# Task DL. The Colruyt Deep Learning Challenge

- Student name: Yori Verbist
- Number: R0697651

Last year, you probably heard about a pilot project of Colruyt, one of the largest supermarket chains in Belgium. Since August 2019, Colruyt has been using Deep Learning in two of its supermarkets to recognize fruit and vegetables and weight them automatically. Now it's up to you to build and train such an image recognition system yourself.

***We are also going to do a small competition.*** *Try to achieve an accuracy as high as possible. During the last lesson, we will give you 20 images of fruits. The student who can classify the most images correctly wins the competition. In case of an ex aequo, the student with the highest accuracy wins. Eternal fame will be his part.*

You can read and watch the news item on vrt via the link below (in Dutch).

vrt NWS

Colruyt made a little video which you can watch below.

In [1]:
```
%%HTML
<iframe width="640" height="360" src="https://www.youtube.com/embed/RPjt1-
```

## 1. The dataset

First you need a large amount of train and test images of fruits. I've downloaded about 500 images of 10 different fruits from https://www.flickr.com/. You can find them in this rar-file:

Colruyt

Start by unpacking the training/validation dataset. First have a look at the different folders and images.

## 2. Binary classification

Next, build and train a classifier (like we did with the cats and dogs example). At first make the classification task a little bit simpler with only two fruits (binary = two-class). This will result in lower complexity and hopefully faster training on your laptop.

`What two fruits did you select?` apples and peaches because they look alike

`Insert a screenshot of the training process and the final accuracy achieved.`

In [29]:
```
from fastai.vision.all import *
from pathlib import Path
```

In [64]:
```
# Insert the code for building and training your classifier.
path1 = Path("binary")

fruits1 = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=RandomResizedCrop(224, min_scale=0.5),
    batch_tfms=aug_transforms())


dls = fruits1.dataloaders(path1)
#dls.valid.show_batch(max_n=16, nrows=4)

learn1 = cnn_learner(dls, resnet18, metrics=accuracy)
learn1.fine_tune(5)
```

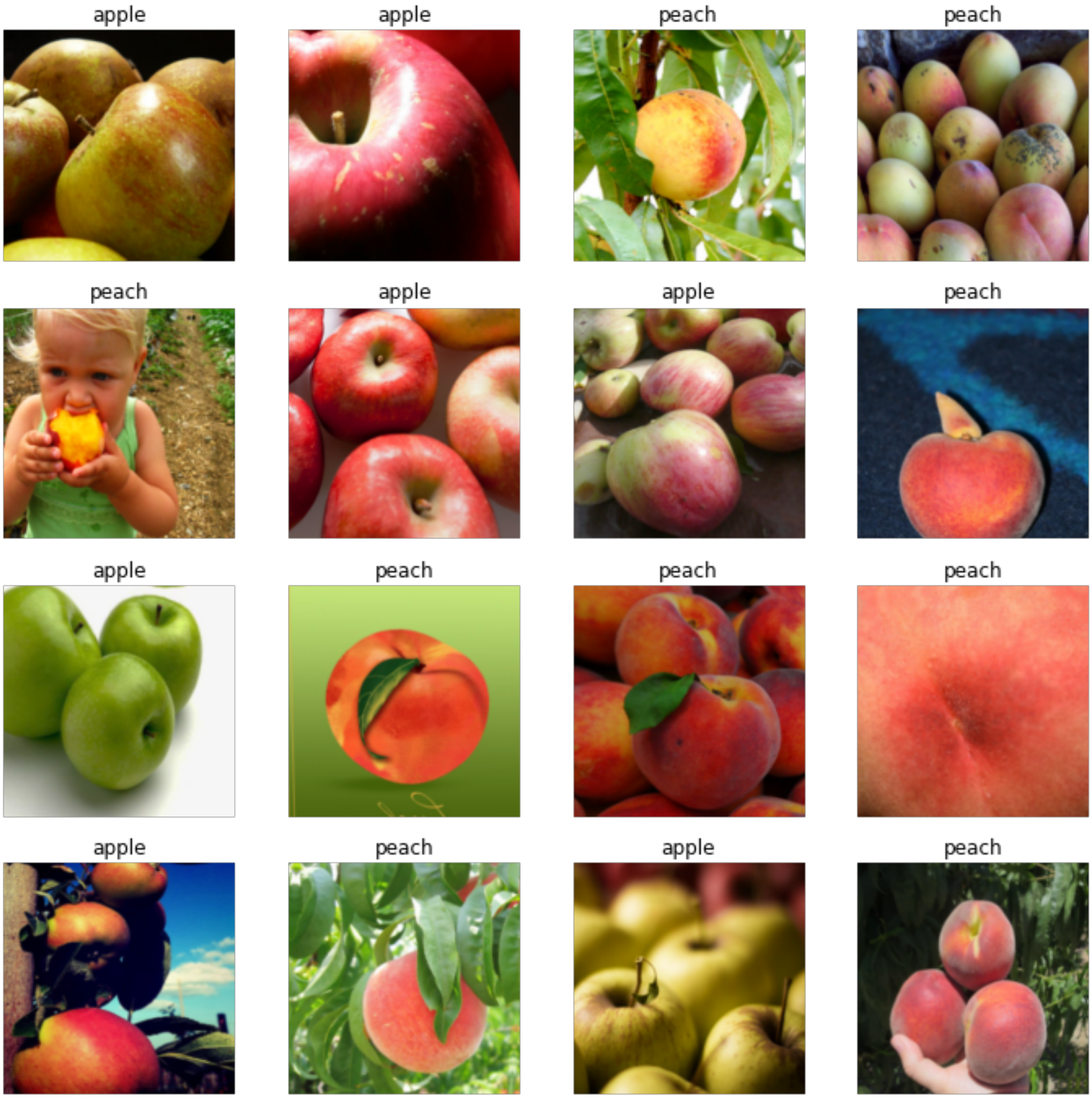
| epoch | train_loss | valid_loss | accuracy | time |
|---|---|---|---|---|
| 0 | 0.893534 | 0.264972 | 0.896341 | 00:04 |

