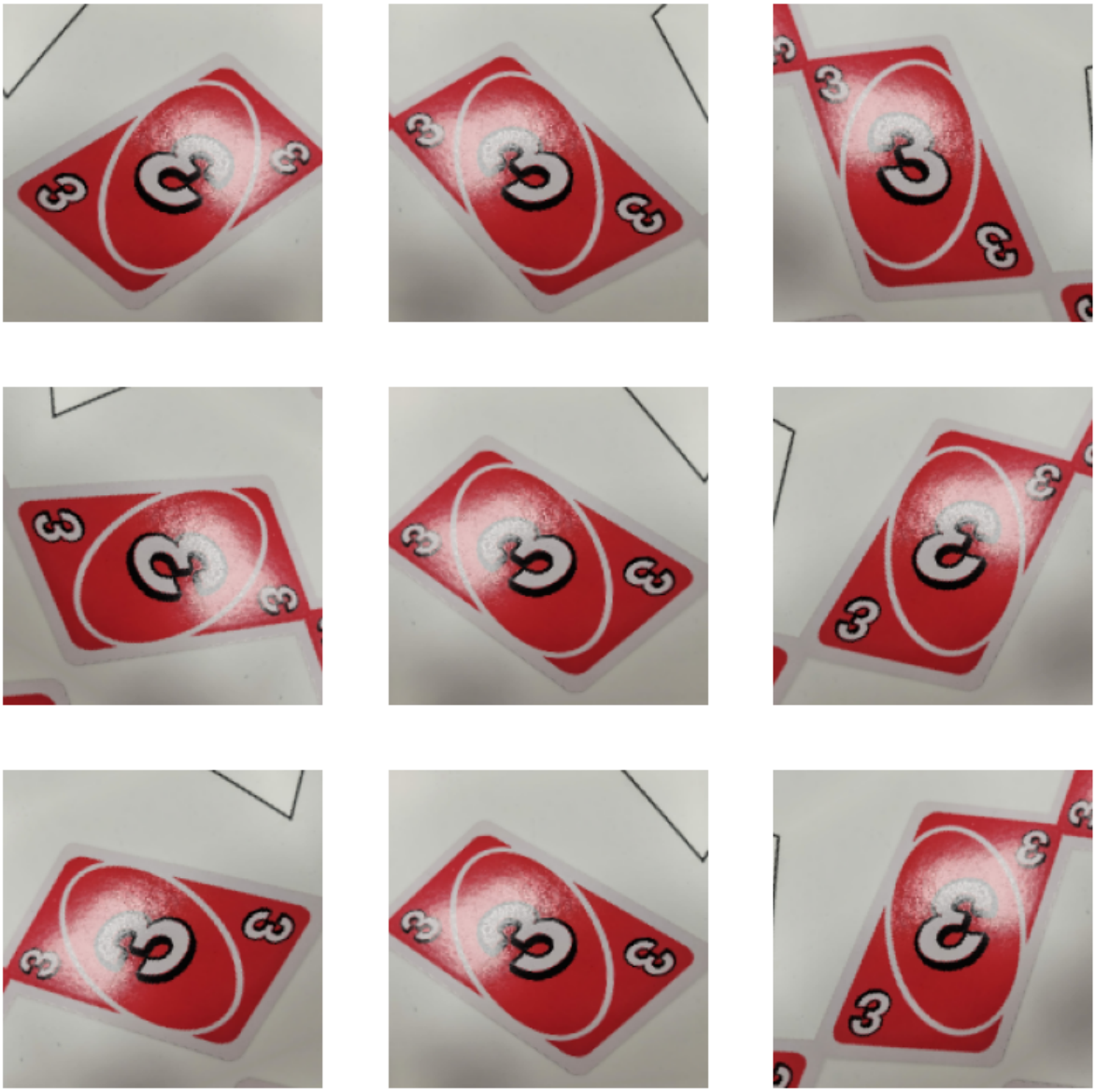
```
Found 182 files belonging to 4 classes.
Using 146 files for training.
Using 36 files for validation.
```

In [15]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(images[i]).astype("uint8"))
        plt.title(int(labels[i]))
        plt.axis("off")
```



In [16]:
```
data_augmentation_layers = [
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
]


def data_augmentation(images):
    for layer in data_augmentation_layers:
        images = layer(images)
    return images
```

In [17]:
```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(augmented_images[0]).astype("uint8"))
        plt.axis("off")
```



In [18]:
```
# Apply `data_augmentation` to the training images.
train_ds = train_ds.map(
    lambda img, label: (data_augmentation(img), label),
    num_parallel_calls=tf_data.AUTOTUNE,
)
# Prefetching samples in GPU memory helps maximize GPU utilization.
train_ds = train_ds.prefetch(tf_data.AUTOTUNE)
val_ds = val_ds.prefetch(tf_data.AUTOTUNE)
```

In [19]:
```
# before normalization
for image, label in train_ds.take(1):  # Take one batch from the dataset
    print(image)

train_ds = train_ds.map(lambda x, y: (x / 255, y))
val_ds = val_ds.map(lambda x, y: (x / 255, y))

# after normalization
for image, label in train_ds.take(1):  # Take one batch from the dataset
    print(image)
```

```
tf.Tensor(
[[[176.82317  148.03293  132.14001 ]
  [173.9921   144.49385  125.15598 ]
  [174.67834  146.24292  125.16182 ]
  ...
  [131.78789   91.98819   66.89101 ]
  [127.00637   88.709366  62.919518]
  [137.99173  100.03912   74.35667 ]]

 [[177.81795  155.28397  139.52873 ]
  [181.90019  158.96402  140.81723 ]
  [179.85066  154.85562  136.40723 ]
  ...
  [128.15646   87.35891   63.555264]
  [127.05505   87.05505   62.055054]
  [129.70909   89.55424   64.63165 ]]

 [[169.0699   137.6117   117.67086 ]
  [175.85959  144.88972  124.5085  ]
  [175.51123  145.57672  125.371544]
  ...
  [125.74949   86.77048   62.107677]
  [121.832     83.08935   57.67023 ]
  [124.75371   85.16048   60.109253]]

 ...

 [[225.89328  223.21286  212.60278 ]
  [217.09607  207.752    187.6875  ]
  [214.98167  203.49915  182.52197 ]
  ...
  [212.11765  228.11765  241.11765 ]
  [211.95868  227.95868  240.95868 ]
  [211.84796  227.84796  240.84796 ]]

 [[229.83896  228.4718   222.19714 ]
  [225.31873  221.81165  209.19833 ]
  [222.29752  216.31177  202.36926 ]
  ...
  [212.12508  228.43558  241.28033 ]
  [210.7042   226.7042   239.7042  ]
  [213.2647   229.2647   242.42458 ]]

 [[229.65756  227.56519  218.17953 ]
  [226.63925  223.4888   211.77759 ]
  [224.75772  219.24173  205.7251  ]
  ...
  [207.28622  217.21613  225.39105 ]
  [208.51219  220.62799  230.66925 ]
  [210.58813  224.64182  237.3302  ]]]


[[[151.6988    20.698801  36.6988  ]
  [151.44443   20.444431  36.444427]
  [149.45505   18.45505   34.45505 ]
  ...
  [160.17172   33.17171   50.17171 ]
  [160.05948   33.059475  50.05947 ]
  [159.95038   32.95038   49.950382]]

 [[150.47086   19.470848  34.806786]
  [150.26051   19.260494  35.26049 ]
  [149.6229    18.669214  34.65377 ]
  ...
```

```
  [160.4797    33.479725  51.183067]
  [161.51299   34.512985  52.034447]
  [158.4499    31.449894  48.449898]]

 [[151.11755   19.39716   33.658283]
  [150.72618   19.711927  34.96437 ]
  [150.89291   19.892923  35.581608]
  ...
  [158.29877   31.298761  50.26055 ]
  [161.999     34.998978  53.998978]
  [160.66113   33.66112   51.05312 ]]

 ...

 [[196.84128  216.57439  230.57608 ]
  [196.6326   218.6326   232.6326  ]
  [199.84555  221.84555  235.84555 ]
  ...
  [198.69083   58.84475   84.86749 ]
  [200.20654   60.56998   85.503235]
  [201.03421   61.07897   87.05659 ]]

 [[198.70398  220.08553  235.08391 ]
  [198.68246  220.66864  234.67325 ]
  [199.28386  221.25493  235.26543 ]
  ...
  [199.0222    59.02218   85.05585 ]
  [199.81026   59.792778  85.55328 ]
  [197.73216   58.99302   84.34383 ]]

 [[196.81114  218.64282  233.14392 ]
  [197.67264  219.67264  233.67264 ]
  [197.09773  219.09773  233.09773 ]
  ...
  [197.62093   56.012924  82.012924]
  [199.68744   57.687447  83.68745 ]
  [196.42352   54.85593   80.74783 ]]]


[[[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]

 [[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]

 [[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]

 ...
```

```
 [[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]

 [[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]

 [[255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  ...
  [255.       255.       255.      ]
  [255.       255.       255.      ]
  [255.       255.       255.      ]]]


...


[[[205.01175  206.01173  200.01173 ]
  [205.       206.       200.      ]
  [205.85316  205.14684  200.      ]
  ...
  [208.52698  208.2182   202.87259 ]
  [209.87325  208.64537  203.49004 ]
  [209.89833  208.94513  203.906   ]]

 [[205.17072  206.       200.08536 ]
  [205.       206.       200.      ]
  [205.13492  205.95807  200.0465  ]
  ...
  [207.70038  208.70038  202.70038 ]
  [208.31969  208.40062  202.86014 ]
  [209.47269  208.47269  203.47269 ]]

 [[206.35417  205.9108   200.63248 ]
  [205.98895  206.       200.49448 ]
  [206.39227  205.80379  200.59802 ]
  ...
  [207.05219  208.05219  202.05219 ]
  [207.6961   208.55339  202.62476 ]
  [208.16354  208.86987  203.01671 ]]

 ...

 [[186.85316  187.85316  181.85316 ]
  [188.05441  187.55339  181.92865 ]
  [189.44055  187.05219  182.      ]
  ...
  [147.36118  142.36118  138.36118 ]
  [147.6718   142.6718   138.6718  ]
  [147.81113  142.81113  138.81113 ]]

 [[185.9583   186.9583   180.9583  ]
```

```
  [187.94534  186.78348  181.32394 ]
  [189.29962  186.29962  181.29962 ]
  ...
  [147.77539  142.77539  138.77539 ]
  [148.32394  143.32394  139.32394 ]
  [148.23514  143.23514  139.31592 ]]

 [[185.35883  186.24913  180.30397 ]
  [187.20221  186.87158  181.38156 ]
  [188.60043  186.65439  181.65439 ]
  ...
  [147.09254  142.09254  138.09254 ]
  [146.77066  141.77066  137.77066 ]
  [147.71777  142.71774  138.73422 ]]]


[[[155.63478  150.63478  146.5596  ]
  [154.03435  149.03433  144.59138 ]
  [153.2894   148.32553  143.35742 ]
  ...
  [151.4824   147.4824   138.4824  ]
  [148.84872  144.84872  135.84874 ]
  [148.44879  144.44879  135.44878 ]]

 [[156.27962  151.27962  148.0548  ]
  [154.46863  149.46863  146.05994 ]
  [153.52068  148.59053  144.60141 ]
  ...
  [150.30876  146.30876  137.30876 ]
  [148.24005  144.24005  135.24005 ]
  [148.57167  144.46309  135.78883 ]]

 [[156.74185  151.74185  146.00935 ]
  [153.76668  148.7667   143.65936 ]
  [152.70569  147.80002  142.84108 ]
  ...
  [151.09009  146.66547  138.09007 ]
  [149.3255   145.01122  136.3255  ]
  [148.62302  144.08458  136.47935 ]]

 ...

