Grapevine Leaves Image Classification

AUGUST 25, 2022

MUHAMMAD AMIN GHIAS

Table of Contents

ABSTRACT:

Fruits are important part of a healthy diet. Grapes is one of the most loved fruits. There are many species of grapes all with their unique qualities and importance. In this project we will help in finding the different types of grapevine leaves from one another. We will classify the leaves to their respective classes efficiently using neural networks – deep learning model of CNN.

INTRODUCTION

Grapes are used fresh or processed. Grapevine leaves are collected once a year as a by-product. The species of grapevine leaves which are collected once a year have unique price and taste and telling apart from different species of grapevine leaves is a difficult timely task. We will use our deep learning model and CNN to classify these grapevine leaves to their respective species

MAIN OBJECTIVE:

The main objective of this project is to study the different 5 species of grapevine leaves and classify them efficiently and accurately to its different classes.

DATA ACQUISITION AND WRANGLING

DATA SOURCE

For this project we will use online data the source is from kaggle

files are from url:

https://www.kaggle.com/datasets/muratkokludataset/grapevine-leaves-image-dataset

This dataset consists of images of 500 vine leaves belonging to 5 species were taken with a special self-illuminating system.

DATA DESCRIPTION

The dataset consists of 100 pictures of 5 species of grapevine leaves which are:

- Ak
- Ala_Idris
- Buzgulu
- Dimnit
- Nazli

DATA EXPLORATION CLEANING & ANALYSIS

Making dataframe

All the images were extracted and formulated in a dataset, the resultant datsest was made whith the following shape

```
1  df.shape
✓ 0.8s

(500, 6)
```

Making the x and y train and test datasets

```
1 x_train, x_test, y_train, y_test = train_test_split(dataset, label, test_size = 0.2, random_state = 42)
2
3 print(x_train.shape)
4 print(y_train.shape)
5
6 x_train = normalize( x_train, axis =1)
7 x_test = normalize( x_test, axis =1)
8
✓ 0.1s

(400, 64, 64, 3)
(400,)
```

converting y to categorical format

As we have 5 classes hence y has to be converted

```
converting y to categorical format

1 num_classes = 5
2
3 y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
4 y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)
✓ 0.6s
```

Applying CNN models

We will use Convolution neural networks model for classification

Model-1

```
model_1 = Sequential()


## 5x5 convolution with 2x2 stride and 32 filters
model_1.add(Conv2D(32, (5, 5), strides = (2,2), padding='same',
                   input_shape=x_train.shape[1:]))
model_1.add(Activation('relu'))

## Another 5x5 convolution with 2x2 stride and 32 filters
model_1.add(Conv2D(32, (5, 5), strides = (2,2)))
model_1.add(Activation('relu'))

## 2x2 max pooling reduces to 3 x 3 x 32
model_1.add(MaxPooling2D(pool_size=(2, 2)))
model_1.add(Dropout(0.25))

## Flatten turns 3x3x32 into 288x1
model_1.add(Flatten())
model_1.add(Dense(512))
model_1.add(Activation('relu'))
model_1.add(Dropout(0.5))
model_1.add(Dense(num_classes))
model_1.add(Activation('softmax'))

model_1.summary()
```

✓ 0.7s

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        2432

activation (Activation)      (None, 32, 32, 32)        0

conv2d_1 (Conv2D)            (None, 14, 14, 32)        25632

activation_1 (Activation)    (None, 14, 14, 32)        0

max_pooling2d (MaxPooling2D  (None, 7, 7, 32)          0
)

dropout (Dropout)            (None, 7, 7, 32)          0

flatten (Flatten)            (None, 1568)              0

dense (Dense)                (None, 512)               803328

activation_2 (Activation)    (None, 512)               0

dropout_1 (Dropout)          (None, 512)               0

...
Total params: 833,957
Trainable params: 833,957
Non-trainable params: 0
_________________________________________________________________
```

Choosing Optimizer

Using optimizers RMSprop(lr=0.0005, decay=1e-6)

```
batch_size = 32

# initiate RMSprop optimizer
opt = tensorflow.keras.optimizers.RMSprop(lr=0.0005, decay=1e-6)


# Let's train the model_1 using RMSprop
model_1.compile(loss='categorical_crossentropy',
                optimizer=opt,
                metrics=['accuracy'])

t= now()
model_1.fit(x_train, y_train,
                batch_size=batch_size,
                epochs=50,
                validation_data=(x_test, y_test),
                shuffle=True)

print('Training time: %s' % (now() - t))

score = model_1.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

✓ 21.5s

```
Epoch 1/50
```

Training time: 0:00:21.436273
Test score: 1.5228713750839233
Test accuracy: 0.4300000071525574

