Collect a data-set of images

In this exercise you will capture images from your webcam. We provide you some code to get your started quickly:

1. Clone the imageclassifier repository from here.
2. Try out `imageclassifier`, by following the instructions in the `README.md`.

With the group, collect a data-set of images. You can take pictures of different objects, gestures or even facial expressions. For each category you should have at least > 100 images. Also include a default category `empty` that contains images of your room without any objects. Use Google Drive to share the images with each other.

Organize your data in such a way that you can easily read it in later:

Implement a Feed-Forward-Network

In this exercise, we will implement a Feed-Forward Network with a hidden layer and a sigmoid activation function using only Python and NumPy. It should replicate architecture below.

Build a model in Keras

Building and training a neural network in Keras consists of three steps:

- First, build a **sequential Keras model** by stacking layers on each other.
- Then **compile** the model to create a TensorFlow computation graph.
- Finally, **fit** the model with your training data.

The Keras documentation contains a list of all available layers, activation functions etc. For your first steps, experiment with the **Dense** layer.

Create an image data-set and train a model

Adapt the following code to create an array of image-arrays `X` and an array of targets `y`:

```
import os
from tensorflow.keras.preprocessing.image import load_img

X = []
y = []
classes = [...]
base_path = '...'

for i, target in enumerate(classes):
      files = os.listdir(base_path+target)
      for file in files:
         # load the image
         img = ...
         # convert it to an array
         img_array = ...
         # append the array to X
         ...
         # append the numeric target to y
         ...

X = np.array(X)
y = np.array(y)

# shuffle the data
shuffler = np.random.permutation(len(X))
X = X[shuffler]
y = y[shuffler]
```

Use the custom dataset and pick an architecture that you have seen so far to train a custom image classifier!

Classify images from your webcam with a pre-trained network

Write a function `predict_frame(image)` that takes a [224x224] frame from your webcam and calculates a prediction with a pre-trained network.

Hint: Remember to apply the same pre-processing as was used to train the model. Also, when reading image frames from your webcam, `opencv` uses the **BGR** format whereas `keras` expects an image to be in the **RGB** format (red comes first, green second and blue last).

Use the following code for reversing the channels and preprocessing:

```
def predict_frame(image):
    # reverse color channels
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # reshape image to (1, 224, 224, 3)
    ...

    # apply pre-processing
    image = preprocess_input(image)

    ...
```

Adapt the code of the `imageclassifier` such that whenever you press the `space` key, instead of writing the image to disk, a prediction is printed out on the terminal.