| epoch | train_loss | valid_loss | accuracy | time |
|---|---|---|---|---|
| 0 | 0.312991 | 0.127075 | 0.951219 | 00:05 |
| 1 | 0.270857 | 0.180389 | 0.939024 | 00:05 |
| 2 | 0.234716 | 0.103726 | 0.963415 | 00:05 |
| 3 | 0.182895 | 0.095573 | 0.969512 | 00:05 |
| 4 | 0.148316 | 0.097250 | 0.969512 | 00:05 |

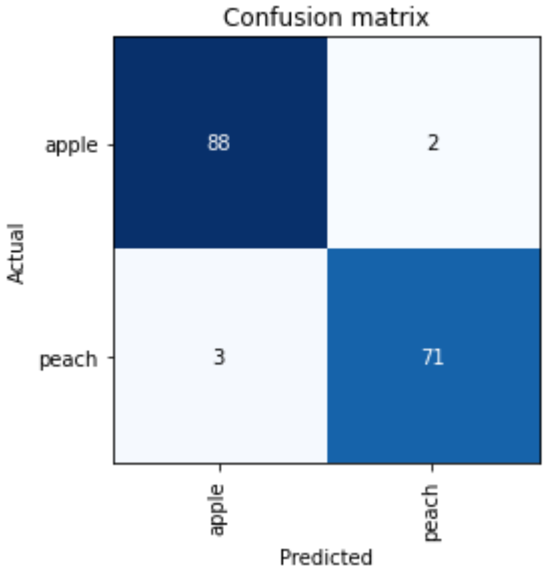
In [69]:
```
learn1.recorder.values[-1][-1]
```

Out[69]:
```
0.9695122241973877
```

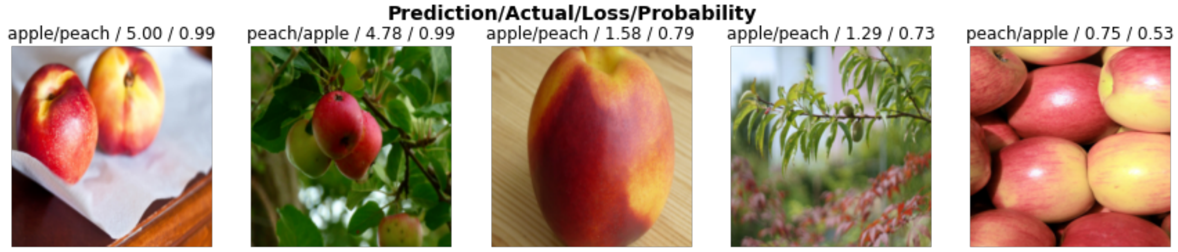
In [70]:
```
dls.train.show_batch(max_n=16, nrows=4)
```