 [[ 88.926575  86.74257   78.93834 ]
  [ 90.64336   87.64336   80.65359 ]
  [ 90.34701   87.34701   80.471695]
  ...
  [ 97.20424   93.518616  85.85841 ]
  [ 95.739876  92.10975   85.19647 ]
  [ 96.309105  93.12032   85.91601 ]]

 [[ 89.93866   87.84665   79.27446 ]
  [ 91.0786    88.0786    80.1989  ]
  [ 91.141426  88.141426  80.79738 ]
  ...
  [ 97.09833   92.69194   84.98099 ]
  [ 97.15651   92.15651   86.357925]
  [ 97.78643   92.78643   87.03206 ]]

 [[ 90.12019   88.36869   81.31419 ]
  [ 91.03465   88.98422   80.27197 ]
  [ 92.06368   89.09242   80.28277 ]
  ...
  [ 95.86966   91.76649   85.816345]
  [ 97.12004   92.27439   87.72017 ]
```

```
   [ 98.73473   93.71773   89.909035]]]


  [[[255.       255.       255.      ]
    [255.       255.       255.      ]
    [254.9116   254.5138   254.5138  ]
    ...
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]]

   [[255.       255.       255.      ]
    [255.       255.       255.      ]
    [239.69434  164.23387  164.23387 ]
    ...
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]]

   [[251.50546  234.27658  234.27658 ]
    [227.69655   93.08418   93.08418 ]
    [212.         0.         0.      ]
    ...
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]]

   ...

   [[212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    ...
    [212.         0.         0.      ]
    [227.6972    93.08807   93.08807 ]
    [251.57646  234.73807  234.73807 ]]

   [[212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    ...
    [239.69516  164.23874  164.23874 ]
    [255.       255.       255.      ]
    [255.       255.       255.      ]]

   [[212.         0.         0.      ]
    [212.         0.         0.      ]
    [212.         0.         0.      ]
    ...
    [255.       255.       255.      ]
    [255.       255.       255.      ]
    [255.       255.       255.      ]]]], shape=(128, 180, 180, 3), dtype=float32)
tf.Tensor(
[[[[0.65071064 0.12070312 0.17620091]
   [0.6515193  0.2273211  0.27783108]
   [0.66463745 0.387459   0.44065773]
   ...
   [0.571347   0.60271955 0.6105627 ]
   [0.577698   0.60907054 0.6169137 ]
   [0.58307713 0.6144497  0.6222928 ]]

  [[0.63415515 0.09690022 0.1596453 ]
   [0.62903285 0.09324323 0.15471089]
   [0.6338915  0.09719141 0.16078278]
```

```
  ...
  [0.57367575 0.6059209  0.61376405]
  [0.57358    0.60611534 0.6139585 ]
  [0.5737922  0.60516477 0.6130079 ]]

 [[0.62676764 0.08954591 0.15222457]
  [0.63744915 0.10019416 0.16293927]
  [0.62752014 0.09026521 0.15301032]
  ...
  [0.5790495  0.6143528  0.62219596]
  [0.5827723  0.61800134 0.62595004]
  [0.5756192  0.60699177 0.6148349 ]]

 ...

 [[0.8529381  0.6402868  0.6616462 ]
  [0.78829277 0.646778   0.6571001 ]
  [0.74792916 0.6916796  0.6866247 ]
  ...
  [0.8472995  0.8677119  0.89988416]
  [0.8160753  0.54576236 0.5995679 ]
  [0.7908099  0.21044184 0.30413845]]

 [[0.6239166  0.59664273 0.60201025]
  [0.6085212  0.6691401  0.6655977 ]
  [0.71962315 0.7909735  0.79256916]
  ...
  [0.8475904  0.9007464  0.93080276]
  [0.84806204 0.75915575 0.77761954]
  [0.8098285  0.30957478 0.38632363]]

 [[0.5518211  0.6509517  0.65323365]
  [0.76615393 0.849418   0.8740829 ]
  [0.8617914  0.9255645  0.95160025]
  ...
  [0.84532493 0.89902574 0.93378454]
  [0.8472083  0.863396   0.89482147]
  [0.8216022  0.45358065 0.5487594 ]]]


[[[0.5587619  0.5624974  0.521414  ]
  [0.5565828  0.5574629  0.52135074]
  [0.55438566 0.55438566 0.5192817 ]
  ...
  [0.5424453  0.5306806  0.49538648]
  [0.5370576  0.5285818  0.49195263]
  [0.5253633  0.5265386  0.48283345]]

 [[0.563049   0.56342727 0.5240225 ]
  [0.5522237  0.5563585  0.5158306 ]
  [0.54765195 0.5537643  0.51216066]
  ...
  [0.5384713  0.5267066  0.49141246]
  [0.53672814 0.52750784 0.4911715 ]
  [0.5212267  0.5248259  0.4793915 ]]

 [[0.5653288  0.5653288  0.5261131 ]
  [0.55600905 0.56198245 0.51978004]
  [0.54878265 0.55652857 0.513464  ]
  ...
  [0.53434235 0.5226263  0.4873146 ]
  [0.53381073 0.5234528  0.48762542]
  [0.5216225  0.522272   0.48204625]]
```

```
...

[[0.3020266  0.29615366 0.266837  ]
 [0.29737318 0.29901612 0.26782033]
 [0.3038026  0.30747962 0.27611107]
 ...
 [0.32249328 0.31726092 0.27552918]
 [0.31930125 0.31629965 0.29580605]
 [0.32731554 0.32447127 0.29301423]]

[[0.302909   0.29484206 0.26615846]
 [0.2991516  0.29869547 0.26752475]
 [0.30832684 0.3109744  0.2789431 ]
 ...
 [0.32098103 0.3160167  0.27773646]
 [0.3208042  0.3183592  0.29168448]
 [0.32675907 0.32244226 0.2884476 ]]

[[0.30355752 0.30186108 0.2712683 ]
 [0.3080878  0.30776155 0.27649775]
 [0.31414774 0.31414774 0.28119168]
 ...
 [0.31462082 0.3105232  0.274496  ]
 [0.31922582 0.31780744 0.2829515 ]
 [0.32510602 0.31916407 0.27978107]]]


[[[0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16437474 0.49830446 0.9960911 ]
  [0.16259873 0.5003223  0.99643964]]

 [[0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.1623731  0.4998613  0.99730676]]

 [[0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.1566949  0.49571913 0.99672335]]

 ...

 [[0.15863723 0.4446287  0.85113966]
  [0.16454154 0.5042639  0.98913574]
  [0.15825558 0.5008405  0.99849916]
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]]

 [[0.15424228 0.48719585 0.96779364]
  [0.15760535 0.49081168 0.97646147]
  [0.15626834 0.501242   0.9962779 ]
```

```
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]]

 [[0.15859699 0.4933584  0.98409086]
  [0.15772    0.4972101  0.9890964 ]
  [0.15694806 0.50118196 0.9950131 ]
  ...
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]
  [0.16470589 0.49803922 0.99607843]]]


...


[[[0.7833174  0.7793958  0.759788  ]
  [0.78431374 0.78039217 0.7607843 ]
  [0.78298026 0.77905875 0.75945085]
  ...
  [0.82944995 0.8287297  0.8075212 ]
  [0.82855606 0.82623166 0.80582523]
  [0.8290245  0.8297244  0.8078058 ]]

 [[0.7865729  0.7798762  0.7602683 ]
  [0.78431374 0.78039217 0.7607843 ]
  [0.78284746 0.778926   0.7593181 ]
  ...
  [0.82634485 0.8302665  0.80673707]
  [0.8246662  0.8263842  0.8039565 ]
  [0.82863826 0.8273319  0.80641645]]

 [[0.7896973  0.78039217 0.76026803]
  [0.7772814  0.7803474  0.75724584]
  [0.77867633 0.77850413 0.7569687 ]
  ...
  [0.8235294  0.827451   0.8039216 ]
  [0.82457966 0.8285012  0.8049718 ]
  [0.8245338  0.8284554  0.804926  ]]

 ...

 [[0.6929982  0.68123347 0.6616256 ]
  [0.69042695 0.67866224 0.6590544 ]
  [0.68924093 0.6774762  0.6578684 ]
  ...
  [0.6798822  0.636745   0.62890184]
  [0.68852687 0.64500666 0.63792944]
  [0.68965536 0.64306265 0.6319961 ]]

 [[0.6946175  0.68285275 0.66324496]
  [0.69526285 0.68349814 0.6638903 ]
  [0.6903674  0.6786027  0.65899485]
  ...
  [0.68184614 0.63730395 0.63227075]
  [0.6814305  0.6343717  0.6343717 ]
  [0.6865526  0.6410528  0.63859075]]

 [[0.69847035 0.68670565 0.6670978 ]
  [0.6977548  0.6859901  0.66638225]
  [0.6882247  0.67645997 0.6568521 ]
  ...
  [0.69033873 0.6462681  0.64326733]
```

```
  [0.6948491  0.64779025 0.64779025]
  [0.6969229  0.649864   0.649864  ]]]


[[[0.8352941  0.8235294  0.79607844]
  [0.83571994 0.82395524 0.79650426]
  [0.8363412  0.82457656 0.7971256 ]
  ...
  [0.8106596  0.7988949  0.7714439 ]
  [0.811802   0.8000373  0.7725864 ]
  [0.81271434 0.80135393 0.7737682 ]]

 [[0.83897066 0.82720596 0.7997549 ]
  [0.83851576 0.8277221  0.7999474 ]
  [0.838345   0.8263265  0.79900235]
  ...
  [0.8115184  0.7997451  0.77229834]
  [0.81224453 0.79948854 0.7725333 ]
  [0.8143448  0.80232614 0.7750022 ]]

 [[0.8355887  0.823824   0.79637307]
  [0.83241373 0.83057904 0.7999269 ]
  [0.8361788  0.82279694 0.7962011 ]
  ...
  [0.8081244  0.7991151  0.7707457 ]
  [0.8143005  0.79668796 0.7721609 ]
  [0.81281185 0.7989528  0.77254903]]

 ...

 [[0.7453971  0.72578925 0.70225984]
  [0.7456348  0.72602695 0.70249754]
  [0.74594945 0.7264643  0.70232147]
  ...
  [0.71277744 0.6931993  0.6695809 ]
  [0.71338844 0.6941073  0.66959774]
  [0.7135005  0.69389266 0.67036325]]

 [[0.74539685 0.7260878  0.70166194]
  [0.748698   0.7324635  0.6981667 ]
  [0.75174004 0.7359867  0.6931968 ]
  ...
  [0.7166494  0.6970416  0.67351216]
  [0.7172708  0.69766295 0.67413354]
  [0.71232504 0.6927172  0.6691878 ]]

 [[0.7452328  0.725625   0.70209557]
  [0.74565846 0.7266111  0.70140034]
  [0.74979097 0.7301831  0.7066537 ]
  ...
  [0.71970046 0.7000926  0.6765632 ]
  [0.7211428  0.701535   0.6780056 ]
  [0.7185549  0.6989471  0.67541766]]]


[[[0.81966686 0.800059   0.78437275]
  [0.81960785 0.8        0.78431374]
  [0.8174441  0.79783636 0.78215   ]
  ...
  [0.8141508  0.79454297 0.7788567 ]
  [0.8154419  0.795834   0.78014773]
  [0.8117647  0.7921569  0.7764706 ]]

 [[0.81960785 0.8        0.7790058 ]
```

```
 [0.8193446  0.79973674 0.78431374]
 [0.8182938  0.79868597 0.78419197]
 ...
 [0.81750154 0.7978937  0.7822074 ]
 [0.8181003  0.7984924  0.7828061 ]
 [0.8152574  0.7956496  0.7799633 ]]

[[0.8226103  0.8030025  0.7824071 ]
 [0.82108325 0.80147535 0.78715235]
 [0.81771815 0.7981103  0.785086  ]
 ...
 [0.8185141  0.79890627 0.78322   ]
 [0.81960785 0.8        0.78431374]
 [0.81907517 0.7994673  0.78378105]]

...

