SGN-26006 Advanced Signal Processing Laboratory

Image Recognition Assignment

Background

In this assignment, we familiarize ourselves with modern machine learning; in particular *deep learning*. Image categorization is probably the most studied application example of a deep learning. As an example of designing a deep neural network, we will consider **smile detection** using a deep network and deploy that for real-time execution. For training the net, we use the GENKI-4k Face, Expression and Pose Dataset [1], where the task is to categorize images of faces into two classes based on the facial expression, smile and non-smile. In this task, your goal is to achieve an accuracy (above 85%) with modern tools.

We use a subset of 4000 images of the full MPLab GENKI dataset (2162 smiles; 1838 nonsmile) for which the ground truth label is available. We resize the color image to 64 by 64 pixels (i.e. 64 x 64 x 3). We choose the input size as a power of two, since it allows us to downsample the image up to 6 times using the maxpooling operator with stride 2.

We consider a typical approach of deep neural network design, where we design a network from scratch (with the structure of Figure 1) for binary classification task. The other possibility would be to take a pretrained network and fine-tune that with our data. The drawback of pretrained networks is that their execution time may be high and not so suitable for real time execution. Optionally, you may test the existing pre-trained nets from `keras.applications` module.

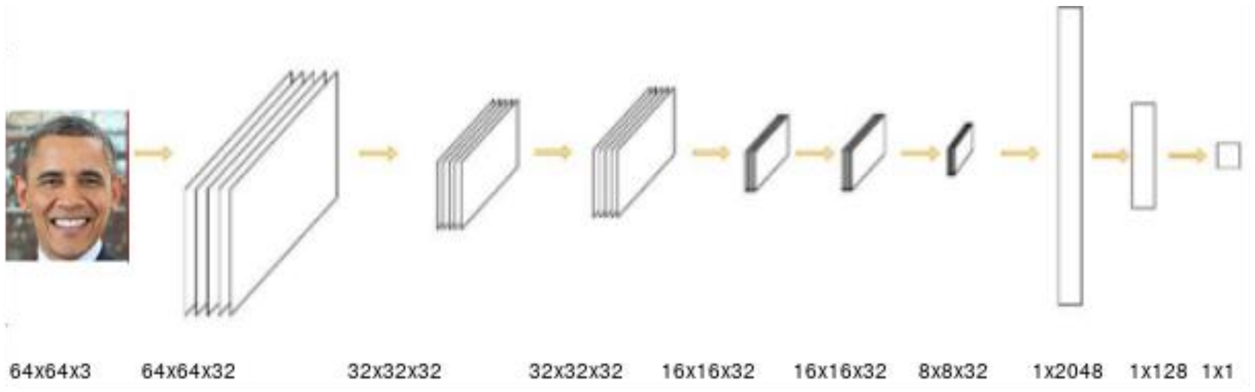

Figure 1. Topology of the small network.

Tasks

Solve the following tasks, prepare a written report describing each item, and return in a zip file along with the code you implemented. It is recommended to use Anaconda environment (Python distribution) with Keras + Tensorflow + OpenCV. This software stack is installed in TC303 classroom, which has powerful GPU's for training as well.

1. Download the GENKI-4K Face dataset from http://mplab.ucsd.edu/wordpress/?page_id=398. Extract the data and parse the smile/non-smile labels such that they can be used for training in Keras. Note that the annotations contain also Head Pose (yaw, pitch and roll parameters), but we will focus on the two-class problem only.

2. Split the dataset into fixed training and testing sets at random (80% / 20%); using, for example, `model_selection.train_test_split` -function from scikit-learn library.

3. Design a convolutional network with the structure of Figure 1. Train the network for 50 epochs, store the classification accuracies, save trained model and add a plot of accuracies to your report.

4. Implement another script that detects smile in real-time from webcam (or mp4 video file if you do not have a webcam).

    a. Grab frames in an infinite loop using the VideoCapture class of OpenCV library[1]. The system works with webcam and video files.

    b. After each grab, send the frame to `model.predict()` in Keras. Overlay the result on the frame using `cv2.putText()`, simply as text ("smile" or "no smile"). Note that the OpenCV grabs frames in Blue-Green-Red (BRG) channel order, while you may have trained the net for RGB images. OpenCV has functions for color channel conversion.

    c. Show the result on the screen using `cv2.imshow()` and `cv2.waitKey()` combination.

Add an explanation of your implementation and a confusion matrix of accuracies of the recognition network to the report. Also add a few screen shots to the report.

Instructor

The instructor of this assignment is Bishwo Adhikari (bishwo.adhikari@tut.fi). Return your report and code to Moodle by the date given at the course Moodle page.

References

[1] `http://mplab.ucsd.edu`, *The MPLab GENKI Database, GENKI-4K Subset.*

[1] OpenCV Python interface can be installed within anaconda simply by "pip install opencv-python" or "conda install opencv".