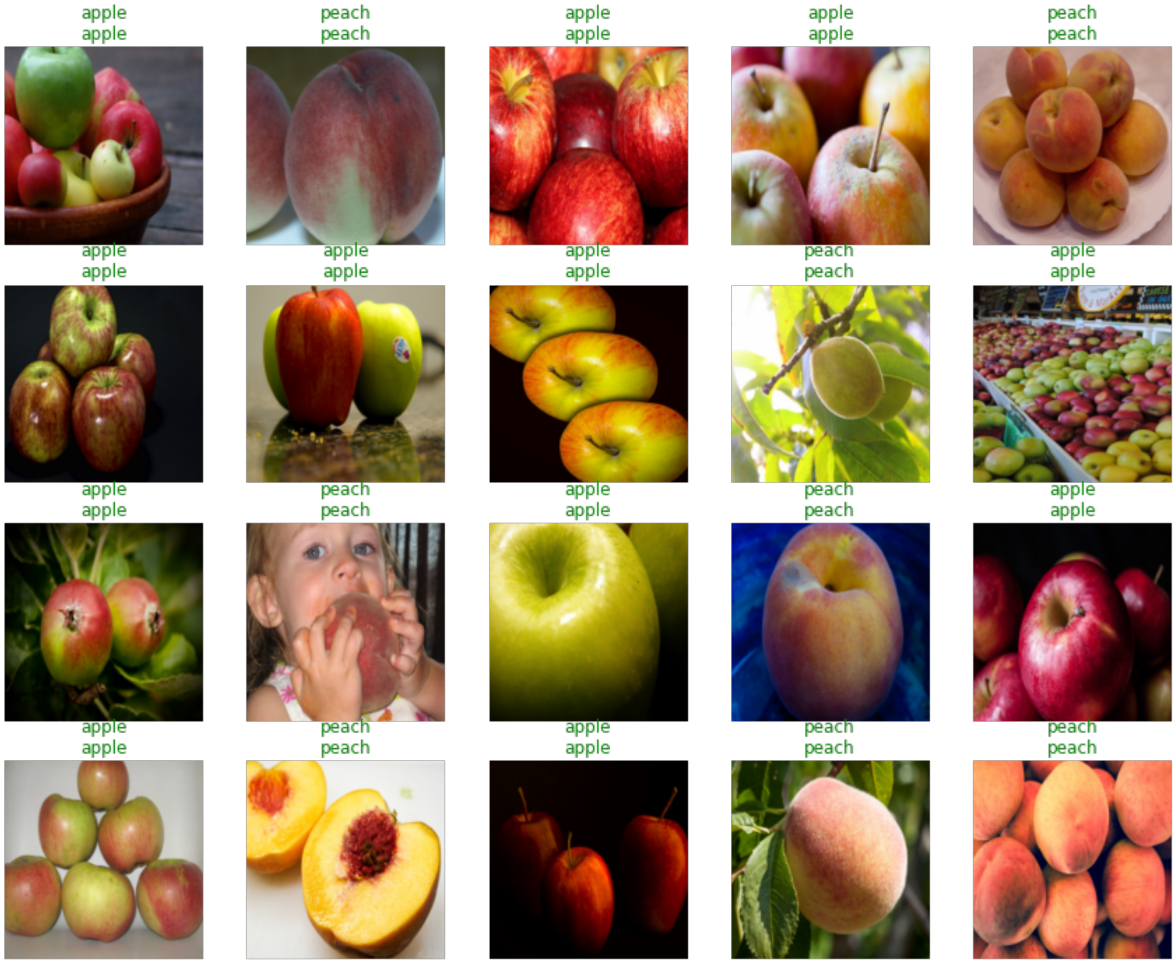
In [71]:
```
interp1 = ClassificationInterpretation.from_learner(learn1)
interp1.plot_confusion_matrix()
```



In [72]:
```
interp1.plot_top_losses(5, nrows=1)
```


Prediction/Actual/Loss/Probability
apple/peach / 5.00 / 0.99
peach/apple / 4.78 / 0.99
apple/peach / 1.58 / 0.79
apple/peach / 1.29 / 0.73
peach/apple / 0.75 / 0.53

```
In [73]: learn1.show_results(max_n=20)
```



Download at least ten images of the two fruits (five of each) you selected and see if your model can classify them correctly.