Using optimizers Adam(learning_rate=0.001)

```
batch_size = 32

# initiate RMSprop optimizer
# opt = tensorflow.keras.optimizers.RMSprop(lr=0.0005, decay=1e-6)
opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Let's train the model_1 using RMSprop
model_1.compile(loss='categorical_crossentropy',
                optimizer=opt,
                metrics=['accuracy'])

t=now()

model_1.fit(x_train, y_train,
                batch_size=batch_size,
                epochs=50,
                validation_data=(x_test, y_test),
                shuffle=True)

print('Training time: %s' % (now() - t))

score = model_1.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

✓ 21.9s

Training time: 0:00:21.828391
Test score: 2.5830678939819336
Test accuracy: 0.44999998807907104
Hence Optimizer Adam(learning_rate=0.001) is best and will be used for all models

Model-2

2nd Model

```
# Let's build a CNN using Keras' Sequential capabilities

model_2 = Sequential()

model_2.add(Conv2D(32, (3, 3), padding='same',
                   input_shape=x_train.shape[1:]))
model_2.add(Activation('relu'))
model_2.add(Conv2D(32, (3, 3)))
model_2.add(Activation('relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
model_2.add(Dropout(0.25))

model_2.add(Conv2D(64, (3, 3), padding='same'))
model_2.add(Activation('relu'))
model_2.add(Conv2D(64, (3, 3)))
model_2.add(Activation('relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
model_2.add(Dropout(0.25))

model_2.add(Flatten())
model_2.add(Dense(512))
model_2.add(Activation('relu'))
model_2.add(Dropout(0.5))
model_2.add(Dense(num_classes))
model_2.add(Activation('softmax'))

model_2.summary()
```

✓ 0.1s

Output exceeds the size limit. Open the full output data in a text

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 64, 64, 32)        896

 activation_4 (Activation)   (None, 64, 64, 32)        0

 conv2d_3 (Conv2D)           (None, 62, 62, 32)        9248

 activation_5 (Activation)   (None, 62, 62, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 dropout_2 (Dropout)         (None, 31, 31, 32)        0

 conv2d_4 (Conv2D)           (None, 31, 31, 64)        18496

 activation_6 (Activation)   (None, 31, 31, 64)        0

 conv2d_5 (Conv2D)           (None, 29, 29, 64)        36928

 activation_7 (Activation)   (None, 29, 29, 64)        0

...
Total params: 6,491,173
Trainable params: 6,491,173
Non-trainable params: 0
_________________________________________________________________
```

```python
batch_size = 32

# initiate RMSprop optimizer
# opt = tensorflow.keras.optimizers.RMSprop(lr=0.0005, decay=1e-6)
opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Let's train the model_2 using RMSprop
model_2.compile(loss='categorical_crossentropy',
                optimizer=opt,
                metrics=['accuracy'])

t=now()
model_2.fit(x_train, y_train,
                batch_size=batch_size,
                epochs=50,
                validation_data=(x_test, y_test),
                shuffle=True)

print('Training time: %s' % (now() - t))

score = model_2.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

✓ 2m 11.5s

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/50
```

Training time: 0:02:11.321232
Test score: 3.2041022777557373
Test accuracy: 0.550000011920929

Model-3

```python
# Let's build a CNN using Keras' Sequential capabilities

model_3 = Sequential()

model_3.add(Conv2D(32, (3, 3), padding='same',
                   input_shape=x_train.shape[1:]))
model_3.add(Activation('relu'))
model_3.add(Conv2D(32, (3, 3)))
model_3.add(Activation('relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))

model_3.add(Conv2D(64, (3, 3), padding='same'))
model_3.add(Activation('relu'))
model_3.add(Conv2D(64, (3, 3)))
model_3.add(Activation('relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))

model_3.add(Conv2D(32, (3, 3), padding='same'))
model_3.add(Activation('relu'))
model_3.add(Conv2D(32, (3, 3)))
model_3.add(Activation('relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))

model_3.add(Conv2D(64, (3, 3), padding='same'))
model_3.add(Activation('relu'))
model_3.add(Conv2D(64, (3, 3)))
model_3.add(Activation('relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))

model_3.add(Flatten())
model_3.add(Dense(512))
model_3.add(Activation('relu'))
model_3.add(Dropout(0.5))
model_3.add(Dense(num_classes))
model_3.add(Activation('softmax'))

model_3.summary()
```