[[0.7325874  0.71297956 0.68945014]
 [0.7328297  0.71322185 0.6896924 ]
 [0.7344674  0.7148596  0.6913302 ]
 ...
 [0.7176471  0.69803923 0.68235296]
 [0.7176471  0.69803923 0.68235296]
 [0.7176471  0.69803923 0.68235296]]

[[0.7473364  0.72772855 0.70419914]
 [0.7356873  0.7160795  0.69255006]
 [0.73380256 0.7141947  0.6906653 ]
 ...
 [0.7178887  0.6982809  0.6825946 ]
 [0.7176471  0.69803923 0.6793378 ]
 [0.71813476 0.6985269  0.6812099 ]]

[[0.75274044 0.7331326  0.7096032 ]
 [0.7402553  0.72064745 0.69711804]
 [0.73700225 0.7173944  0.693865  ]
 ...
 [0.7176471  0.69803923 0.68196774]
 [0.7178242  0.6982164  0.67790246]
 [0.71970797 0.7001001  0.6785262 ]]], shape=(128, 180, 180, 3), dtype=float32)
```

In [20]:
```
model = Sequential()

model.add(Conv2D(filters = 64, kernel_size=(3, 3), activation="relu", input_shape=(
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=128, kernel_size=(4, 4), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(units = 34, activation="relu"))
model.add(Dense(units=4, activation="softmax"))
```

In [21]:
```
epochs = 20

callbacks = [
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.keras"),
]

model.compile(
    optimizer=keras.optimizers.Adam(3e-4),
    loss=keras.losses.SparseCategoricalCrossentropy(),
    metrics=["accuracy"]
)
```

```
model.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
    validation_data=val_ds,
)
```

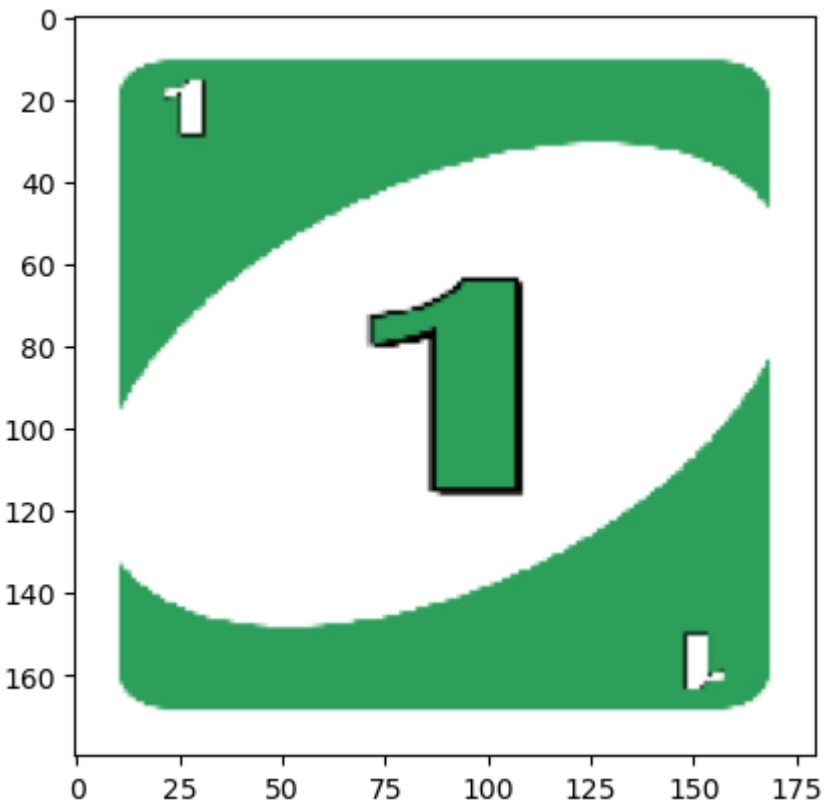
```
Epoch 1/20
WARNING:tensorflow:From C:\Users\selas\anaconda3\Lib\site-packages\keras\src\utils
\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use t
f.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\selas\anaconda3\Lib\site-packages\keras\src\engin
e\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is depr
ecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

2/2 [==============================] - 6s 1s/step - loss: 1.4424 - accuracy: 0.287
7 - val_loss: 4.5477 - val_accuracy: 0.1667
Epoch 2/20
2/2 [==============================] - 5s 1s/step - loss: 3.5802 - accuracy: 0.274
0 - val_loss: 1.5360 - val_accuracy: 0.6111
Epoch 3/20
2/2 [==============================] - 5s 1s/step - loss: 1.5957 - accuracy: 0.472
6 - val_loss: 1.0253 - val_accuracy: 0.4722
Epoch 4/20
2/2 [==============================] - 5s 1s/step - loss: 1.1189 - accuracy: 0.369
9 - val_loss: 0.9268 - val_accuracy: 0.6389
Epoch 5/20
2/2 [==============================] - 6s 1s/step - loss: 0.8548 - accuracy: 0.657
5 - val_loss: 0.9389 - val_accuracy: 0.5833
Epoch 6/20
2/2 [==============================] - 6s 1s/step - loss: 0.7956 - accuracy: 0.630
1 - val_loss: 0.7167 - val_accuracy: 0.7500
Epoch 7/20
2/2 [==============================] - 6s 1s/step - loss: 0.6179 - accuracy: 0.856
2 - val_loss: 0.5476 - val_accuracy: 0.7222
Epoch 8/20
2/2 [==============================] - 6s 1s/step - loss: 0.4819 - accuracy: 0.863
0 - val_loss: 0.4058 - val_accuracy: 1.0000
Epoch 9/20
2/2 [==============================] - 6s 1s/step - loss: 0.3928 - accuracy: 1.000
0 - val_loss: 0.3794 - val_accuracy: 0.7778
Epoch 10/20
2/2 [==============================] - 5s 1s/step - loss: 0.3037 - accuracy: 0.883
6 - val_loss: 0.3446 - val_accuracy: 0.7778
Epoch 11/20
2/2 [==============================] - 6s 1s/step - loss: 0.2437 - accuracy: 0.883
6 - val_loss: 0.2128 - val_accuracy: 0.9722
Epoch 12/20
2/2 [==============================] - 6s 1s/step - loss: 0.1787 - accuracy: 0.979
5 - val_loss: 0.1476 - val_accuracy: 1.0000
Epoch 13/20
2/2 [==============================] - 6s 1s/step - loss: 0.1378 - accuracy: 1.000
0 - val_loss: 0.2908 - val_accuracy: 0.8056
Epoch 14/20
2/2 [==============================] - 6s 1s/step - loss: 0.1758 - accuracy: 0.876
7 - val_loss: 0.0967 - val_accuracy: 0.9722
Epoch 15/20
2/2 [==============================] - 6s 1s/step - loss: 0.1333 - accuracy: 0.958
9 - val_loss: 0.0705 - val_accuracy: 1.0000
Epoch 16/20
2/2 [==============================] - 6s 1s/step - loss: 0.0767 - accuracy: 1.000
0 - val_loss: 0.2275 - val_accuracy: 0.8611
Epoch 17/20
2/2 [==============================] - 6s 1s/step - loss: 0.1444 - accuracy: 0.911
0 - val_loss: 0.1184 - val_accuracy: 0.9722
Epoch 18/20
2/2 [==============================] - 6s 1s/step - loss: 0.0734 - accuracy: 0.958
9 - val_loss: 0.0408 - val_accuracy: 1.0000
Epoch 19/20
2/2 [==============================] - 6s 1s/step - loss: 0.0577 - accuracy: 0.986
```

```
3 - val_loss: 0.0509 - val_accuracy: 1.0000
Epoch 20/20
2/2 [==============================] - 7s 2s/step - loss: 0.0647 - accuracy: 0.986
3 - val_loss: 0.0225 - val_accuracy: 1.0000
```

Out[21]:
```
<keras.src.callbacks.History at 0x209447c5410>
```

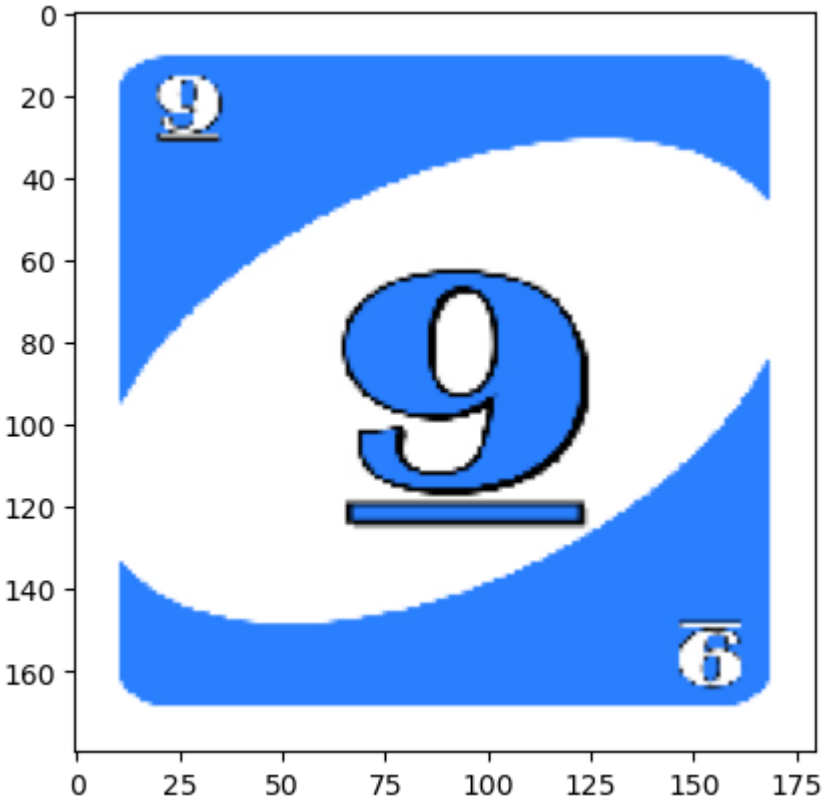
In [22]:
```
img = keras.utils.load_img("University/Dataset/green/Green_1.jpg", target_size=imag
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
predicted_color_index = tf.argmax(predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]

print(f"The predicted color is: {predicted_color}")
```

```
1/1 [==============================] - 0s 53ms/step
2
The predicted color is: Green
```



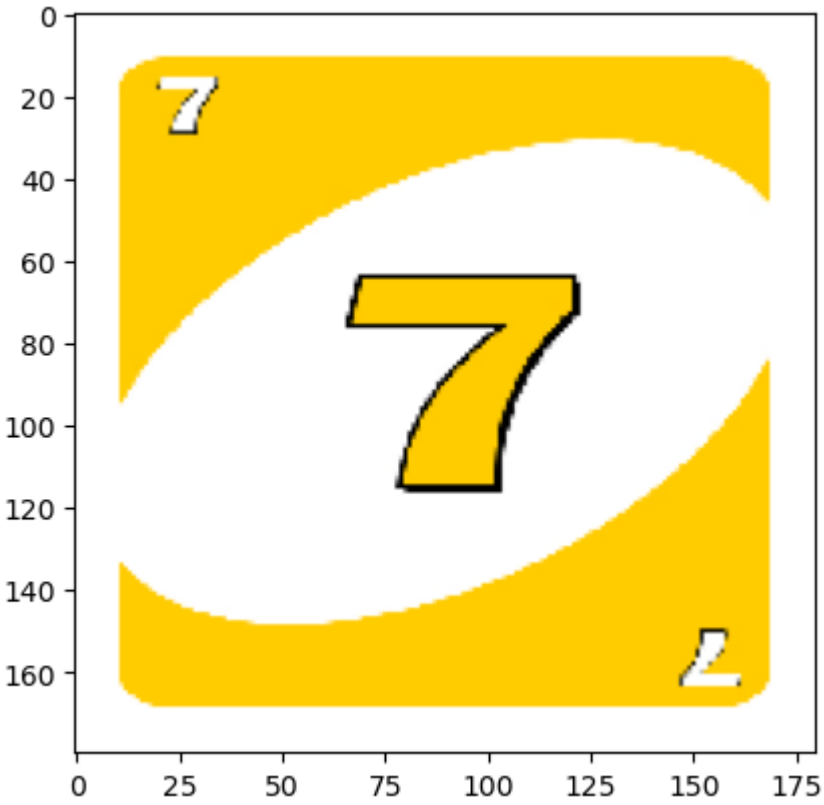
In [23]:
```
img = keras.utils.load_img("University/Dataset/Blue/Blue_9.jpg", target_size=image_
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
predicted_color_index = tf.argmax(predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]