## 3. Classification with all fruits

Now we make things a little more complicated and challenging. Repeat the proces from above with all the (10) classes. It might take some time now to train the model! You also need to change your program because there are 10 classes now.

Once you're finished, try to achieve an accuracy as high as possible. This is not an exact science. So you will have to experiment with the parameters of your CNN. Possible things to consider:

- the input shape of the images
- the number of filters

- the use of layers to prevent overfitting
- the number of layers
- the number of epochs
- the number of steps per epoch
- ...

**It might be a good idea to have a look at some existing famous CNN Architectures for Image Classification (VGGNet, ...), built by researchers around the world. So search the Internet and find some good examples of CNN's and look for some best practices.**

This website can be a good start: Five Powerful CNN Architectures

Write a small report (10 lines) about the things you've tried to optimize your classifier and which configuration gave the best results. Where did you find the information?

Insert a screenshot of the training process and the final accuracy achieved.

In [40]:

```
from fastai.vision.all import *
from pathlib import Path
```

In [49]:

```
# Insert the code for building and training your classifier.
path = Path("all")

fruits = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=RandomResizedCrop(224, min_scale=0.5),
    batch_tfms=aug_transforms())

dls = fruits.dataloaders(path)
#dls.valid.show_batch(max_n=16, nrows=4)

learn = cnn_learner(dls, resnet18, metrics=accuracy)
learn.fine_tune(20)
```

| epoch | train_loss | valid_loss | accuracy | time |
|---|---|---|---|---|
| 0 | 1.143781 | 0.275712 | 0.919717 | 00:11 |

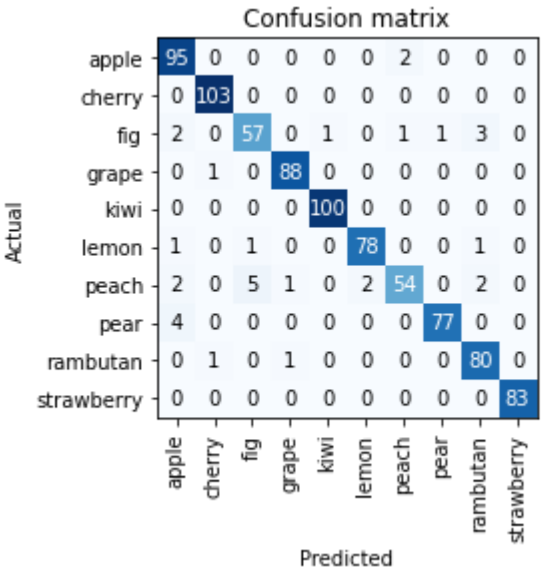
| epoch | train_loss | valid_loss | accuracy | time |
|---|---|---|---|---|
| 0 | 0.340018 | 0.231448 | 0.935065 | 00:12 |
| 1 | 0.252680 | 0.202523 | 0.946871 | 00:12 |
| 2 | 0.212636 | 0.199563 | 0.950413 | 00:12 |
| 3 | 0.163889 | 0.199313 | 0.949233 | 00:12 |
| 4 | 0.134962 | 0.189700 | 0.952775 | 00:12 |
| 5 | 0.111481 | 0.198960 | 0.956316 | 00:12 |
| 6 | 0.095298 | 0.172625 | 0.957497 | 00:12 |
| 7 | 0.076800 | 0.201236 | 0.953955 | 00:12 |

