COMP.SGN.240 Advanced Signal Processing Laboratory
Image Recognition

Background

In this assignment, we familiarize ourselves with modern machine learning, in particular deep learning methods. Classification is probably the most studied application example and a fundamental block for many complex deep learning systems. So, we are considering building a **smile detector** using a deep network and deploying that for real-time execution. For training the network, 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 of at least 85%.

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 64x64 pixels, which will allow us to down-sample 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 pre-trained network and fine-tune that with our data. The drawback of pre-trained 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.

Tasks

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

1. Download the GENKI-4K Face dataset from https://inc.ucsd.edu/mplab/398
2. 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.
3. 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.

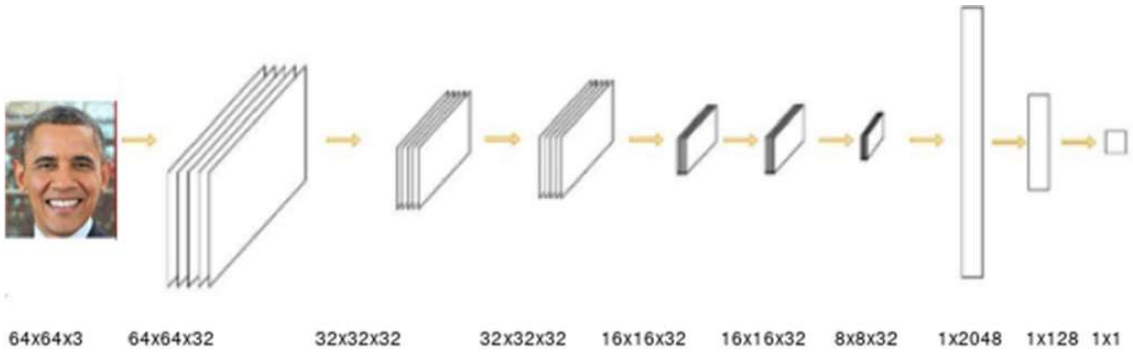

Figure 1: Topology of the small example network.

4. Design a convolutional network with the structure of Figure 1. Train the network for 50 epochs, store the classification accuracies, save the trained model, and add a plot of accuracies to your report. You can change the size of networks by adding layers. **Using pretrained network is fine. However, you need to design a multilayer network as in Figure 1 and report its performance first.**

5. 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 (RGB) 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 of detection result to the report.

Google colab [2] is a good alternative for those who need GPU for network training.
Alternatively, you can request the university GPU cluster "Narvi" [3].

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

FAQs

1. **Why is my model accuracy too low?** You could try following techniques to improve your network performance:
   (a) Deeper network (increase the network size by adding layers)
   (b) Hyper parameter tuning (try different optimizers, loss functions)
   (c) Data augmentation (increase the amount of training samples)
   (d) Transfer learning (use pretrained network and fine-tune with your data)
2. **Can I submit Jupyter notebook?** Yes, you can submit the well commented notebook. Remember that the requirements are the project code and report. You cannot replace a report with a notebook.
3. **Do I need to complete all tasks if I work alone?** Yes, no matter how you work, you need to complete all tasks to pass the assignment.
4. **More questions?** Send your questions to afshin.dini@tuni.fi with the subject "COMP.SGN.240 Project …"

Submission

Return your report and commented source code as a single zip file to Moodle. Your submission file should be named as "student1_student2_image_recognition.zip". Note that late submissions will not be evaluated.

Instructor

The instructor of this assignment is Afshin Dini (afshin.dini@tuni.fi).

References

[1] http://mplab.ucsd.edu, The MPLab GENKI Database, GENKI-4K Subset.
[2] https://colab.research.google.com/notebooks/intro.ipynb, Introduction to Colab.
[3] https://tuni-itc.github.io/wiki/Technical-Notes/tuni-narvi-cluster/