print(f"The predicted color is: {predicted_color}")
```

```
1/1 [==============================] - 0s 26ms/step
0
The predicted color is: Blue
```



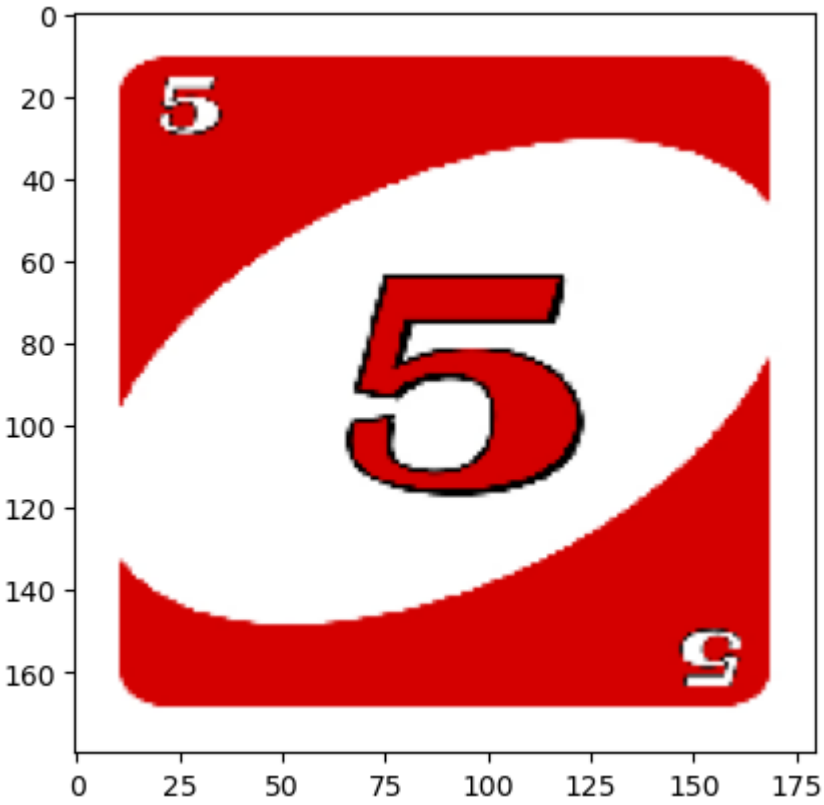
In [24]:
```
img = keras.utils.load_img("University/Dataset/yellow/Yellow_7.jpg", target_size=im
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
predicted_color_index = tf.argmax(predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]

print(f"The predicted color is: {predicted_color}")
```

```
1/1 [==============================] - 0s 27ms/step
3
The predicted color is: Yellow
```



In [25]:
```
img = keras.utils.load_img("University/Dataset/Red/Red_5.jpg", target_size=image_si
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)  # Create batch axis

predictions = model.predict(img_array)
predicted_color_index = tf.argmax(predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]

print(f"The predicted color is: {predicted_color}")
```

```
1/1 [==============================] - 0s 25ms/step
1
The predicted color is: Red
```



In [26]:
```
model = model.save('University/modelofcolors.keras')
```

Code for detecting the number of a given card.

In [21]:
```
image_size = (180, 180)
batch_size = 128

train_ds, val_ds = keras.utils.image_dataset_from_directory(
    "University/Numbers_Dataset",
    validation_split=0.2,
    subset="both",
    seed=13,
    image_size=image_size,
    batch_size=batch_size,
)
```

```
Found 10388 files belonging to 13 classes.
Using 8311 files for training.
Using 2077 files for validation.
```

In [22]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(images[i]).astype("uint8"))
        plt.title(int(labels[i]))
        plt.axis("off")
```

In [23]:
```
data_augmentation_layers = [
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
]


def data_augmentation(images):
    for layer in data_augmentation_layers:
        images = layer(images)
    return images
```

In [24]:
```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(np.array(augmented_images[0]).astype("uint8"))
        plt.axis("off")
```

In [25]:
```
# Apply `data_augmentation` to the training images.
train_ds = train_ds.map(
    lambda img, label: (data_augmentation(img), label),
    num_parallel_calls=tf_data.AUTOTUNE,
)
# Prefetching samples in GPU memory helps maximize GPU utilization.
train_ds = train_ds.prefetch(tf_data.AUTOTUNE)
val_ds = val_ds.prefetch(tf_data.AUTOTUNE)
```