| epoch | train_loss | valid_loss | accuracy | time |
|---|---|---|---|---|
| 8 | 0.066619 | 0.187142 | 0.956316 | 00:12 |
| 9 | 0.064351 | 0.238715 | 0.953955 | 00:12 |
| 10 | 0.058398 | 0.219932 | 0.958678 | 00:12 |
| 11 | 0.045188 | 0.169479 | 0.959858 | 00:12 |
| 12 | 0.032612 | 0.163163 | 0.962220 | 00:12 |
| 13 | 0.031780 | 0.161692 | 0.957497 | 00:12 |
| 14 | 0.026243 | 0.176382 | 0.962220 | 00:12 |
| 15 | 0.018629 | 0.172599 | 0.964581 | 00:12 |
| 16 | 0.016365 | 0.174691 | 0.963400 | 00:12 |
| 17 | 0.017444 | 0.166571 | 0.963400 | 00:12 |
| 18 | 0.013323 | 0.177221 | 0.961039 | 00:12 |

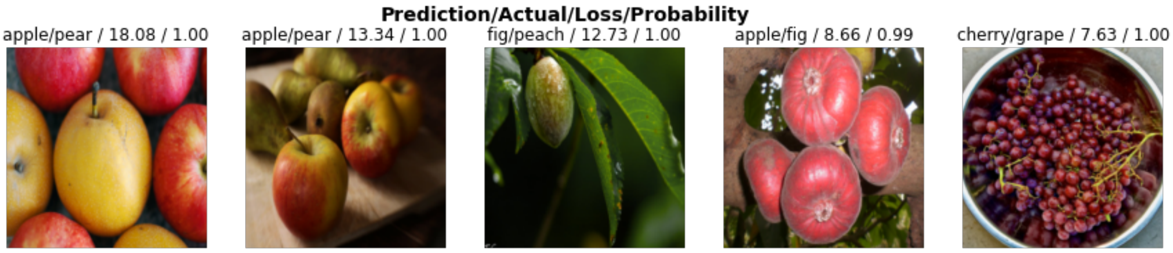
In [60]:
```
learn.recorder.values[-1][1]
```

