```
!rm -rf clone && git clone https://github.com/Bileth/RUAP_projekt clone && cp -a clone/. .
```

```
Cloning into 'clone'...
remote: Enumerating objects: 71561, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 71561 (delta 2), reused 18 (delta 1), pack-reused 71539
Receiving objects: 100% (71561/71561), 649.46 MiB | 13.81 MiB/s, done.
Resolving deltas: 100% (2/2), done.
Checking out files: 100% (71431/71431), done.
```

```
import warnings
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import tensorflow as tf
```

```
from keras.models import load_model
from keras.preprocessing import image
import os
import numpy as np
import matplotlib.pyplot as plt
```

```
batch_size = 64
num_classes = 70
epochs = 600
model_name = "fruit_reco_model.h5"
save_path = ""

path_to_train = "Training"
path_to_test = "Test"
```

```
Generator = ImageDataGenerator()
train_data = Generator.flow_from_directory(path_to_train, (100, 100), batch_size=batch_siz

test_data = Generator.flow_from_directory(path_to_test, (100, 100), batch_size=batch_size)
```

```
Found 53276 images belonging to 70 classes.
Found 18025 images belonging to 70 classes.
```

```
model = Sequential()

model.add(Conv2D(16, (5, 5), input_shape=(100, 100, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
#model.add(Dropout(0.05))

model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
```

```
#model.add(Dropout(0.05))

model.add(Conv2D(64, (5, 5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
#model.add(Dropout(0.05))

model.add(Conv2D(128, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
#model.add(Dropout(0.05))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation="softmax"))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 96, 96, 16)        1216

 max_pooling2d_8 (MaxPooling  (None, 48, 48, 16)       0
 2D)

 conv2d_9 (Conv2D)           (None, 44, 44, 32)        12832

 max_pooling2d_9 (MaxPooling  (None, 22, 22, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 18, 18, 64)        51264

 max_pooling2d_10 (MaxPoolin  (None, 9, 9, 64)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 5, 5, 128)         204928

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 128)        0
 g2D)

 flatten_2 (Flatten)         (None, 512)               0

 dense_6 (Dense)             (None, 1024)              525312

 dropout_4 (Dropout)         (None, 1024)              0

 dense_7 (Dense)             (None, 256)               262400

 dropout_5 (Dropout)         (None, 256)               0

 dense_8 (Dense)             (None, 70)                17990
```

```
=================================================================
Total params: 1,075,942
Trainable params: 1,075,942
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=tf.keras.optimizers.Ada
```

```
history = model.fit(train_data, steps_per_epoch=60, epochs=epochs, verbose=1, validation_d
```

```
60/60 [==============================] - 4s 64ms/step - loss: 0.1321 - accuracy: 0.
Epoch 570/600
60/60 [==============================] - 4s 63ms/step - loss: 0.0841 - accuracy: 0.
Epoch 571/600
60/60 [==============================] - 4s 62ms/step - loss: 0.1553 - accuracy: 0.
Epoch 572/600
60/60 [==============================] - 4s 62ms/step - loss: 0.1543 - accuracy: 0.
Epoch 573/600
60/60 [==============================] - 4s 62ms/step - loss: 0.0628 - accuracy: 0.
Epoch 574/600
60/60 [==============================] - 4s 62ms/step - loss: 0.2621 - accuracy: 0.
Epoch 575/600
60/60 [==============================] - 4s 62ms/step - loss: 0.6979 - accuracy: 0.
Epoch 576/600
60/60 [==============================] - 4s 63ms/step - loss: 0.4287 - accuracy: 0.
Epoch 577/600
60/60 [==============================] - 4s 63ms/step - loss: 0.1324 - accuracy: 0.
Epoch 578/600
60/60 [==============================] - 4s 62ms/step - loss: 0.1176 - accuracy: 0.
Epoch 579/600
60/60 [==============================] - 4s 64ms/step - loss: 0.2342 - accuracy: 0.
Epoch 580/600
60/60 [==============================] - 4s 62ms/step - loss: 0.7185 - accuracy: 0.
Epoch 581/600
60/60 [==============================] - 4s 66ms/step - loss: 0.7320 - accuracy: 0.
Epoch 582/600
60/60 [==============================] - 4s 64ms/step - loss: 0.2216 - accuracy: 0.
Epoch 583/600
60/60 [==============================] - 4s 66ms/step - loss: 0.2137 - accuracy: 0.
Epoch 584/600
60/60 [==============================] - 4s 64ms/step - loss: 0.1829 - accuracy: 0.