In [26]:
```
# before normalization
for image, label in train_ds.take(1):  # Take one batch from the dataset
    print(image)

train_ds = train_ds.map(lambda x, y: (x / 255, y))
val_ds = val_ds.map(lambda x, y: (x / 255, y))

# after normalization
for image, label in train_ds.take(1):  # Take one batch from the dataset
    print(image)
```

```
tf.Tensor(
[[[174.        169.        165.       ]
  [173.87274   168.87274   164.87274  ]
  [173.20547   168.20547   164.20547  ]
  ...
  [179.0496    170.0496    163.86003  ]
  [178.94992   169.94992   161.8346   ]
  [178.92119   169.9212    160.9212   ]]

 [[174.        169.        165.       ]
  [173.67131   168.67131   164.67131  ]
  [173.36005   168.36005   164.36005  ]
  ...
  [179.58344   170.58344   162.4278   ]
  [178.65298   169.65298   160.86087  ]
  [178.91772   169.91772   160.91772  ]]

 [[174.        169.        165.       ]
  [174.        169.        165.       ]
  [173.95018   168.95018   164.9502   ]
  ...
  [179.60577   170.60577   161.60577  ]
  [179.06241   170.06241   161.06241  ]
  [179.85492   170.85492   161.8549   ]]

 ...

 [[150.71527   146.71527   147.71527  ]
  [150.92978   146.92978   147.92978  ]
  [150.61841   146.61841   147.61841  ]
  ...
  [132.19476   126.19476   126.46402  ]
  [133.32825   127.328255  127.99406  ]
  [133.70845   127.70845   129.70845  ]]

 [[150.98172   147.34744   146.96767  ]
  [150.77344   148.1246    149.05185  ]
  [150.42137   147.53519   148.19861  ]
  ...
  [133.27594   127.27593   129.27594  ]
  [131.57356   125.57356   126.43884  ]
  [132.02637   126.02637   126.91104  ]]

 [[153.3027    149.46031   150.28918  ]
  [152.10284   149.39049   150.25603  ]
  [149.90576   148.65103   148.14224  ]
  ...
  [133.28937   127.28937   129.28937  ]
  [133.09955   127.09954   127.35406  ]
  [132.38263   126.38262   126.38262  ]]]


[[[ 58.629623  146.00598    90.16449  ]
  [ 43.05101   131.80624    74.101265 ]
  [ 40.695324  129.13179    71.13179  ]
  ...
  [124.148026  104.41853    89.65396  ]
  [124.92531   104.30563    88.98256  ]
  [122.11557    98.56219    82.65894  ]]

 [[ 50.419075  139.54137    82.3841   ]
  [ 47.056755  136.19548    77.77931  ]
  [ 52.703297  141.49455    83.49454  ]
  ...
```

```
  [126.77876   106.24392    92.18065  ]
  [122.11619   101.21312    84.36934  ]
  [125.20097   100.94757    83.01668  ]]

 [[ 44.075516  133.10031    75.21105  ]
  [ 44.757225  133.75723    75.75723  ]
  [ 47.32634   136.32635    78.32634  ]
  ...
  [130.4357    111.57292    96.92892  ]
  [122.5764    100.835075   83.14235  ]
  [122.47963    94.774734   72.92561  ]]

 ...

 [[ 42.13327    88.0679     87.07237  ]
  [ 57.36193   142.6997    105.84882  ]
  [ 54.961376  150.2245    105.823586 ]
  ...
  [158.03769   138.64816   123.37422  ]
  [160.0292    135.69518   120.41568  ]
  [160.0938    138.47885   124.43642  ]]

 [[ 48.18345    98.729095   90.79979  ]
  [ 65.17806   149.31926   112.07996  ]
  [ 70.74298   164.71936   120.627884 ]
  ...
  [158.00937   138.78937   124.78937  ]
  [155.63101   129.77165   113.62981  ]
  [157.2443    133.43726   117.366196 ]]

 [[ 36.317596   80.0798     77.51187  ]
  [ 54.12307   144.93039   102.84984  ]
  [ 51.90015   148.2082    103.55784  ]
  ...
  [157.25287   137.75282   123.605774 ]
  [157.1976    132.67654   117.37216  ]
  [154.75003   128.3012    112.53356  ]]]


[[[255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  ...
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]]

 [[255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  ...
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]]

 [[255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  ...
  [255.        255.        255.       ]
  [255.        255.        255.       ]
  [255.        255.        255.       ]]

 ...
```

```
       [[255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]

       [[255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]

       [[255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]]


      ...


      [[[255.        204.          0.       ]
        [255.        204.          0.       ]
        [255.        204.          0.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]

       [[255.        204.          0.       ]
        [255.        204.          0.       ]
        [255.        204.          0.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]

       [[255.        204.          0.       ]
        [255.        204.          0.       ]
        [255.        204.          0.       ]
        ...
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]]

       ...

       [[255.        255.        255.       ]
        [255.        255.        255.       ]
        [255.        255.        255.       ]
        ...
        [255.        204.          0.       ]
        [255.        204.          0.       ]
        [255.        204.          0.       ]]

       [[255.        255.        255.       ]
```

```
 [255.        255.        255.       ]
 [255.        255.        255.       ]
 ...
 [255.        204.          0.       ]
 [255.        204.          0.       ]
 [255.        204.          0.       ]]

[[255.        255.        255.       ]
 [255.        255.        255.       ]
 [255.        255.        255.       ]
 ...
 [255.        204.          0.       ]
 [255.        204.          0.       ]
 [255.        204.          0.       ]]]


[[[182.53223   166.53223   150.53223  ]
  [182.62357   166.62358   150.62357  ]
  [181.85529   165.85529   149.85529  ]
  ...
  [130.90579   114.90577    98.92733  ]
  [127.78542   112.0682     95.18716  ]
  [121.79977   106.61259    87.89336  ]]

 [[181.30403   165.79745   148.31715  ]
  [181.12909   165.13539   149.72795  ]
  [182.20679   166.20679   150.21634  ]
  ...
  [128.42337   112.42337    96.84857  ]
  [131.18596   115.1914    100.16074  ]
  [128.20346   112.30047    96.912415 ]]

 [[187.67755   172.026     155.45407  ]
  [181.87753   167.60194   151.95882  ]
  [180.27405   164.30428   148.21358  ]
  ...
  [129.69875   113.78945    98.75922  ]
  [127.26813   111.26813    95.81406  ]
  [129.28557   113.28557    97.79879  ]]

 ...

 [[216.39734   209.46661   201.67786  ]
  [220.78973   216.02235   208.52965  ]
  [216.14676   211.7185    205.05798  ]
  ...
  [199.53772   187.53772   172.6063   ]
  [198.30249   186.30249   171.53644  ]
  [196.98676   184.98676   170.38608  ]]

 [[211.0069    202.39494   197.29793  ]
  [218.0117    210.27423   203.42542  ]
  [217.04492   210.22552   202.33604  ]
  ...
  [197.98628   185.67442   169.13828  ]
  [193.18799   181.68863   164.93971  ]
  [197.04178   185.04178   169.0636   ]]

 [[217.86946   212.68227   206.68227  ]
  [208.01503   201.60242   195.5871   ]
  [214.05698   207.05698   200.91345  ]
  ...
  [192.99017   180.98419   163.37595  ]
  [194.06671   181.49307   163.46838  ]
```

```
   [195.80602   181.83391   165.70012  ]]]


 [[[206.63678   201.63678   195.63678  ]
   [206.40948   201.40948   195.40947  ]
   [207.56126   202.56126   196.56126  ]
   ...
   [211.50554   206.50554   202.50554  ]
   [210.75436   205.75436   201.75436  ]
   [211.26724   206.26724   202.26724  ]]

  [[208.61084   203.61084   197.61084  ]
   [207.96106   202.96104   196.96106  ]
   [208.69795   203.69795   197.69795  ]
   ...
   [208.97235   203.97235   199.97235  ]
   [207.24265   202.24266   198.24266  ]
   [211.38152   206.38152   202.38152  ]]

  [[210.55435   205.55435   199.55435  ]
   [208.69058   203.69058   197.69058  ]
   [208.22064   203.22064   197.22063  ]
   ...
   [213.5828    208.5828    204.5828   ]
   [210.10405   205.10405   201.10405  ]
   [212.04993   207.04993   203.04993  ]]

  ...

  [[169.09975   126.44042     5.408354 ]
   [169.62581   128.13754     6.0181665]
   [171.40727   129.65843    10.484199 ]
   ...
   [116.72588   106.867874  105.506836 ]
   [118.18425   108.18425   107.18425  ]
   [120.70364   110.70364   109.70364  ]]

  [[172.33554   130.33554    11.475949 ]
   [172.97354   130.97353    12.973527 ]
   [170.51785   128.51788    10.473329 ]
   ...
   [117.51622   107.51622   106.51622  ]
   [119.787094  109.787094  108.787094 ]
   [120.492355  110.492355  109.492355 ]]

  [[170.53796   128.53796    10.537962 ]
   [171.9328    129.93282    11.932814 ]
   [171.33621   129.33621    11.336205 ]
   ...
   [121.09727   111.09727   110.09727  ]
   [120.8026    110.80259   109.80259  ]
   [117.97604   107.97604   106.97604  ]]]], shape=(128, 180, 180, 3), dtype=float
32)
tf.Tensor(
[[[[0.8057798  0.81962454 0.8156703 ]
   [0.80777204 0.81748587 0.81356424]
   [0.80516785 0.8170977  0.8131761 ]
   ...
   [0.7231916  0.7271132  0.7349563 ]
   [0.72434974 0.7282713  0.73611444]
   [0.7294546  0.73337615 0.7412193 ]]

  [[0.80933917 0.8248186  0.8208952 ]
   [0.81330705 0.8265597  0.82263815]
```

```
 [0.80887055 0.82001483 0.81609327]
 ...
 [0.7274155  0.7313371  0.7391802 ]
 [0.7315123  0.7354339  0.743277  ]
 [0.7308291  0.7347507  0.7425938 ]]

[[0.8079091  0.81638944 0.8124679 ]
 [0.8050648  0.81972516 0.81453496]
 [0.8076485  0.8235965  0.81842756]
 ...
 [0.73035717 0.7342787  0.7421219 ]
 [0.7335968  0.73751837 0.7453615 ]
 [0.7379835  0.7419051  0.7497483 ]]

...

[[0.7826052  0.7904483  0.78652674]
 [0.77571815 0.7835613  0.7796397 ]
 [0.7762963  0.78413945 0.7802179 ]
 ...
 [0.6714467  0.67310584 0.680949  ]
 [0.6855008  0.6855008  0.69334394]
 [0.6766014  0.6766014  0.68444455]]

[[0.78216016 0.7900033  0.78608173]
 [0.7812988  0.78914195 0.7852204 ]
 [0.77491695 0.7827601  0.7788385 ]
 ...
 [0.6782217  0.679161   0.6870042 ]
 [0.67816764 0.67816764 0.6860108 ]
 [0.674837   0.674837   0.68268013]]

[[0.76574737 0.7735905  0.76966894]
 [0.7764935  0.7843366  0.7804151 ]
 [0.773939   0.78178215 0.7778606 ]
 ...
 [0.67876667 0.678986   0.68682915]
 [0.6769288  0.6769288  0.68477196]
 [0.6705244  0.6705244  0.67836756]]]


[[[0.8942225  0.86479986 0.8344132 ]
  [0.89128226 0.85767156 0.82937884]
  [0.87733394 0.84878385 0.8179609 ]
  ...
  [0.9313131  0.90386206 0.87126404]
  [0.9318463  0.90439534 0.8730228 ]
  [0.92870206 0.9012511  0.86987853]]

 [[0.8800584  0.84744436 0.8186533 ]
  [0.8792468  0.84766835 0.81835955]
  [0.87705874 0.846106   0.81648433]
  ...
  [0.926435   0.89898396 0.8676114 ]
  [0.932433   0.90498203 0.8736095 ]
  [0.9270643  0.8996133  0.8682408 ]]

 [[0.8798505  0.8445564  0.8171054 ]
  [0.8767009  0.84150785 0.8140064 ]
  [0.87443453 0.83977896 0.81200874]
  ...
  [0.9226177  0.8951667  0.86379415]
  [0.9263569  0.898906   0.86753345]
  [0.9303922  0.9029412  0.8715687 ]]
```

```
 ...

 [[0.5690096  0.5180292  0.49449974]
  [0.5759998  0.5250194  0.50149   ]
  [0.57405835 0.52307796 0.49954858]
  ...
  [0.8399931  0.806702   0.76740193]
  [0.8411764  0.80685574 0.7696147 ]
  [0.83751225 0.80582756 0.76331466]]

 [[0.5676009  0.5108078  0.49018472]
  [0.58107114 0.53001136 0.50652164]
  [0.5721211  0.51922727 0.49665457]
  ...
  [0.8407711  0.80547696 0.77018285]
  [0.83959633 0.80632883 0.76698136]
  [0.83380985 0.8024373  0.7593    ]]

 [[0.56562394 0.51133054 0.4944124 ]
  [0.5668471  0.51414907 0.49283466]
  [0.5774427  0.51924235 0.5028481 ]
  ...
  [0.8358761  0.800582   0.7652879 ]
  [0.82755643 0.79243577 0.75679475]
  [0.8322827  0.8002447  0.75843847]]]


[[[0.8058246  0.7862167  0.7705304 ]
  [0.7943284  0.77472055 0.7590343 ]
  [0.79746777 0.77785987 0.76217365]
  ...
  [0.83153814 0.8119303  0.796244  ]
  [0.8279727  0.80836487 0.7926786 ]
  [0.8325642  0.81295633 0.79727006]]

 [[0.8036257  0.7840179  0.76833165]
  [0.797991   0.77838314 0.76269686]
  [0.78984314 0.7702353  0.754549  ]
  ...
  [0.8445077  0.82489985 0.8092136 ]
  [0.8285787  0.80897087 0.7932846 ]
  [0.8269815  0.8073736  0.7916873 ]]

 [[0.8055126  0.78590477 0.7702185 ]
  [0.797644   0.7780362  0.7623499 ]
  [0.7915668  0.7719589  0.7562726 ]
  ...
  [0.8414242  0.8218164  0.8061301 ]
  [0.83003706 0.8104292  0.79474294]
  [0.8262956  0.8066878  0.7910015 ]]

 ...

 [[0.6940028  0.67749506 0.6781311 ]
  [0.7062317  0.68491304 0.69165087]
  [0.71443576 0.69090635 0.6987495 ]
  ...
  [0.45244494 0.30806857 0.02752236]
  [0.41974714 0.31518653 0.0939209 ]
  [0.43417808 0.39943543 0.3466767 ]]

 [[0.6961369  0.67758965 0.6728285 ]
  [0.71316445 0.6920629  0.698692  ]
```

```
  [0.7237444  0.7017541  0.70690405]
  ...
  [0.44109082 0.2972135  0.01657913]
  [0.43299392 0.30556828 0.05132028]
  [0.43793    0.36134657 0.22287187]]

 [[0.69542813 0.67373526 0.66611296]
  [0.7086285  0.684993   0.6925109 ]
  [0.7165115  0.6929821  0.7008252 ]
  ...
  [0.43363982 0.29045954 0.00846103]
  [0.44369045 0.30446544 0.02658972]
  [0.42970476 0.31542256 0.08628495]]]


...


[[[0.7931616  0.7774753  0.7304165 ]
  [0.79441035 0.7787241  0.73166525]
  [0.79441315 0.7787269  0.73166806]
  ...
  [0.71971625 0.6931979  0.6304528 ]
  [0.71661335 0.6897247  0.6269796 ]
  [0.71115494 0.6871606  0.6244155 ]]

 [[0.7892505  0.7735642  0.7265054 ]
  [0.7936501  0.7781451  0.7305424 ]
  [0.7918152  0.77869654 0.72393507]
  ...
  [0.71125644 0.68734205 0.624597  ]
  [0.71522    0.69169056 0.62894547]
  [0.7141249  0.6905955  0.6278504 ]]

 [[0.79028034 0.77459407 0.72753525]
  [0.7954282  0.77974194 0.7326831 ]
  [0.79455143 0.7820029  0.7255306 ]
  ...
  [0.7102854  0.686756   0.6240109 ]
  [0.71761304 0.69408363 0.63133854]
  [0.71609956 0.69257015 0.62982506]]

 ...

 [[0.8315223  0.8079929  0.76093405]
  [0.83109224 0.8079081  0.76084924]
  [0.8286535  0.8051241  0.7580653 ]
  ...
  [0.70972633 0.68227535 0.61953026]
  [0.70594424 0.67849326 0.61574817]
  [0.70641553 0.67896456 0.61621946]]

 [[0.83011985 0.80659044 0.7595317 ]
  [0.8298649  0.80633545 0.7592766 ]
  [0.82893234 0.80540293 0.7583441 ]
  ...
  [0.71284527 0.6853943  0.6226492 ]
  [0.71043617 0.6829852  0.6202401 ]
  [0.7083174  0.6808664  0.6181213 ]]

 [[0.8319189  0.8083895  0.76133066]
  [0.8312798  0.8077504  0.7606916 ]
  [0.83174586 0.80821645 0.7611577 ]
  ...
```

```
  [0.71375364 0.68630266 0.6235575 ]
  [0.7114169  0.6839659  0.6212208 ]
  [0.7098901  0.68243915 0.61969405]]]


[[[0.8217726  0.7913025  0.7789492 ]
  [0.82055026 0.7891777  0.777413  ]
  [0.8193516  0.7884884  0.7765539 ]
  ...
  [0.46172768 0.5540328  0.40338793]
  [0.54021484 0.6002863  0.48446634]
  [0.736267   0.7301751  0.6975197 ]]

 [[0.8263467  0.79497415 0.78320944]
  [0.8252645  0.79389197 0.78212714]
  [0.8198434  0.79117185 0.7785069 ]
  ...
  [0.4484295  0.5469462  0.394005  ]
  [0.51242906 0.58160186 0.45583132]
  [0.7079699  0.7206163  0.6627154 ]]

 [[0.8238377  0.79246515 0.78070056]
  [0.81295663 0.7815841  0.7698194 ]
  [0.82149565 0.79012305 0.7783583 ]
  ...
  [0.43327734 0.5307057  0.38275826]
  [0.47443998 0.5522473  0.422169  ]
  [0.6320826  0.6507282  0.591638  ]]

 ...

 [[0.49608803 0.4502596  0.45748755]
  [0.5069735  0.46065128 0.4681261 ]
  [0.5022597  0.45520085 0.45760712]
  ...
  [0.19575067 0.34816778 0.12621833]
  [0.19865921 0.3524938  0.137403  ]
  [0.17533183 0.32510176 0.11844906]]

 [[0.5086766  0.4653337  0.4713189 ]
  [0.504059   0.46248165 0.46758404]
  [0.5064921  0.45943326 0.46632132]
  ...
  [0.19955207 0.34984916 0.12658277]
  [0.18673934 0.341329   0.12064435]
  [0.17718422 0.33005884 0.11737204]]

 [[0.5025358  0.45690396 0.46315768]
  [0.5099867  0.46648654 0.47255033]
  [0.5094226  0.46254146 0.47029576]
  ...
  [0.1892547  0.341613   0.12428941]
  [0.21550873 0.3629246  0.14558801]
  [0.2104807  0.35851434 0.14549093]]]


[[[0.85603666 0.8391439  0.8221001 ]
  [0.85471445 0.8351066  0.81942034]
  [0.85502756 0.83541965 0.8197334 ]
  ...
  [0.87107694 0.8543084  0.8372025 ]
  [0.8728878  0.8599924  0.8409499 ]
  [0.8700181  0.8552275  0.8371326 ]]
```

```
[[0.85701    0.83913416 0.82258195]
 [0.8586548  0.839047   0.82336074]
 [0.855075   0.8354672  0.81978095]
 ...
 [0.8694611  0.8498927  0.8341867 ]
 [0.871178   0.8516218  0.83539194]
 [0.87079734 0.85438234 0.8365154 ]]

[[0.85761684 0.83839345 0.8225149 ]
 [0.858564   0.83919203 0.82338786]
 [0.8562609  0.83665305 0.8209668 ]
 ...
 [0.86384326 0.8471773  0.8284633 ]
 [0.86180854 0.84587425 0.8241955 ]
 [0.86542886 0.8496582  0.82939583]]

...

[[0.69672096 0.41366637 0.41510466]
 [0.73802465 0.70361197 0.68560845]
 [0.7321909  0.714054   0.7022893 ]
 ...
 [0.48436573 0.31368533 0.30760324]
 [0.46262196 0.09341524 0.10395295]
 [0.46647638 0.04039152 0.05106514]]

[[0.6603724  0.2126405  0.2189878 ]
 [0.7276207  0.6889937  0.67171276]
 [0.7305078  0.71089995 0.69913524]
 ...
 [0.50374544 0.49089056 0.47194076]
 [0.4750392  0.32517466 0.32103613]
 [0.462774   0.10741606 0.11347454]]

[[0.6452222  0.17156516 0.18126993]
 [0.6921123  0.560531   0.5552136 ]
 [0.7279732  0.7077863  0.6977588 ]
 ...
 [0.50072193 0.486003   0.4678723 ]
 [0.5058634  0.478642   0.46328273]
 [0.4846623  0.31929415 0.31573272]]]], shape=(128, 180, 180, 3), dtype=float32)
```