Out[60]:
```
0.9622195959091187
```

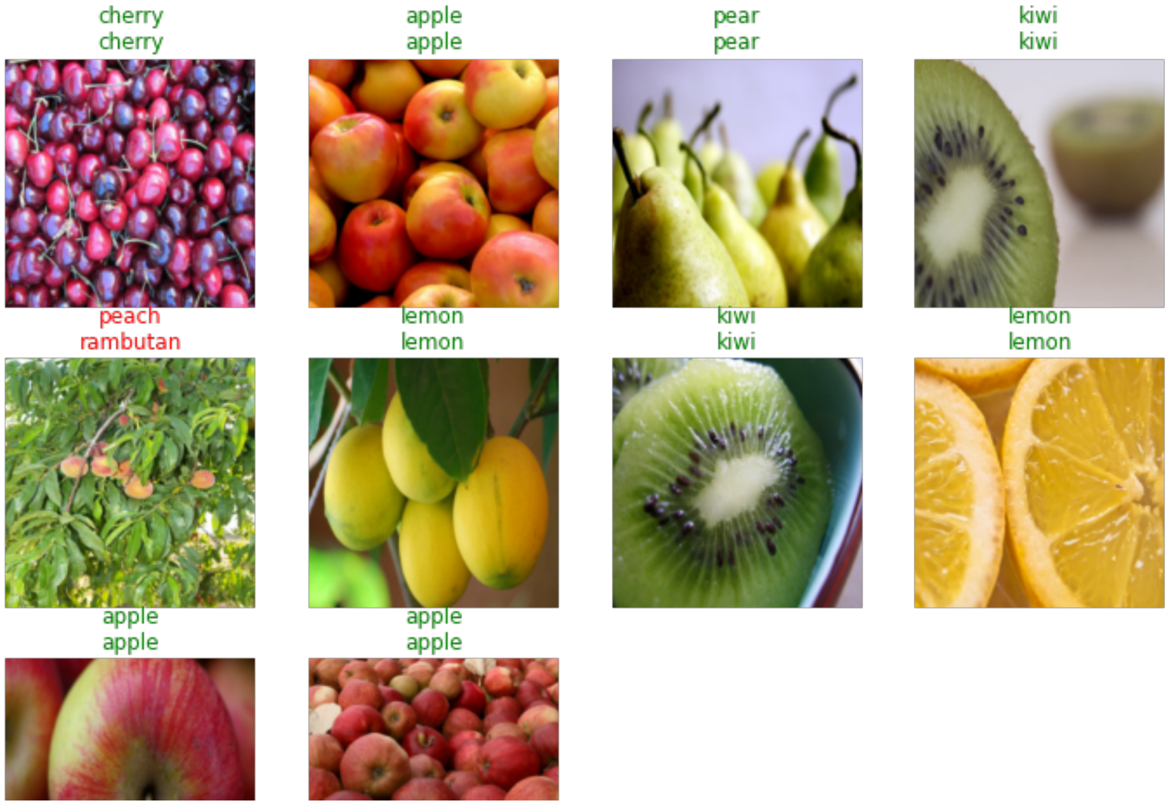
In [59]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
interp.plot_top_losses(5, nrows=1)
```



| Actual \ Predicted | apple | cherry | fig | grape | kiwi | lemon | peach | pear | rambutan | strawberry |
|---|---|---|---|---|---|---|---|---|---|---|
| apple | 95 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| cherry | 0 | 103 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| fig | 2 | 0 | 57 | 0 | 1 | 0 | 1 | 1 | 3 | 0 |
| grape | 0 | 1 | 0 | 88 | 0 | 0 | 0 | 0 | 0 | 0 |
| kiwi | 0 | 0 | 0 | 0 | 100 | 0 | 0 | 0 | 0 | 0 |
| lemon | 1 | 0 | 1 | 0 | 0 | 78 | 0 | 0 | 1 | 0 |
| peach | 2 | 0 | 5 | 1 | 0 | 2 | 54 | 0 | 2 | 0 |
| pear | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 77 | 0 | 0 |
| rambutan | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 80 | 0 |
| strawberry | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 83 |


Prediction/Actual/Loss/Probability
apple/pear / 18.08 / 1.00
apple/pear / 13.34 / 1.00
fig/peach / 12.73 / 1.00
apple/fig / 8.66 / 0.99
cherry/grape / 7.63 / 1.00

In [51]:
```
learn.show_results(max_n=10)
```



## 4. Classify the sample images

Now that your classifier is ready, have a look at the images in the `single_images` folder. Write a Python program that automatically retrieves a list of all the images in this folder and classifies these images. Print a list with in the first column the name of the image file and in the second column the recognized fruit. Check how many fruits were classified correctly. Are you ready to compete in our Colruyt Deep Learning Challenge?

```
In [52]:
# Insert a list of images and classification
# Correct x/20 ?
for i in range(20):
    name = "colruyt_orig/single_images/img" + f'{i + 1:02}' + ".jpg"
    x = name.find("img")
    predict = learn.predict(name)
    print(name[x:] +  "\t\t", predict[0]+ "\t certainty:", predict[2][pred
```

```
img01.jpg                    apple   certainty: tensor(1.0000)
img02.jpg                    cherry  certainty: tensor(1.0000)
img03.jpg                    apple   certainty: tensor(1.0000)
img04.jpg                    cherry  certainty: tensor(1.0000)
img05.jpg                    kiwi    certainty: tensor(1.)
img06.jpg                    peach   certainty: tensor(1.0000)
img07.jpg                    fig     certainty: tensor(1.0000)
img08.jpg                    rambutan        certainty: tensor(1.0000)
img09.jpg                    grape   certainty: tensor(1.0000)
img10.jpg                    lemon   certainty: tensor(1.0000)
```

```
img11.jpg                grape   certainty: tensor(1.)
img12.jpg                strawberry      certainty: tensor(1.0000)
img13.jpg                lemon   certainty: tensor(1.0000)
img14.jpg                strawberry      certainty: tensor(1.)
img15.jpg                kiwi    certainty: tensor(1.)
img16.jpg                pear    certainty: tensor(1.0000)
img17.jpg                pear    certainty: tensor(1.0000)
img18.jpg                peach   certainty: tensor(1.)
img19.jpg                fig     certainty: tensor(1.0000)
```

Print your Jupyter Notebook to pdf and upload it via Canvas.

In [ ]: