INSTALLATION GUIDE

A. Overview

This guide shows how to run the code used to produce the images in our paper:
http://www.evolvingai.org/fooling

B. Requirements

Software:

This is an installation process that requires two main software packages (included in this package):

- Caffe: http://caffe.berkeleyvision.org
  - Our libraries installed to work with Caffe
    - Cuda 6.0
    - Boost 1.52
- Sferes: https://github.com/jbmouret/sferes2
  - Our libraries installed to work with Caffe
    - OpenCV 2.4.10
    - Boost 1.52

Computing Environment:

- An MNIST experiment (Fig. 4, 5 in the paper) can be run directly on a local machine (4-core) within a reasonable amount of time (around 5 minutes or less for 200 generations)
- An ImageNet experiment needs to be run on a cluster environment. It took us 4 days x 128 cores to run 5000 generations and produce 1000 images (Fig. 8 in the paper).

C. Installation

1. Install Caffe and its required libraries.
   - The specific version provided is different from the Caffe master branch and it has the modification that enables feeding OpenCV data from memory to a Caffe model for evaluation via ImageDataLayer.
   - Caffe installation guide: http://caffe.berkeleyvision.org/installation.html
2. Find a model you want to “fool” or train one yourself. We used the BVLC CaffeNet model provided by Caffe.
   - See the list of models here.
   - Make sure that it is running and gives you validation / test errors as expected.
3. Install Sferes and its required libraries:
   - The specific Sferes version provided has all the modules needed to run the experiment and is very similar to (but not the same as) the Sferes master branch.
   - Sferes installation guide: https://github.com/jbmouret/sferes2/wiki/Compilation
4. Linking Caffe and Sferes:
   - Our code already does the linking between Caffe and Sferes:

        - Sending an image in memory (OpenCV::Mat) to Caffe for evaluation
        - Get the output confidence scores from the Caffe model for that image
        - See ./sferes/exp/images/fit/fit_map_deep_learning.hpp
    - Install the Caffe libraries
        - Install the Caffe library (caffe.so and caffe.a) and its headers so that Sferes can find it for compilation.
            - You can refer to ./sferes/install_caffe.sh to see how we install it on our system.
        - Modify Sferes waf configuration file according to your system
            - In file: ./sferes/exp/images/wscript
            - Make sure that the path in obj.includes = points to the correct locations
5. Compile a Sferes experiment:
    - Our Sferes experiments for the paper are located in the ./sferes/exp/images folder
    - Update the experiment config to compile an experiment:
        - In file: ./sferes/exp/images/wscript
        - Modify the field obj.source = to a file name in the experiment folder
    - Make sure that the code compiles correctly with waf:
        - waf -- configure …
            - see ./sferes/build.sh for example parameters we used on our system
        - waf build
        - waf -- exp images
        - This is the result of our working waf -- configure for your reference

```
WARNING simplejson not found some function may not work
module : [modules/nn2]
module : [modules/map_elite]
Check for program g++ or c++            : /usr/bin/g++
Check for program cpp                   : /usr/bin/cpp
Check for program ar                    : /usr/bin/ar
Check for program ranlib                : /usr/bin/ranlib
Checking for g++                        : ok
boost headers                           : Version 1_52 (/home/anh/src/sferes/include)
library boost_serialization             : ok
library boost_filesystem                : ok
library boost_system                    : ok
library boost_unit_test_framework       : ok
library boost_program_options           : ok
library boost_graph                     : ok
library boost_mpi                       : ok
library boost_python                    : ok
library boost_thread                    : ok
Checking for header tbb/parallel_for.h  : not found
```

```
Checking for header tbb/parallel_for.h   : not found
Checking for header mpi.h                : ok /usr/include/mpi
Checking for SDL (optional)              : not found
Checking for header Eigen/Core           : ok /home/anh/src/sferes/include
Check for program ode-config             : not found
Checking for ODE (optional)              : not found
Checking for GSL (optional)              : not found
```

6. List of experiment to reproduce :
    - Direct Encoding (section 3.1 and 3.3 in the paper)
        - MNIST: dl_map_elites_images_mnist_direct_encoding.cpp
        - ImageNet: dl_map_elites_images_imagenet_direct_encoding.cpp
    - Indirect Encoding (section 3.2 and 3.4 in the paper)
        - MNIST: dl_map_elites_images_mnist.cpp
        - ImageNet: dl_map_elites_images.cpp
7. Configure the experiment in the cpp file:
    - After selecting an experiment and compiling successfully, now configure it as you want.
    - File ./sferes/exp/images/dl_images.hpp contains generic parameters which are by default in inherited (and can be overwritten) by the experiments:
        - Params::image::size : size of the images to evolve (default to 256)
        - Params::image::crop_size : crop size according to Caffe config (default to 227)
        - Params::image::use_crops : use crop or not.
            - True: Use 10 crops
            - False: Use 1 center crop
        - Params::image::color : True/False for MNIST grayscale or ImageNet color images.
        - Params::image::num_categories : default to 1000 corresponding to 1000 ImageNet categories. Change to 10 for MNIST.
            - Note: in the experiment cpp file also update Params::ea::res_y to be equal to the num_categories unless you know MAP-Elites really well to experiment differently.
    - In the experiment cpp file (e.g. dl_map_elites_images.cpp):
        - Params::pop::nb_gen : the number of generations to run.
        - Params::pop::size : the number of population
        - Params::image::model_definition : path to the prototxt file used by your model
            - See the example prototxt file
        - Params::image::pretrained_model : path to your model
        - Params::pop::dump_period : how frequently your images will be printed out
        - To run on cluster with MPI, uncomment this line
            - typedef eval::MpiParallel<Params> eval_t;
        - To run locally on a single core, uncomment this line, and comment out the above

        - typedef eval::Eval<Params> eval_t;
8. Test running your experiment:
    - The binary will be located in the directories:
        - ./sferes/default/exp/images
        - ./sferes/debug/exp/images
    - If running locally, you can test running this binary directly and the result will be written in a newly created directory: "**mmm**" (i.e. ./sferes/mmm/)
        - The fitness or confidence score of the champions in each generation will be written in the archive_x.dat (e.g., for generation **235**, see ./sferes/mmm/archive_235.dat).
            - The first column is the category index
            - The second column is the fitness
        - Images are stored in ./sferes/mmm/map_gen_x (e.g., for generation **236**, see ./sferes/mmm/map_gen_236).
9. To run on a cluster
    - We use 128 cores usually to run ImageNet experiments.
    - If you need help configuring, consult the file : ./sferes/submit_jobs.py. This is how we schedule jobs on a cluster environment.

10. Caffe version
    - Our particular Caffe version requires a dummy source parameter in the prototxt in order to initialize the input data blobs at the IMAGE_DATA layer properly.
        - Please refer to the ./model directory on how we configure the prototxt
    - The batch size in the prototxt file is default to 10 as we use 10 crops per evaluation.

For specific Sferes or Caffe questions, please see their github pages or Google groups.
For other questions regarding these experiments from our paper, please don't hesitate to ask. :)

Cheers,