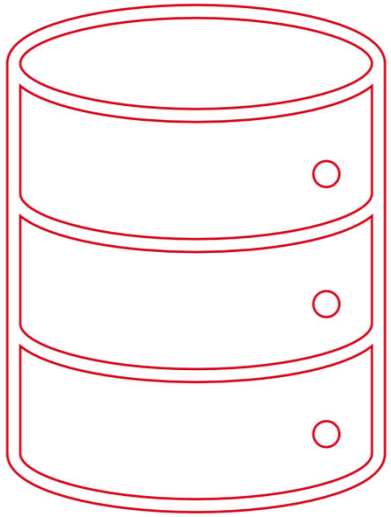
Session 6

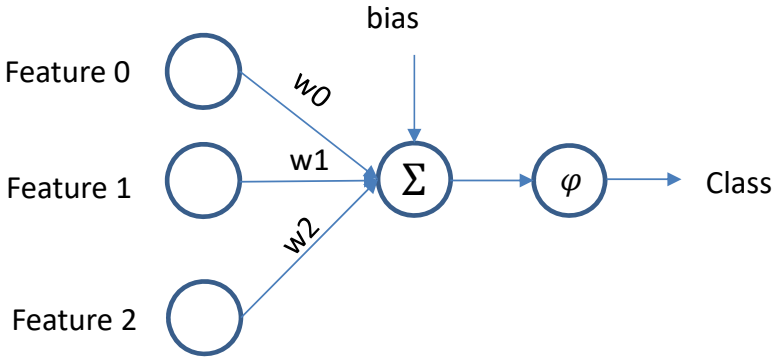
1. Your first steps in Python for Machine Learning
2. The Iris ML problem
3. Exploring the Iris dataset
4. Simple statistical classifier generation in python
5. Neuronal Network SLP/ MLP in Python
6. **Create your own dataset, data collection, and processing**
7. Implementing supervised / unsupervised learning on datasets.
8. Machine learning example and briefing of final project

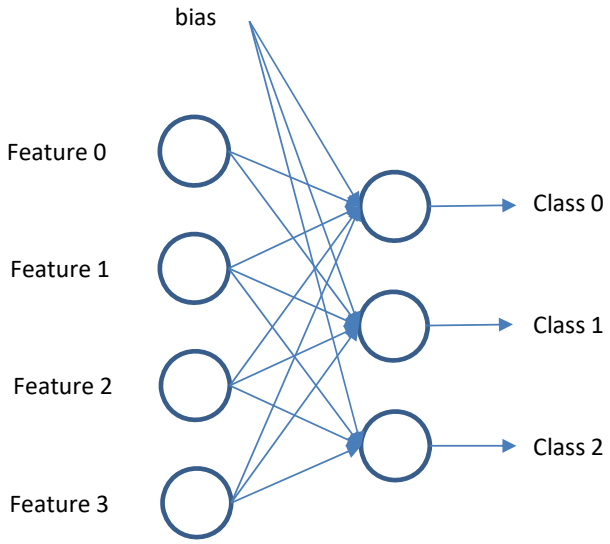


Topics for today

1. Task 5 review
2. Defining Task 6: Extending existing database.
    - Task 6.1: Creation of custom images and extending database.
    - Task 6.2: Exploring and applying Image processing methods for feature extraction.
    - Task 6.3: Preparing the feature extracted images for an ANN.

Session 5 review





Show what you have found out!?

Task 6.1 Solution

"Why reinvent the wheel if you can just make it better"

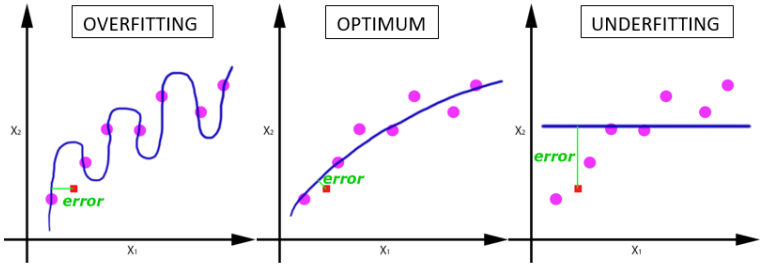
To ensure your understanding of the code, please complete the following questions:

a. In less than **100 words**, explain what the SLP_Iris.py application does.
b. Which activation function is being used?

Review the following piece of code (around line 134).

```
# map the labels to a binary integer value
y = np.where(y == 'Iris-setosa', 1, -1)
```

a. What is the code doing?
b. Why is it needed?
c. What is the learning rate set to for the SLP model?
d. What is epoch and what is it set to?



Task 6.2 Solution

Lets review code/ solution sheet.

Neural Network

Task 6: Create your own dataset, data collection, and processing

Task 6: Creating and improving the Kitchenware dataset

The aim of this task is to familiarize how to:

1. **Create a custom image database**,
2. **Extract features** from these images and
3. **Pre-process** them to apply in an Artificial Neural Network (ANN) for later.

It is important to note, the modelling and training of the ANN will be done in the next seminar (Task 8)
The objects of interest for classifications are **cups, bowls, and plates.**

Task 6.1

1. Take pictures of cups, bowls, and plates
2. Sort the images by using specified naming convention and data structure

Task 6.2

1. Load images and process images
2. Explore and apply different image processing techniques using OpenCV and Matplotlib

Task 6.3

1. Load images and visualize using Keras TensorFlow
2. Pre-process images for ANN
3. Visualize using NumPy, Matplotlib, and Keras TensorFlow

Required libraries for Task 6

```
# Necessary python libraries via import
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv2
```

https://www.tensorflow.org/install/pip

In terminal >> $ python -m pip install tensorflow
Or >> $ python -m pip install --upgrade tensorflow

https://keras.io/

In terminal >> $ pip install keras

https://pypi.org/project/opencv-python/

In terminal >> $ pip install opencv-python

Task 6.1: Extending the image database

1. **Take pictures** of cups, dishes, and plates found at or around your house using for example your mobile phone camera. Arrange the images in the local folders you created.

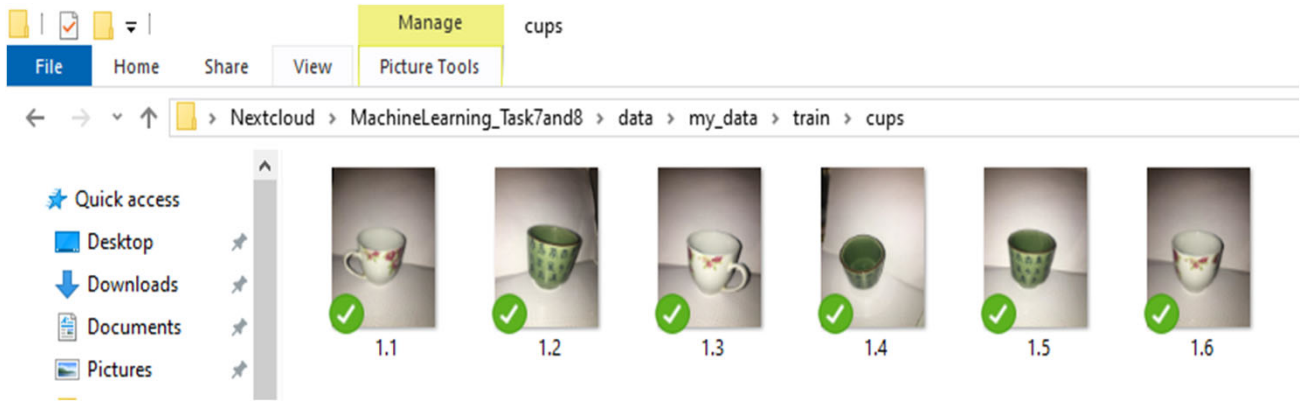


Figure 2: Screenshot of cup image database

Note: Categories have already been defined as cups, dishes, and plates as folder names. The number format must be noted and followed respectively within each of the sub folders.