✓ 0.2s

```
Output exceeds the size limit. Open the full output data in a text edi
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 64, 64, 32)        896

 activation_10 (Activation)  (None, 64, 64, 32)        0

 conv2d_7 (Conv2D)           (None, 62, 62, 32)        9248

 activation_11 (Activation)  (None, 62, 62, 32)        0

 max_pooling2d_3 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 dropout_5 (Dropout)         (None, 31, 31, 32)        0

 conv2d_8 (Conv2D)           (None, 31, 31, 64)        18496

 activation_12 (Activation)  (None, 31, 31, 64)        0

 conv2d_9 (Conv2D)           (None, 29, 29, 64)        36928

 activation_13 (Activation)  (None, 29, 29, 64)        0

...
Total params: 282,853
Trainable params: 282,853
Non-trainable params: 0
_________________________________________________________________
```

```python
batch_size = 32

# initiate RMSprop optimizer
opt = tensorflow.keras.optimizers.RMSprop(lr=0.0005, decay=1e-6)
# opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Let's train the model_3 using RMSprop
model_3.compile(loss='categorical_crossentropy',
                optimizer=opt,
                metrics=['accuracy'])

t=now()

model_3.fit(x_train, y_train,
                batch_size=batch_size,
                epochs=20,
                validation_data=(x_test, y_test),
                shuffle=True)

print('Training time: %s' % (now() - t))

score = model_3.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

✓ 55.3s

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/20
13/13 [==============================] - 4s 210ms/step - loss: 1.6114 - accuracy
```

Training time: 0:00:55.094510
Test score: 1.3176687955856323
Test accuracy: 0.5099999904632568

Winner model

The model-2 is the best model with accuracy of 0.55

We will use this model 2 further

Using Transfer Learning on winner model

Using transfer learning for winner model 2

```
feature_layers = [

    (Conv2D(32, (3, 3), padding='same',
                        input_shape=x_train.shape[1:])),
    (Activation('relu')),
    (Conv2D(32, (3, 3))),
    (Activation('relu')),
    (MaxPooling2D(pool_size=(2, 2))),
    (Dropout(0.25)),

    (Conv2D(64, (3, 3), padding='same')),
    (Activation('relu')),
    (Conv2D(64, (3, 3))),
    (Activation('relu')),
    (MaxPooling2D(pool_size=(2, 2))),
    (Dropout(0.25)),

    (Flatten())

]


classification_layers = [
    (Dense(512)),
    (Activation('relu')),
    (Dropout(0.5)),
    (Dense(num_classes)),
    (Activation('softmax'))
]

model = Sequential(feature_layers + classification_layers)

model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 64, 64, 32)        896

 activation_24 (Activation)  (None, 64, 64, 32)        0

 conv2d_17 (Conv2D)          (None, 62, 62, 32)        9248

 activation_25 (Activation)  (None, 62, 62, 32)        0

 max_pooling2d_8 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 dropout_12 (Dropout)        (None, 31, 31, 32)        0

 conv2d_18 (Conv2D)          (None, 31, 31, 64)        18496

 activation_26 (Activation)  (None, 31, 31, 64)        0

 conv2d_19 (Conv2D)          (None, 29, 29, 64)        36928

 activation_27 (Activation)  (None, 29, 29, 64)        0

...
Total params: 6,491,173
Trainable params: 6,491,173
Non-trainable params: 0
_________________________________________________________________
```

```
batch_size = 32

opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Let's train the model_3 using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])


t = now()


model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=50,
              validation_data=(x_test, y_test),
              shuffle=True)


print('Training time: %s' % (now() - t))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

Output exceeds the size limit. Open the full output data in a

Training time: 0:02:15.397319
Test score: 3.3391921520233154
Test accuracy: 0.5099999904632568

Using Freezing Layer on winner Model

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 64, 64, 32)        896

 activation_24 (Activation)  (None, 64, 64, 32)        0

 conv2d_17 (Conv2D)          (None, 62, 62, 32)        9248

 activation_25 (Activation)  (None, 62, 62, 32)        0

 max_pooling2d_8 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 dropout_12 (Dropout)        (None, 31, 31, 32)        0

 conv2d_18 (Conv2D)          (None, 31, 31, 64)        18496

 activation_26 (Activation)  (None, 31, 31, 64)        0

 conv2d_19 (Conv2D)          (None, 29, 29, 64)        36928

 activation_27 (Activation)  (None, 29, 29, 64)        0

...
Total params: 6,491,173
Trainable params: 6,425,605
Non-trainable params: 65,568
_________________________________________________________________
```

```
batch_size = 32

opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Let's train the model_3 using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])


t = now()


model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=50,
              validation_data=(x_test, y_test),
              shuffle=True)


print('Training time: %s' % (now() - t))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
Output exceeds the size limit. Open the full output data in a text edit
Epoch 1/50
```

Training time: 0:00:52.449559
Test score: 4.026365280151367
Test accuracy: 0.5400000214576721

RESULTS:

CNN has been used for classification grapevine leaves, the best Optimizer is Adam with learning_rate=0.001 and the best model is model-2 gives which gives accuracy of 0.55 and with transfer learning and freezing layer the accuracy is 0.54

DISCUSSION:

In this project we used CNN and found the following:

- Best Optimizer
- Best model with greater accuracy

We also used transfer learning and freezing layer

CONCLUSION

The results of model 2 is best as compared to other models and can be used to classify the grapevine leaves and achieve our goal.

SUGGESTIONS

The dataset only had 500 images greater bigger dataset will improve the accuracy of our model