Epoch 585/600
60/60 [==============================] - 4s 64ms/step - loss: 0.4174 - accuracy: 0.
Epoch 586/600
60/60 [==============================] - 4s 63ms/step - loss: 0.2248 - accuracy: 0.
Epoch 587/600
60/60 [==============================] - 4s 63ms/step - loss: 0.1890 - accuracy: 0.
Epoch 588/600
60/60 [==============================] - 4s 63ms/step - loss: 0.1334 - accuracy: 0.
Epoch 589/600
60/60 [==============================] - 4s 64ms/step - loss: 0.1709 - accuracy: 0.
Epoch 590/600
60/60 [==============================] - 4s 66ms/step - loss: 0.2826 - accuracy: 0.
Epoch 591/600
60/60 [==============================] - 4s 67ms/step - loss: 0.1039 - accuracy: 0.
Epoch 592/600
60/60 [==============================] - 4s 66ms/step - loss: 0.0286 - accuracy: 0.
Epoch 593/600
```

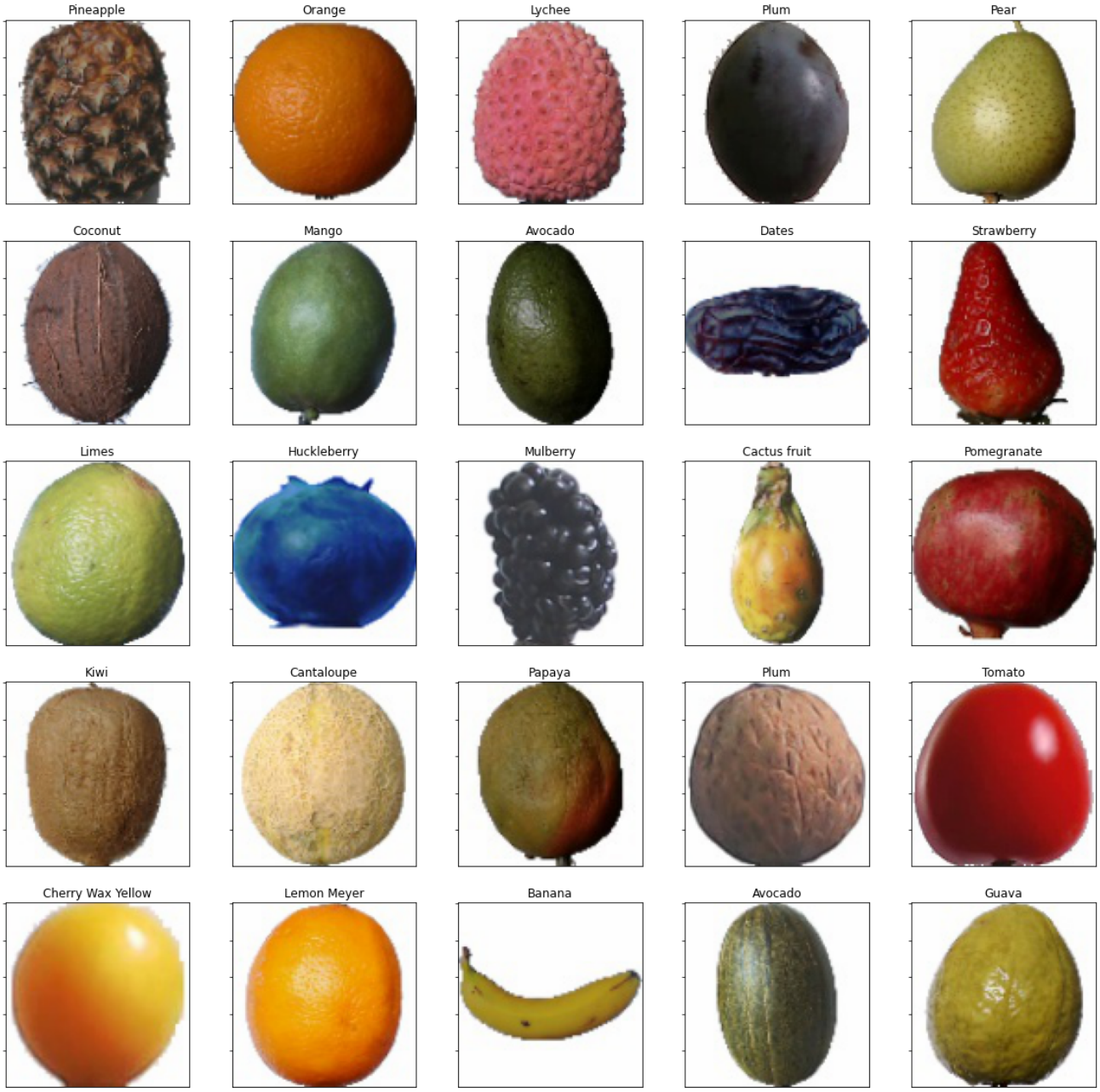
```
60/60 [==============================] - 4s 63ms/step - loss: 0.0728 - accuracy: 0.
Epoch 594/600
60/60 [==============================] - 4s 63ms/step - loss: 0.0104 - accuracy: 0.
Epoch 595/600
60/60 [==============================] - 4s 63ms/step - loss: 0.0694 - accuracy: 0.
Epoch 596/600
60/60 [==============================] - 4s 66ms/step - loss: 0.2376 - accuracy: 0.
Epoch 597/600
60/60 [==============================] - 4s 64ms/step - loss: 0.8117 - accuracy: 0.
```

```
model.save(model_name)
```

```
class Fruit:

    def __init__(self, img_dir = ''):
        self.img_dir = img_dir
        self.cnt = 0
        self.batch_holder = None
        self.model =  tf.keras.models.load_model('fruit_reco_model.h5')
        self.Label_dict = labels =  {
             'Apple': 0,
             'Apple Braeburn': 1,
             'Apple Crimson Snow': 2,
             'Apple Golden': 3,
             'Apple Granny Smith': 4,
             'Apple Pink Lady': 5,
             'Apple Red Delicious': 6,
             'Apricot': 7,
             'Avocado': 8,
             'Banana': 9,
             'Banana Lady Finger': 10,
             'Blueberry': 11,
             'Cactus fruit': 12,
             'Cantaloupe': 13,
             'Carambula': 14,
             'Cherry': 15,
             'Cherry Rainer': 16,
             'Cherry Wax Black': 17,
             'Cherry Wax Red': 18,
             'Cherry Wax Yellow': 19,
             'Clementine': 20,
             'Coconut': 21,
             'Corn': 22,
             'Dates': 23,
             'Eggplant': 24,
             'Fig': 25,
             'Grape Blue': 26,
             'Grape Pink': 27,
             'Grape White': 28,
             'Grapefruit Pink': 29,
             'Grapefruit White': 30,
             'Guava': 31,
             'Huckleberry': 32,
```

```
            'Kaki': 33,
            'Kiwi': 34,
            'Kumquats': 35,
            'Lemon': 36,
            'Lemon Meyer': 37,
            'Limes': 38,
            'Lychee': 39,
            'Mandarine': 40,
            'Mango': 41,
            'Mangostan': 42,
            'Maracuja': 43,
            'Mulberry': 44,
            'Nectarine': 45,
            'Orange': 46,
            'Papaya': 47,
            'Passion Fruit': 48,
            'Peach': 49,
            'Pear': 50,
            'Pear Abate': 51,
            'Pear Forelle': 52,
            'Pear Kaiser': 53,
            'Pear Monster': 54,
            'Pear Red': 55,
            'Pear Stone': 56,
            'Pear Williams': 57,
            'Pepper': 58,
            'Pineapple': 59,
            'Pitahaya Red': 60,
            'Plum': 61,
            'Pomegranate': 62,
            'Pomelo Sweetie': 63,
            'Quince': 64,
            'Raspberry': 65,
            'Redcurrant': 66,
            'Strawberry': 67,
            'Tomato': 68,
            'Watermelon': 69}
        self.label = list(self.Label_dict.keys())

    def read_images(self):
        self.cnt = len(os.listdir(self.img_dir))
        self.batch_holder = np.zeros((self.cnt, 100, 100, 3))
        for i,img in enumerate(os.listdir(self.img_dir)):
            img = image.load_img(os.path.join(self.img_dir,img), target_size=(100, 100))
            self.batch_holder[i, :] = img
        return self.batch_holder

    def predict(self):
        fig = plt.figure(figsize=(20, 20))
        for i,img in enumerate(self.batch_holder):
            fig.add_subplot(5, 5, i+1)
            result=self.model.predict(self.batch_holder)
            result_classes = result.argmax(axis=-1)
            plt.title(self.label[result_classes[i]])
            plt.tick_params(
```

```
                axis='both',
                which='both',
                bottom=False,
                top=False,
                labelbottom=False,
                labelleft=False)
            plt.imshow(img/256.)
        plt.show()


obj = Fruit('testing')


obj.read_images()
obj.predict()
```



✓ 24s completed at 5:25 PM