Expected outcomes for Task 6.1

1. The given database need to be extended with more images of cups, plates and dishes. Make sure all are in .jpg format.
2. For a team of 2 with 3 categories and 4 images for each category [2x3x4 = 24 images more (minimum)].
3. Ensure to take screenshots and document the process in your task reports

Task 6.2: Applying methods for feature extraction

To complete Task 6.2 we need to make use of OpenCV.


```
import cv2 as cv2
```

- OpenCV is a computer vision library
- Provides Application Protocol Interface (API) for image core functions, image processing, etc.

→Imread: responsible to import the image in to the python space

→Canny: responsible to make the edge detection

```
for x in range(3):
    img = cv2.imread('data/my_data/train/cups/1.' + str(x+1) + str('.') + str('jpeg'),0)
    print(img.shape)
    edges = cv2.Canny(img,100,200)
    img = cv2.imwrite('data/my_data/edge_detected/train/cups/1.' + str(x+1) + str('.') + str('jpeg'),edges)
```

```
#How would I change this line of code to be
able to read more data at the same time?
```

Task 6.2 : Making use of Matplotlib

```
from matplotlib import pyplot as plt
```

→ Matplotlib as been renamed as plt

```
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

→ Plt.subplot enables the sub plotting function so that several images can be plotted in the one window. It is important to then refer to the image that we/you would like to plot.

→ Plt.show: will then show the images with a new window popped out

Examples of expected results Task 6.2

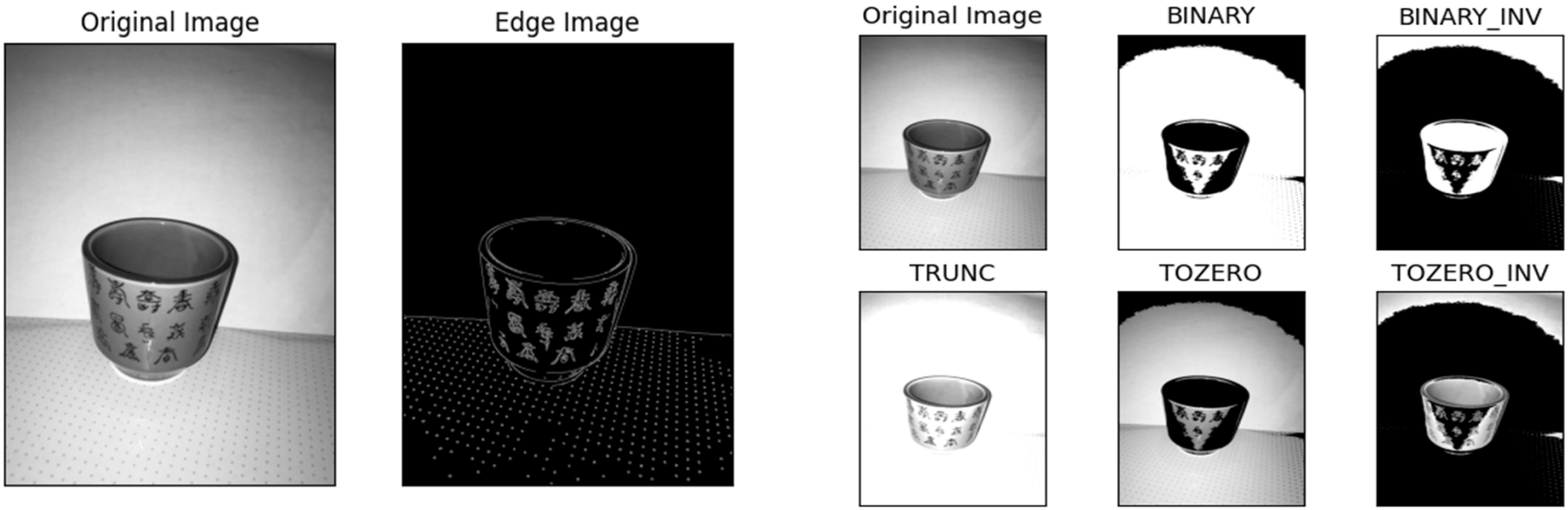


Figure 3: Before and after feature extraction using OpenCV

Examples of expected results Task 6.2

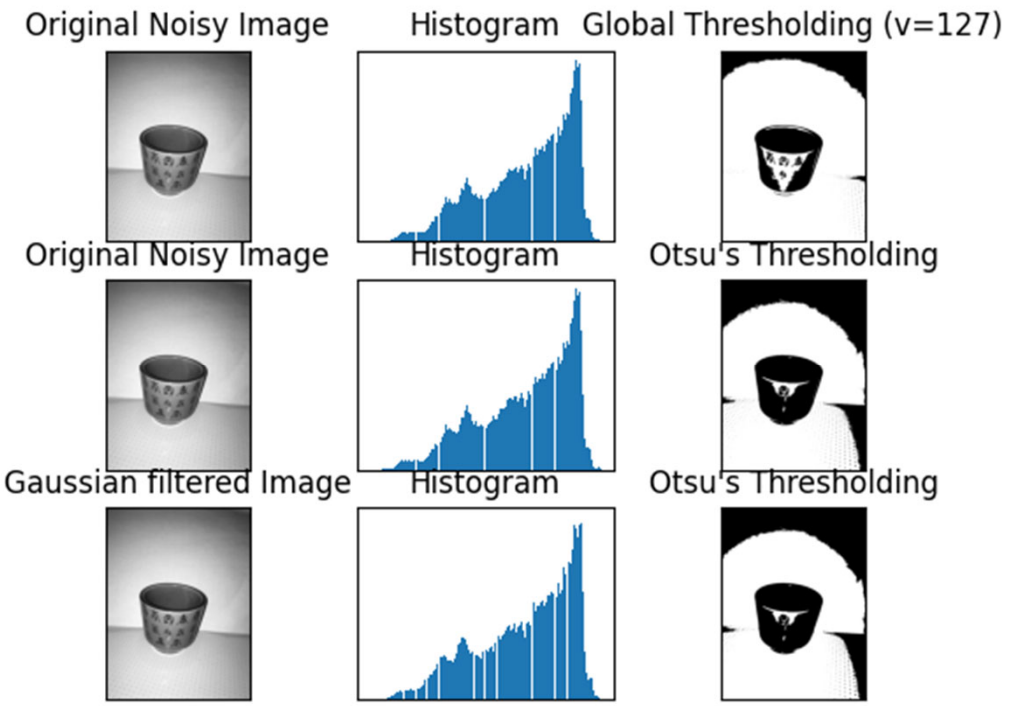


Figure 3: Before and after feature extraction using OpenCV with Histogram

Expected outcomes for Task 6.2

1. Explore at least 3 more image processing methods from OpenCV
   Hint: [Changing color space, geometric transformations, smoothing…etc.]

2. Visualize before and after and include these in your documentation
   [Have you found better feature extractions?]

3. Package (zip) your code and upload

Task 6.3: Preparing the feature extracted images for an ANN.

In other words, Keras TensorFlow import.

- TensorFlow is an open source machine learning platform (Power AI framework) that allows for the creation, pre-processing, modelling, and training of ANN for various application.
- It also comes with a high-end Application protocol Interface (API) called ‘Keras’, which makes the creation and pre-processing of the images easy to implement, with a few lines of codes.

The TensorFlow’ framework and Keras’ has been imported as ‘tf’ and ‘keras’

as seen in the image a below.

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
```

Task 6.3: Pre-processing with Keras TensorFlow.

```
# Desired image resolution
img_height = 128
img_width = 128

# Location of train and test images
data_path_train = 'C:/Users/subas/Nextcloud/MachineLearning_Task7and8/data/my_data/train'
data_path_test = 'C:/Users/subas/Nextcloud/MachineLearning_Task7and8/data/my_data/test'

# Prepare training data from image to tf.data.Dataset
# Check the tutorial: https://www.tensorflow.org/tutorials/load_data/images
# For seed,guarantee the same set of randomness [e.g. initializing weights of ANN, if not set, very different results can arrise]
ds_train = tf.keras.preprocessing.image_dataset_from_directory(
    data_path_train,
    color_mode = 'rgb',
    image_size=(img_height,img_width), # reshape
    shuffle = False,
    seed=123,
)
```

The TensorFlow Keras API function ‘tf.keras.preprocessing.image_dataset_from_directory’, allows us to modify and allocate the data path of the train/test and the path of the created image dataset.

From above:

1. Which ‘Color’ mode is better? [colour/ grayscale?]
2. With Image resolution, height and width is important as higher resolution will cause performance to be slower due the increase in data. What would yield better results? [high or low res]

Task 6.3: Retrieving from variable and plotting image.

```
# Retriving and displaying training images from tf.data.Dataset of training data
plt.figure(figsize=(10, 10))
for images, labels in ds_train.take(-1):
    for i in range(18):
        ax = plt.subplot(5, 4, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
plt.tight_layout()
plt.show()
```

Code of interest:
For images, labels in ds_train.take(-1),
We retrieve all the images and the labels, so that we can use them to plot appropriately

Examples of expected results for Task 6.3

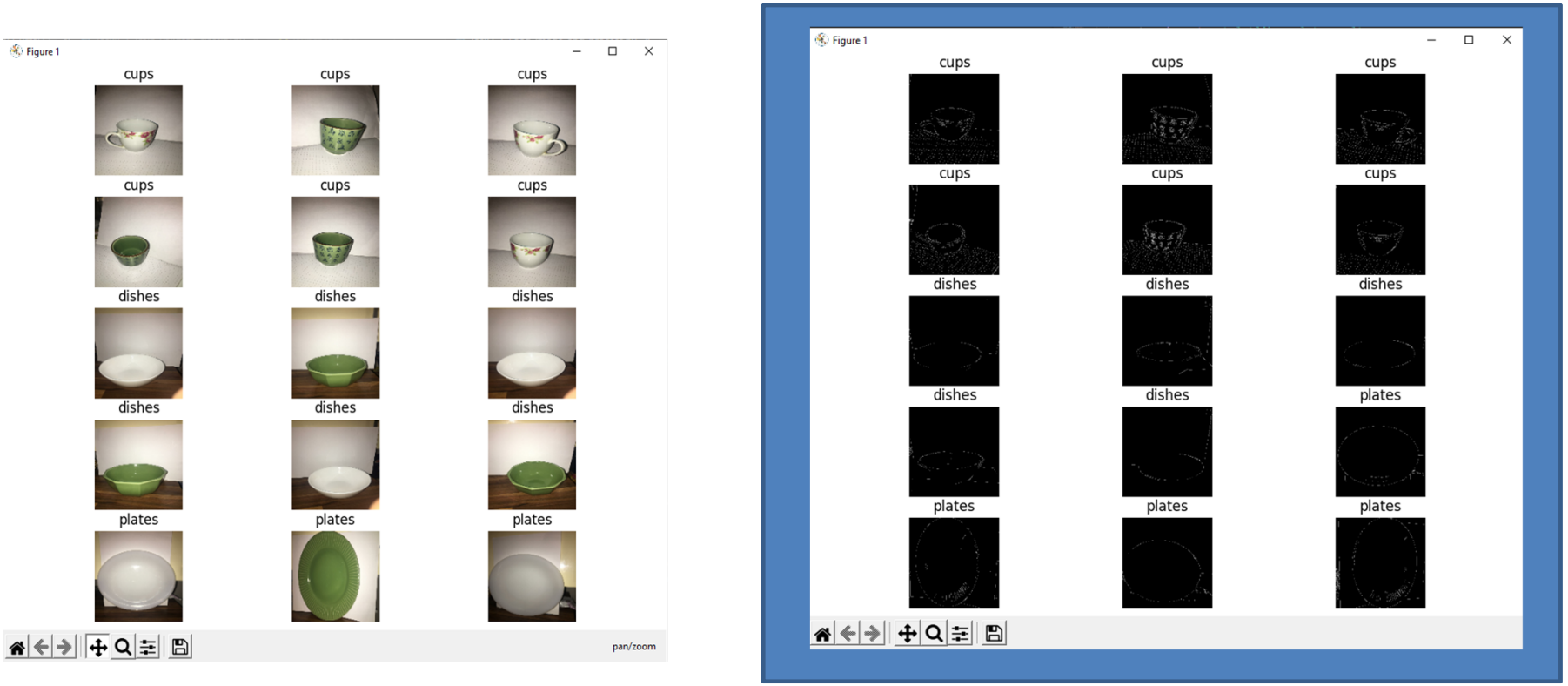


Figure 4: Before and after images using Keras image pre-processing.

Examples of expected results for Task 6.3

```
Found 9 files belonging to 3 classes.

['bowls', 'cups', 'plates']

2021-12-09 17:02:59.635898: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)

['bowls', 'cups', 'plates'
```

Figure 5: Terminal output screenshot of pre-processed images

Expected outcomes for Task 6.3

1. Explore at least 3 or more functionalities with the Keras TensorFlow API for image pre-processing
2. Create before and after visualization.
3. Document [Have you found better methods?]
4. Package (zip) your code and upload

Args	
`directory`	Directory where the data is located. If `labels` is "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
`labels`	Either "inferred" (labels are generated from the directory structure), None (no labels), or a list/tuple of integer labels of the same size as the number of image files found in the directory. Labels should be sorted according to the alphanumeric order of the image file paths (obtained via `os.walk(directory)` in Python).
`label_mode`	• 'int': means that the labels are encoded as integers (e.g. for `sparse_categorical_crossentropy` loss). • 'categorical' means that the labels are encoded as a categorical vector (e.g. for `categorical_crossentropy` loss). • 'binary' means that the labels (there can be only 2) are encoded as `float32` scalars with values 0 or 1 (e.g. for `binary_crossentropy`). • None (no labels).
`class_names`	Only valid if "labels" is "inferred". This is the explict list of class names (must match names of subdirectories). Used to control the order of the classes (otherwise alphanumerical order is used).
`color_mode`	One of "grayscale", "rgb", "rgba". Default: "rgb". Whether the images will be converted to have 1, 3, or 4 channels.
`batch_size`	Size of the batches of data. Default: 32.
`image_size`	Size to resize images to after they are read from disk. Defaults to `(256, 256)`. Since the pipeline processes batches of images that must all have the same size, this must be provided.
`shuffle`	Whether to shuffle the data. Default: True. If set to False, sorts the data in alphanumeric order.
`seed`	Optional random seed for shuffling and transformations.
`validation_split`	Optional float between 0 and 1, fraction of data to reserve for validation.
`subset`	One of "training" or "validation". Only used if `validation_split` is set.
`interpolation`	String, the interpolation method used when resizing images. Defaults to `bilinear`. Supports `bilinear`, `nearest`, `bicubic`, `area`, `lanczos3`, `lanczos5`, `gaussian`, `mitchellcubic`.
`follow_links`	Whether to visits subdirectories pointed to by symlinks. Defaults to False.
`smart_resize`	If True, the resizing function used will be `tf.keras.preprocessing.image.smart_resize`, which preserves the aspect ratio of the original image by using a mixture of resizing and cropping. If False (default), the resizing function is `tf.image.resize`, which does not preserve aspect ratio.

Important to note!

Document your research/ outcomes of each step via your worksheet.

Package (zip) your code and dataset in one file.

Upload and submit both via Moodle

References, extra reading links, and resources

- OpenCV resources
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html
  - https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html
  - https://docs.opencv.org/master/da/d22/tutorial_py_canny.html

- Preprocessing with Keras - https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory