```
# ------------------------------------------------------------------
#
#
# (1) Place in the working directory the following
# auxiliary files:
#
# Simple_MNIST_NN_from_scratch_11.py
# Var__noise1.py
# im1__4__.pgm
# im3__0__.pgm
# im5__5__.pgm
# train.csv
# test.csv
#
# - The training data set (train.csv) can be found at:
# https://app.box.com/s/wg99hpaosqe5rhsyrliguz6c666ksxtk
#
# - The test data set (test.csv) can be found at:
# https://app.box.com/s/fa8qi7t4y5xyerzws0h3j9fh6mv2ikjw
#
#
#
#
# (2) Launch python as:
python -i Var__noise1.py
#
# Or, depending on the installation: python3 -i Var__noise1.py
#
#
# During the execution, some sample images are shown
# if displaying is active, and the correct label and
# prediction are printed.
#
#
# NOTE: in the python interpreter, execute the instructions:
i1 = im1__4__; print(call__make_predictions(i1))
#
# They would normally print the correct result:
# the 4 digit (particularly, '[4]'). If it is not the
# case, please quit and execute again
# python -i Var__noise1
# until print(call__make_predictions(i1)) displays that
# correct result. (The reason is that, for the following
# items, im1__4__ should be correctly classified after
# some randomized initializations, etc. that have been
# performed.)
#
#
# (3) In the python interpreter, experiment with sample
# image "im1__4__.pgm" (which corresponds to digit 4):
#
#
# # Classification without and with noise (standard deviation: 5.0):
in1, _ = call__make_predictions__adding__noise(i1, 5.0)
#
#
#
#
# (4) Increase the standard deviation of the noise
# in increments of 5.0 until the classification fails,
# and save the corresponding noisy image as:
# im1__4__noisy.pgm
#
# # Write the noisy image
cv2.imwrite('im1__4__noisy.pgm', in1.reshape((28, 28)))
#
#
#
#
# (5) For the noisy image obtained in (4), try if
# applying a previous filtering can achieve a successful
```

```
# classification, and save the corresponding filtered
# image as:
# im1__4__filtered.pgm
#
# In order to do that, use the auxiliary function
# 'call__make_predictions__with__filtering'
# as in the following example:
#
if1, _ = call__make_predictions__with__filtering(in1)
cv2.imwrite('im1__4__filtered.pgm', if1.reshape((28, 28)))
#
#
# Note: call__make_predictions__with__filtering has
# an optional argument named 'if_open' (with default
# value False). If 'if_open' is True, only an
# opening is performed; otherwise, a close-open
# filtering is applied.
#
#
#
#
# The outputs of this exercise to be returned and uploaded
# are:
# im1__4__noisy.pgm
# im1__4__filtered.pgm
#
#
#
# ----------------------------------------------------------------
#
#
#
# ################################################
# ################################################
#
#
#
# # Notes for installing pip, numpy, matplotlib, pandas:
#
# python -m pip install -U pip
# python -m pip install -U numpy
# python -m pip install -U matplotlib
#
# python -m pip install -U pandas
#
#
# # https://pypi.org/project/opencv-python/
# # Note: for installing opencv:
# python -m pip install opencv-python
#
#
# # (Depending on the system, perhaps python3 should be
# # used instead of python.)
#
#
#
# ################################################
# ################################################
```