In [27]:
```
model1 = Sequential()

model1.add(Conv2D(filters = 32, kernel_size=(3, 3), activation="relu", input_shape=
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(filters=64, kernel_size=(4, 4), activation="relu"))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(filters=128, kernel_size=(4, 4), activation="relu"))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(filters=256, kernel_size=(4, 4), activation="relu"))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Flatten())
model1.add(Dense(units = 64, activation="relu"))
model1.add(Dense(units = 32, activation="relu"))
model1.add(Dense(units=13, activation="softmax"))
```

In [28]:
```
epochs = 70

callbacks = [
```

```
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.keras"),
]

model1.compile(
    optimizer=keras.optimizers.Adam(3e-4),
    loss=keras.losses.SparseCategoricalCrossentropy(),
    metrics=["accuracy"]
)

model1.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
    validation_data=val_ds,
)
```

```
Epoch 1/70
65/65 [==============================] - 258s 4s/step - loss: 2.5454 - accuracy:
0.1008 - val_loss: 2.5166 - val_accuracy: 0.1184
Epoch 2/70
65/65 [==============================] - 279s 4s/step - loss: 2.4784 - accuracy:
0.1473 - val_loss: 2.4383 - val_accuracy: 0.1762
Epoch 3/70
65/65 [==============================] - 312s 5s/step - loss: 2.3843 - accuracy:
0.1916 - val_loss: 2.3496 - val_accuracy: 0.1733
Epoch 4/70
65/65 [==============================] - 307s 5s/step - loss: 2.1906 - accuracy:
0.2613 - val_loss: 2.3403 - val_accuracy: 0.2321
Epoch 5/70
65/65 [==============================] - 305s 4s/step - loss: 1.9357 - accuracy:
0.3574 - val_loss: 1.8903 - val_accuracy: 0.3664
Epoch 6/70
65/65 [==============================] - 306s 4s/step - loss: 1.6451 - accuracy:
0.4560 - val_loss: 2.2359 - val_accuracy: 0.3216
Epoch 7/70
65/65 [==============================] - 305s 4s/step - loss: 1.4503 - accuracy:
0.5167 - val_loss: 1.6670 - val_accuracy: 0.4444
Epoch 8/70
65/65 [==============================] - 305s 4s/step - loss: 1.2465 - accuracy:
0.5873 - val_loss: 1.2961 - val_accuracy: 0.5532
Epoch 9/70
65/65 [==============================] - 303s 4s/step - loss: 1.1466 - accuracy:
0.6142 - val_loss: 1.4621 - val_accuracy: 0.5181
Epoch 10/70
65/65 [==============================] - 305s 4s/step - loss: 0.9785 - accuracy:
0.6772 - val_loss: 1.6085 - val_accuracy: 0.5099
Epoch 11/70
65/65 [==============================] - 301s 4s/step - loss: 0.8872 - accuracy:
0.7093 - val_loss: 1.6050 - val_accuracy: 0.5373
Epoch 12/70
65/65 [==============================] - 303s 4s/step - loss: 0.8030 - accuracy:
0.7337 - val_loss: 1.3001 - val_accuracy: 0.5883
Epoch 13/70
65/65 [==============================] - 301s 4s/step - loss: 0.7058 - accuracy:
0.7663 - val_loss: 1.2050 - val_accuracy: 0.6273
Epoch 14/70
65/65 [==============================] - 300s 4s/step - loss: 0.6132 - accuracy:
0.7962 - val_loss: 1.5386 - val_accuracy: 0.5961
Epoch 15/70
65/65 [==============================] - 305s 4s/step - loss: 0.6083 - accuracy:
0.7970 - val_loss: 1.1245 - val_accuracy: 0.6610
Epoch 16/70
65/65 [==============================] - 302s 4s/step - loss: 0.5425 - accuracy:
0.8148 - val_loss: 1.0047 - val_accuracy: 0.6976
Epoch 17/70
65/65 [==============================] - 304s 4s/step - loss: 0.4929 - accuracy:
0.8320 - val_loss: 0.9517 - val_accuracy: 0.7044
Epoch 18/70
65/65 [==============================] - 304s 4s/step - loss: 0.4479 - accuracy:
0.8516 - val_loss: 0.9006 - val_accuracy: 0.7285
Epoch 19/70
65/65 [==============================] - 304s 4s/step - loss: 0.4231 - accuracy:
0.8589 - val_loss: 1.0975 - val_accuracy: 0.7005
Epoch 20/70
65/65 [==============================] - 336s 5s/step - loss: 0.3848 - accuracy:
0.8739 - val_loss: 1.1328 - val_accuracy: 0.7015
Epoch 21/70
65/65 [==============================] - 317s 5s/step - loss: 0.3740 - accuracy:
0.8738 - val_loss: 0.9026 - val_accuracy: 0.7496
Epoch 22/70
```

```
65/65 [==============================] - 295s 4s/step - loss: 0.3458 - accuracy:
0.8828 - val_loss: 0.9279 - val_accuracy: 0.7410
Epoch 23/70
65/65 [==============================] - 306s 5s/step - loss: 0.3163 - accuracy:
0.8975 - val_loss: 0.6833 - val_accuracy: 0.7954
Epoch 24/70
65/65 [==============================] - 313s 5s/step - loss: 0.3078 - accuracy:
0.9012 - val_loss: 0.8539 - val_accuracy: 0.7501
Epoch 25/70
65/65 [==============================] - 318s 5s/step - loss: 0.2884 - accuracy:
0.9054 - val_loss: 1.0925 - val_accuracy: 0.7217
Epoch 26/70
65/65 [==============================] - 320s 5s/step - loss: 0.2655 - accuracy:
0.9181 - val_loss: 0.7605 - val_accuracy: 0.7747
Epoch 27/70
65/65 [==============================] - 310s 5s/step - loss: 0.2317 - accuracy:
0.9260 - val_loss: 1.2162 - val_accuracy: 0.7058
Epoch 28/70
65/65 [==============================] - 304s 4s/step - loss: 0.2433 - accuracy:
0.9217 - val_loss: 0.8390 - val_accuracy: 0.7684
Epoch 29/70
65/65 [==============================] - 304s 4s/step - loss: 0.1977 - accuracy:
0.9357 - val_loss: 0.9319 - val_accuracy: 0.7597
Epoch 30/70
65/65 [==============================] - 304s 4s/step - loss: 0.1938 - accuracy:
0.9373 - val_loss: 0.4809 - val_accuracy: 0.8512
Epoch 31/70
65/65 [==============================] - 304s 4s/step - loss: 0.2002 - accuracy:
0.9360 - val_loss: 0.5827 - val_accuracy: 0.8329
Epoch 32/70
65/65 [==============================] - 301s 4s/step - loss: 0.1823 - accuracy:
0.9412 - val_loss: 0.3677 - val_accuracy: 0.8830
Epoch 33/70
65/65 [==============================] - 303s 4s/step - loss: 0.1729 - accuracy:
0.9451 - val_loss: 0.7272 - val_accuracy: 0.7987
Epoch 34/70
65/65 [==============================] - 302s 4s/step - loss: 0.1677 - accuracy:
0.9463 - val_loss: 0.7504 - val_accuracy: 0.8108
Epoch 35/70
65/65 [==============================] - 303s 4s/step - loss: 0.1538 - accuracy:
0.9510 - val_loss: 0.5355 - val_accuracy: 0.8532
Epoch 36/70
65/65 [==============================] - 316s 5s/step - loss: 0.2061 - accuracy:
0.9318 - val_loss: 0.4870 - val_accuracy: 0.8517
Epoch 37/70
65/65 [==============================] - 314s 5s/step - loss: 0.1389 - accuracy:
0.9584 - val_loss: 0.4489 - val_accuracy: 0.8671
Epoch 38/70
65/65 [==============================] - 310s 5s/step - loss: 0.1319 - accuracy:
0.9582 - val_loss: 0.8894 - val_accuracy: 0.7805
Epoch 39/70
65/65 [==============================] - 311s 5s/step - loss: 0.1120 - accuracy:
0.9668 - val_loss: 0.7458 - val_accuracy: 0.8026
Epoch 40/70
65/65 [==============================] - 305s 4s/step - loss: 0.1158 - accuracy:
0.9626 - val_loss: 0.4125 - val_accuracy: 0.8787
Epoch 41/70
65/65 [==============================] - 299s 4s/step - loss: 0.1285 - accuracy:
0.9607 - val_loss: 0.6612 - val_accuracy: 0.8142
Epoch 42/70
65/65 [==============================] - 299s 4s/step - loss: 0.1183 - accuracy:
0.9650 - val_loss: 0.4225 - val_accuracy: 0.8753
Epoch 43/70
65/65 [==============================] - 305s 4s/step - loss: 0.1129 - accuracy:
```

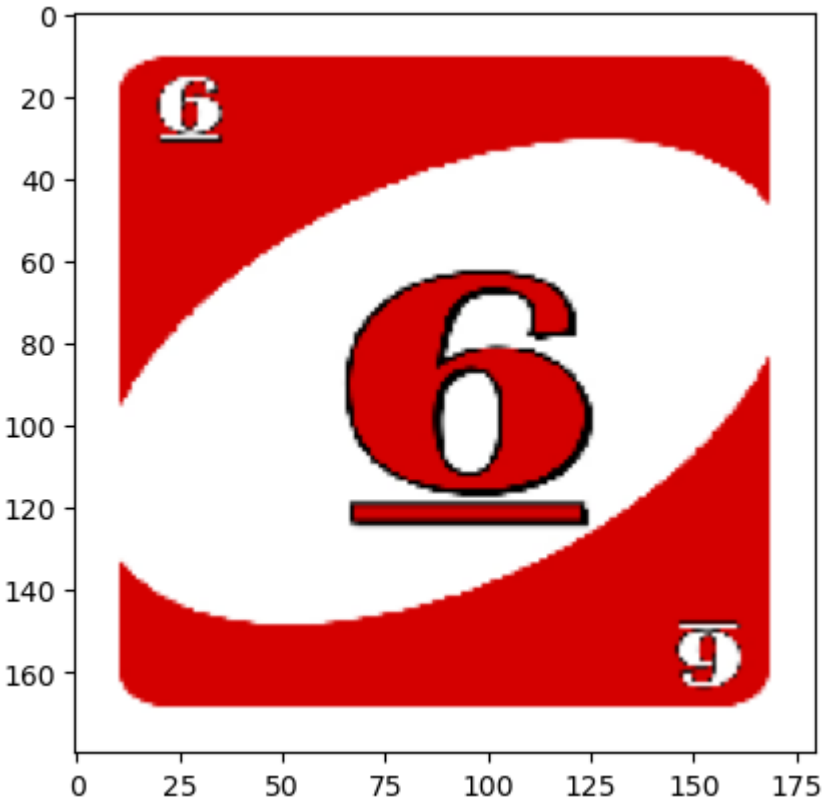
```
0.9655 - val_loss: 0.3957 - val_accuracy: 0.8758
Epoch 44/70
65/65 [==============================] - 299s 4s/step - loss: 0.1121 - accuracy:
0.9634 - val_loss: 0.6095 - val_accuracy: 0.8440
Epoch 45/70
65/65 [==============================] - 304s 4s/step - loss: 0.1042 - accuracy:
0.9659 - val_loss: 0.3965 - val_accuracy: 0.8820
Epoch 46/70
65/65 [==============================] - 304s 4s/step - loss: 0.1028 - accuracy:
0.9691 - val_loss: 0.7072 - val_accuracy: 0.8402
Epoch 47/70
65/65 [==============================] - 304s 4s/step - loss: 0.0765 - accuracy:
0.9777 - val_loss: 0.4487 - val_accuracy: 0.8777
Epoch 48/70
65/65 [==============================] - 306s 4s/step - loss: 0.0970 - accuracy:
0.9730 - val_loss: 0.5594 - val_accuracy: 0.8575
Epoch 49/70
65/65 [==============================] - 306s 4s/step - loss: 0.1060 - accuracy:
0.9669 - val_loss: 0.6561 - val_accuracy: 0.8397
Epoch 50/70
65/65 [==============================] - 314s 5s/step - loss: 0.0825 - accuracy:
0.9756 - val_loss: 0.4282 - val_accuracy: 0.8729
Epoch 51/70
65/65 [==============================] - 270s 4s/step - loss: 0.0752 - accuracy:
0.9777 - val_loss: 0.2235 - val_accuracy: 0.9278
Epoch 52/70
65/65 [==============================] - 285s 4s/step - loss: 0.0840 - accuracy:
0.9747 - val_loss: 0.5448 - val_accuracy: 0.8681
Epoch 53/70
65/65 [==============================] - 310s 5s/step - loss: 0.0793 - accuracy:
0.9741 - val_loss: 1.3295 - val_accuracy: 0.7390
Epoch 54/70
65/65 [==============================] - 287s 4s/step - loss: 0.2369 - accuracy:
0.9240 - val_loss: 0.6504 - val_accuracy: 0.8276
Epoch 55/70
65/65 [==============================] - 309s 5s/step - loss: 0.0956 - accuracy:
0.9721 - val_loss: 0.3717 - val_accuracy: 0.8931
Epoch 56/70
65/65 [==============================] - 305s 4s/step - loss: 0.0687 - accuracy:
0.9820 - val_loss: 0.3795 - val_accuracy: 0.8936
Epoch 57/70
65/65 [==============================] - 308s 5s/step - loss: 0.0627 - accuracy:
0.9801 - val_loss: 0.5090 - val_accuracy: 0.8690
Epoch 58/70
65/65 [==============================] - 307s 4s/step - loss: 0.0616 - accuracy:
0.9816 - val_loss: 0.2012 - val_accuracy: 0.9302
Epoch 59/70
65/65 [==============================] - 307s 4s/step - loss: 0.0529 - accuracy:
0.9851 - val_loss: 0.4294 - val_accuracy: 0.8724
Epoch 60/70
65/65 [==============================] - 306s 4s/step - loss: 0.0522 - accuracy:
0.9851 - val_loss: 0.4559 - val_accuracy: 0.8734
Epoch 61/70
65/65 [==============================] - 305s 4s/step - loss: 0.0699 - accuracy:
0.9788 - val_loss: 0.1991 - val_accuracy: 0.9326
Epoch 62/70
65/65 [==============================] - 308s 5s/step - loss: 0.0956 - accuracy:
0.9690 - val_loss: 0.9268 - val_accuracy: 0.7800
Epoch 63/70
65/65 [==============================] - 308s 5s/step - loss: 0.0711 - accuracy:
0.9786 - val_loss: 0.1600 - val_accuracy: 0.9470
Epoch 64/70
65/65 [==============================] - 307s 4s/step - loss: 0.0469 - accuracy:
0.9865 - val_loss: 0.8793 - val_accuracy: 0.8146
```

```
Epoch 65/70
65/65 [==============================] - 306s 4s/step - loss: 0.0615 - accuracy:
0.9805 - val_loss: 0.4939 - val_accuracy: 0.8647
Epoch 66/70
65/65 [==============================] - 307s 4s/step - loss: 0.0629 - accuracy:
0.9805 - val_loss: 0.3792 - val_accuracy: 0.9018
Epoch 67/70
65/65 [==============================] - 305s 5s/step - loss: 0.0737 - accuracy:
0.9774 - val_loss: 0.3497 - val_accuracy: 0.9148
Epoch 68/70
65/65 [==============================] - 310s 5s/step - loss: 0.0505 - accuracy:
0.9852 - val_loss: 0.1377 - val_accuracy: 0.9509
Epoch 69/70
65/65 [==============================] - 306s 4s/step - loss: 0.0662 - accuracy:
0.9799 - val_loss: 0.1917 - val_accuracy: 0.9427
Epoch 70/70
65/65 [==============================] - 306s 4s/step - loss: 0.0620 - accuracy:
0.9799 - val_loss: 0.7010 - val_accuracy: 0.8334
```

Out[28]:
```
<keras.src.callbacks.History at 0x1fac4557710>
```

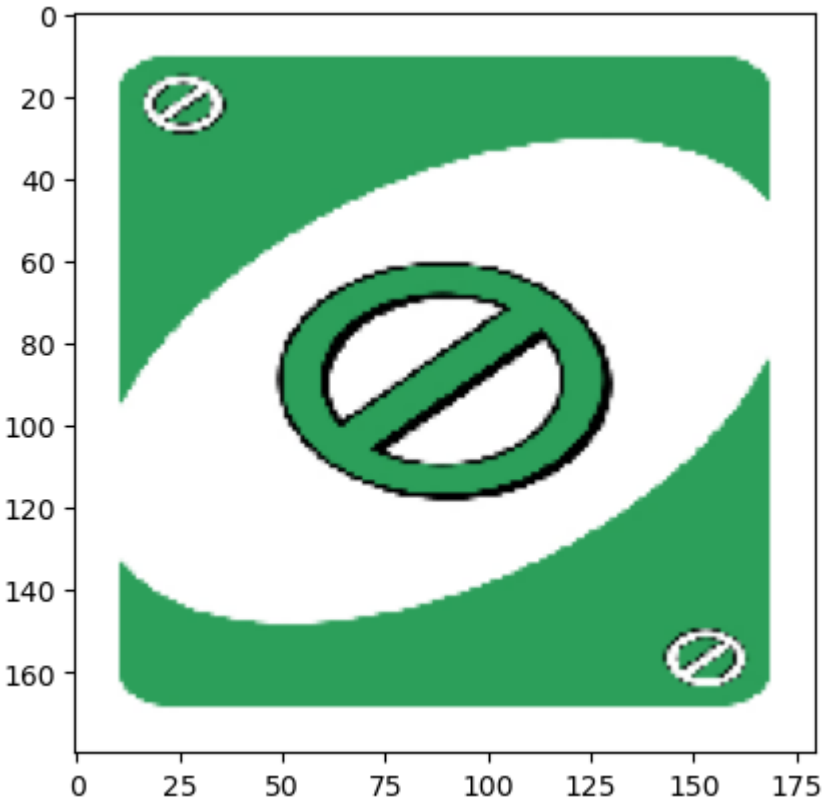
In [34]:
```
img = keras.utils.load_img("University/Dataset/Red/Red_6.jpg", target_size=image_si
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The predicted number is: {predicted_number}")
```

```
1/1 [==============================] - 0s 37ms/step
7
The predicted number is: 6
```



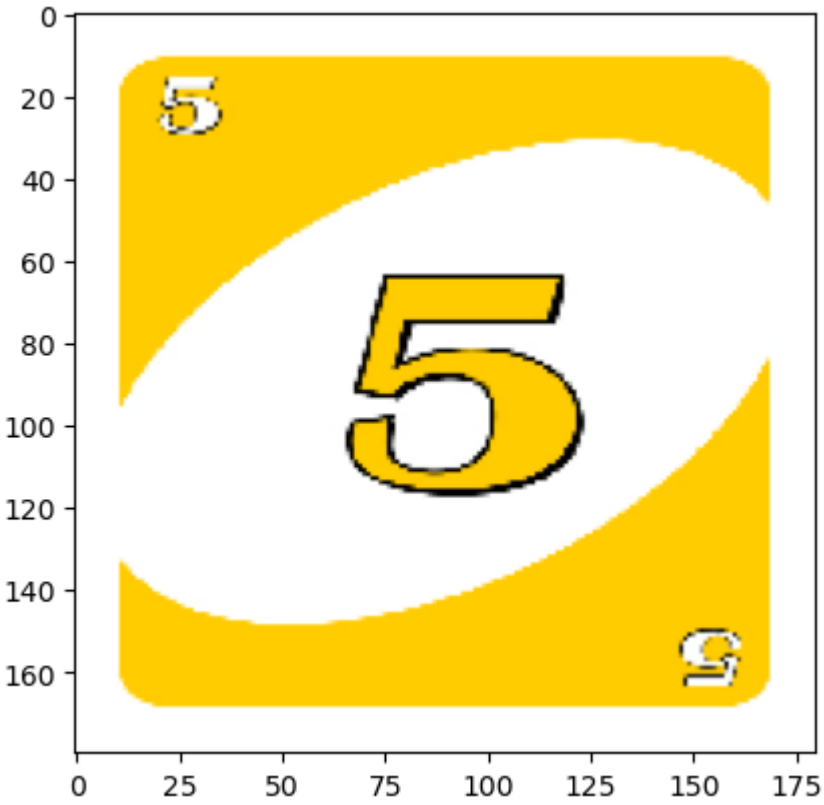
In [30]:
```
img = keras.utils.load_img("University/Dataset/green/Green_Skip.jpg", target_size=i
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The predicted number is: {predicted_number}")
```

```
1/1 [==============================] - 0s 52ms/step
12
The predicted number is: skip
```



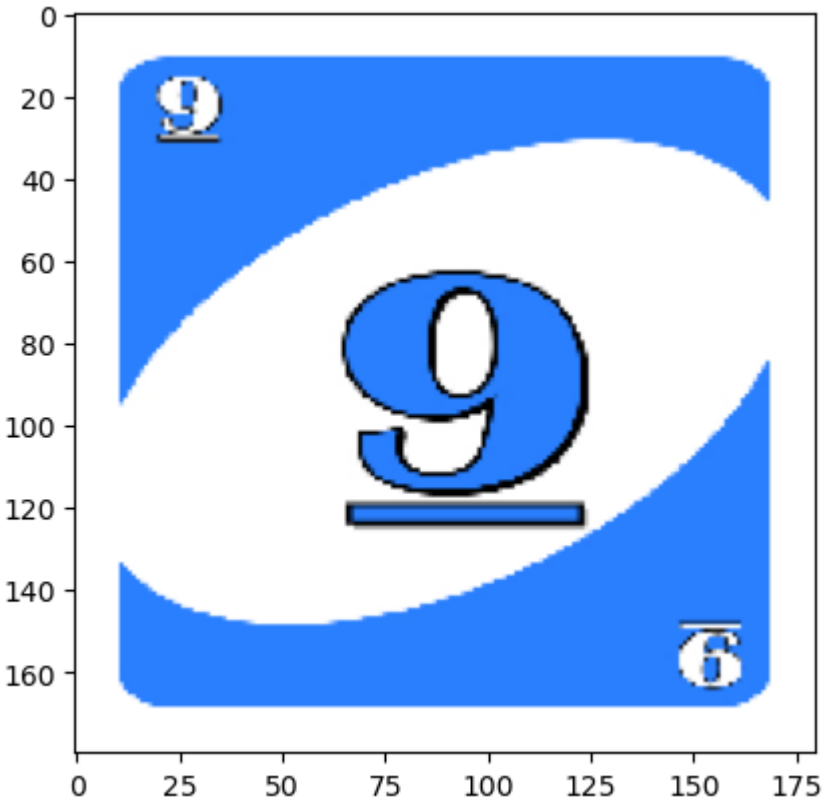
In [46]:
```
img = keras.utils.load_img("University/Dataset/yellow/Yellow_5.jpg", target_size=im
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The predicted number is: {predicted_number}")
```

```
1/1 [==============================] - 0s 24ms/step
6
The predicted number is: 5
```



In [41]:
```
img = keras.utils.load_img("University/Dataset/Blue/Blue_9.jpg", target_size=image_
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The predicted number is: {predicted_number}")
```

```
1/1 [==============================] - 0s 43ms/step
10
The predicted number is: 9
```

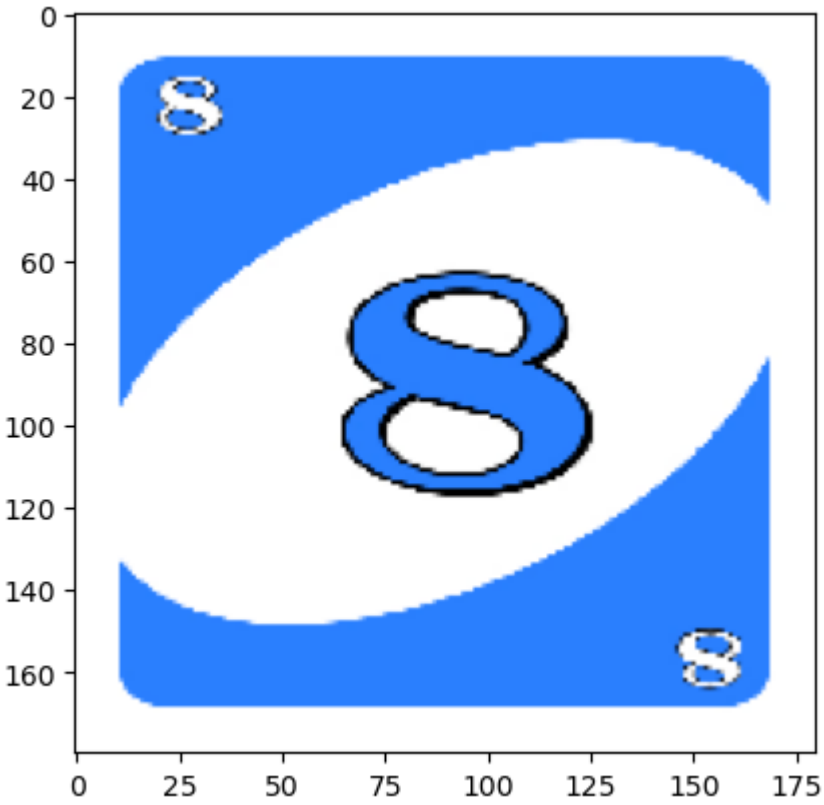


In [15]:
```
model1 = model1.save('University/model_4_numbers.keras')
```

Code for detecting the type of the card (both color and number)

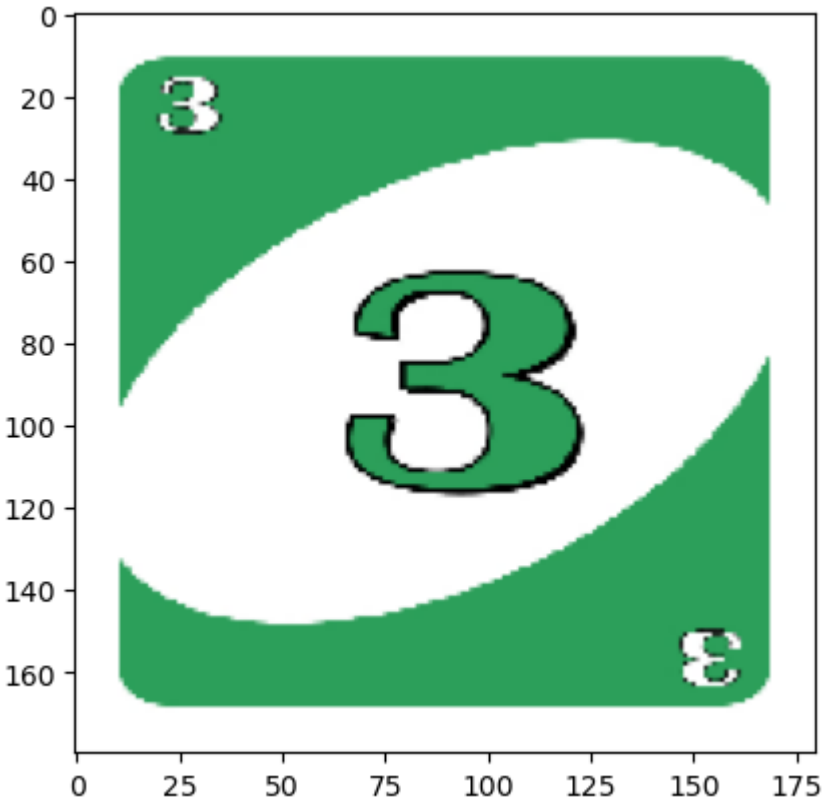
In [16]:
```
model = keras.models.load_model("University/modelofcolors.keras")
model1 = keras.models.load_model("University/model_4_numbers.keras")

image_size = (180, 180)
img = keras.utils.load_img("University/Dataset/Blue/Blue_8.jpg", target_size=image_
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
col_predictions = model.predict(img_array)
predicted_color_index = tf.argmax(col_predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]
predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The Card is: {predicted_color} {predicted_number}")
```

```
1/1 [==============================] - 0s 111ms/step
0
1/1 [==============================] - 0s 133ms/step
9
The Card is: Blue 8
```



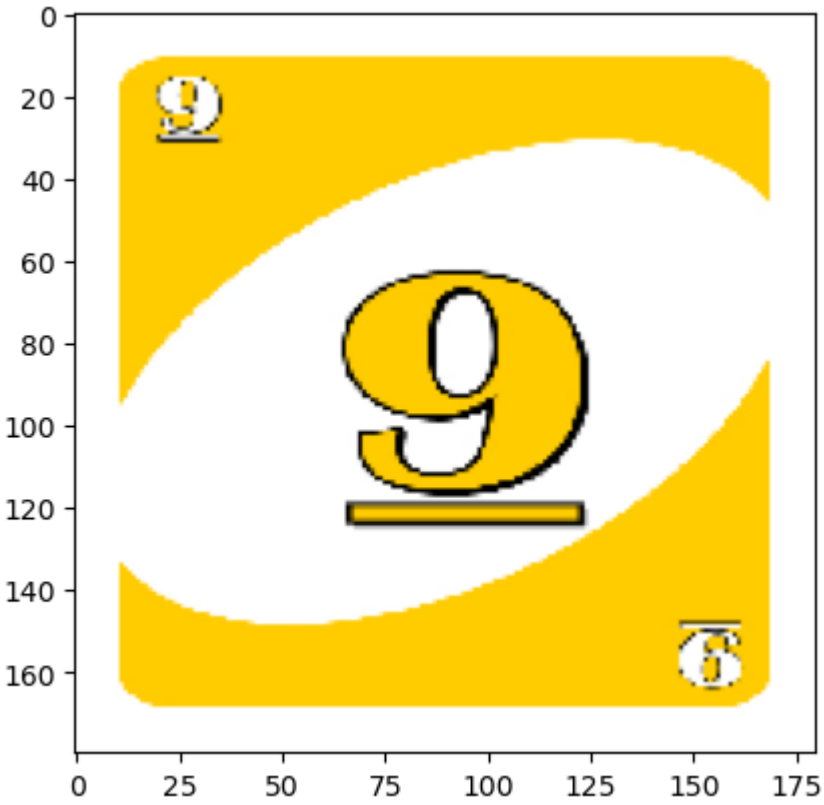
In [12]:
```
model = keras.models.load_model("University/modelofcolors.keras")
model1 = keras.models.load_model("University/model_4_numbers.keras")

image_size = (180, 180)
img = keras.utils.load_img("University/Dataset/green/Green_3.jpg", target_size=imag
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
col_predictions = model.predict(img_array)
predicted_color_index = tf.argmax(col_predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]
predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The Card is: {predicted_color} {predicted_number}")
```

```
1/1 [==============================] - 0s 88ms/step
2
1/1 [==============================] - 0s 96ms/step
10
The Card is: Green 9
```



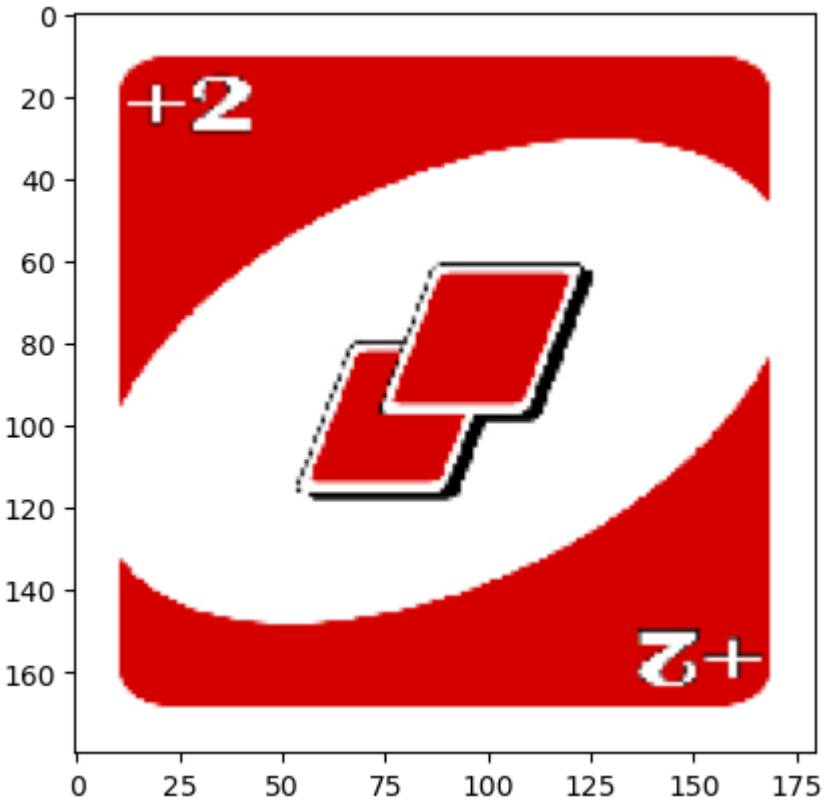
In [4]:
```
model = keras.models.load_model("University/modelofcolors.keras")
model1 = keras.models.load_model("University/model_4_numbers.keras")

image_size = (180, 180)
img = keras.utils.load_img("University/Dataset/yellow/Yellow_9.jpg", target_size=im
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
col_predictions = model.predict(img_array)
predicted_color_index = tf.argmax(col_predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]
predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The Card is: {predicted_color} {predicted_number}")
```

```
1/1 [==============================] - 0s 95ms/step
3
1/1 [==============================] - 0s 97ms/step
10
The Card is: Yellow 9
```



In [20]:
```
model = keras.models.load_model("University/modelofcolors.keras")
model1 = keras.models.load_model("University/model_4_numbers.keras")

image_size = (180, 180)
img = keras.utils.load_img("University/Dataset/Red/Red_Draw_2.jpg", target_size=ima
plt.imshow(img)

img_array = keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
col_predictions = model.predict(img_array)
predicted_color_index = tf.argmax(col_predictions, axis=1)[0].numpy()
color_names = ["Blue", "Red", "Green", "Yellow"]
print(predicted_color_index)
predicted_color = color_names[predicted_color_index]
predictions = model1.predict(img_array)
predicted_number_index = tf.argmax(predictions, axis=1)[0].numpy()
print(predicted_number_index)

number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "Revers
predicted_number = number_names[predicted_number_index]

print(f"The Card is: {predicted_color} {predicted_number}")
```

```
1/1 [==============================] - 0s 94ms/step
1
1/1 [==============================] - 0s 99ms/step
0
The Card is: Red Draw_2
```



Code for detecting the type of the card using a camera (openCV).

In [14]:
```
import cv2
```

In [15]:
```
color_model = keras.models.load_model("University/modelofcolors.keras")
number_model = keras.models.load_model("University/model_4_numbers.keras")

vc = cv2.VideoCapture(0)

predicted_number = ""

while vc.isOpened():
    rval, frame = vc.read()  # read frame
    target_size = (180, 180)
    frame_resized = cv2.resize(frame, target_size)
    img_array = keras.utils.img_to_array(frame_resized)
    img_array = tf.expand_dims(img_array, 0)  # Create batch axis

    predictions = color_model.predict(img_array)
    predicted_color_index = tf.argmax(predictions, axis=1)[0].numpy()
    color_names = ["Blue", "Red", "Green", "Yellow"]
    predicted_color = color_names[predicted_color_index]

    image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # convert to grayscale
    edges = cv2.Canny(image, 100, 200)
    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)

    # Create an empty mask
    mask = np.zeros_like(image)

    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 500:
            x, y, w, h = cv2.boundingRect(contour)
```

```
            # Draw the contour on the mask
            cv2.drawContours(mask, [contour], -1, (255, 255, 255),
                             thickness=cv2.FILLED)

            number_roi = frame[y:y + h, x:x + w]
            number_roi_resized = cv2.resize(number_roi, (180, 180))
            number_img_array = keras.utils.img_to_array(number_roi_resized)
            number_img_array = tf.expand_dims(number_img_array, 0)

            number_predictions = number_model.predict(number_img_array)
            predicted_number_index = tf.argmax(number_predictions, axis=1)[0].numpy
            number_names = ["Draw_2", "0", "1", "2", "3", "4", "5", "6",
                            "7", "8", "9", "Reverse", "skip"]
            predicted_number = number_names[predicted_number_index]

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    result_frame = cv2.bitwise_and(frame, frame, mask=mask)

    cv2.putText(result_frame, f"Color: {predicted_color}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    cv2.putText(result_frame, f"Number: {predicted_number}", (10, 70),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    cv2.imshow('Card Detection', result_frame)

    key = cv2.waitKey(1)
    if key == 27:  # exit on ESC
        break

cv2.destroyAllWindows()  # close all images
vc.release()
```

```
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
```

In [ ]: