dcgan.torch User Documentation

This user documentation is designed to assist users in installing and training their own DCGAN model. The guide is designed around the DCGAN implementation proposed by Soumith Chintala, which is available through GitHub. The instructions detail the hardware, software requirements pertinent to the installation process and a simple tutorial to begin training and generating your own AI images.

The executable terminal commands in the guide will be highlighted:
Example: `ping 8.8.8.8`

Table of Contents

SOFTWARE REQUIREMENTS **3**

HARDWARE REQUIREMENTS **3**

TORCH INSTALLATION **4**

- Without GPU 4
- With NVIDIA GPU 4
- Display UI - Optional 5

DCGAN.TORCH **6**

- Import DCGAN 6
- Generating AI images 6

TRAINING YOUR OWN DCGAN **7**

- Automated Image Collection 7
- Manual Image Collection 7
- Pre-processing Images 7
- Execution 8

ADDITIONAL INFORMATION **9**

- main.lua 9
- Checkpoints 9
- Resume from checkpoint 9
- Missing File or Directory error 10

Software Requirements
A list of OS, software and version information used in this documentation, there are recommended and tested for compatibility.

Recommended:
- Ubuntu 18.04
- CUDA Toolkit 10.2
- cuDNN v7.6.5 for CUDA 10.2

Optional:
- Atom 1.44.0
- Affinity Designer 1.8.1

Hardware Requirements
A list of hardware components used in this project:

Recommended:
- SSD (100GB+)

Optional:
- Portable Hard Drive (backup neural network checkpoints)
- NVDIA GPU

Torch Installation

Torch is an essential computing framework providing support for machine learning algorithms.
Read more about Torch: http://torch.ch/

Without GPU

1. **Install Torch & Luarocks** (link)[1]

   # Installing Torch to a local folder
   git clone https://github.com/torch/distro.git ~/torch --recursive
   cd ~/torch; bash install-deps;
   ./install.sh

   Choose an option based on current shell

   - # On Linux with bash
     source ~/.bashrc

   git clone https://github.com/torch/distro.git ~/torch --recursive
   cd ~/torch
   TORCH_LUA_VERSION=LUA52 ./install.sh
   $ luarocks install image
   $ luarocks list

With NVIDIA GPU

1. **Install CUDA ToolKit 10.2** (link)[2]
   Execute the following commands sequentially, in a terminal:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin

$ sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600

$ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub

$ sudo add-apt-repository "deb http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/ /"

$sudo apt-get update

$ sudo apt-get -y install cuda
```

2. **Install Torch & Luarocks**
   Refer to previous section

3. **Restart Computer**

[1] http://torch.ch/docs/getting-started.html
[2] https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork

4. **Install cuDNN v7.6.5 for CUDA 10.2** (link)[3]

   Select the link to visit the cuDNN download webpage
   Create an account to access the download page
   Download cuDNN v7.6.5 (November 18th, 2019), for CUDA 10.2

   Download the following files:
   - cuDNN Library for Linux (cudnn-10.2-linux-x64-v7.6.5.32.tgz)
   - cuDNN Runtime Library for Ubuntu18.04(Deb)
     - (libcudnn7_7.6.5.32-1+cuda10.2_amd64.deb)
   - cuDNN Developer Library for Ubuntu18.04(Deb)
     - (libcudnn7-dev_7.6.5.32-1+cuda10.2_amd64.deb)

   Installing cuDNN Library:

   ```
   tar -xvf cudnn-10.2-linux-x64-v7.6.5.32.tgz
   sudo cp cuda/include/*.h /usr/local/cuda/include
   sudo cp cuda/lib64/*.so* /usr/local/cuda/lib64
   ```

   Installing cuDNN Runtime & Developer Library:

   ```
   sudo dpkg -i libcudnn7_7.6.5.32-1+cuda10.2_amd64.deb
   sudo dpkg -i libcudnn7-dev_7.6.5.32-1+cuda10.2_amd64.deb
   ```

Display UI - Optional

There is a display package available which displays training process from beginning to end.

```
luarocks install https://raw.githubusercontent.com/szym/display/master/display-scm-0.rockspec
```

- Start the server with: `th -ldisplay.start`
- Open this URL in your browser: http://localhost:8000

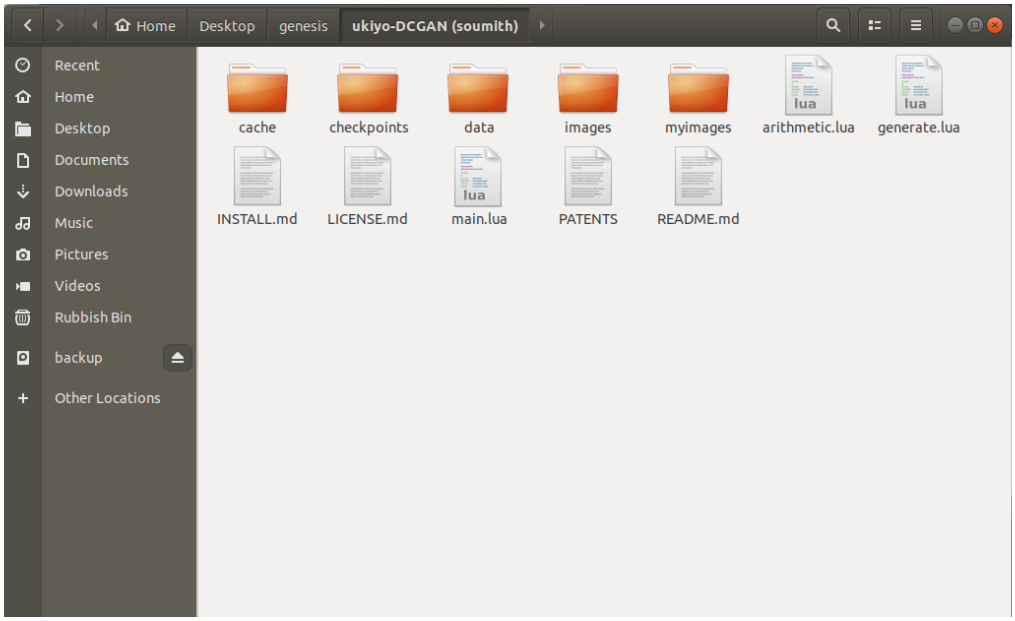
[3] https://developer.nvidia.com/rdp/cudnn-download

dcgan.torch

There are several trained models and datasets provided in Chinatala and Barrat's GitHub repositories. The trained model can be used to become familiarize with different parameters and to intuitively understand the training and generating scripts. (link[4])

Import DCGAN

execute the following command in a terminal, to clone DCGAN.torch to a local folder
git clone https://github.com/soumith/dcgan.torch.git



Generating AI images

Weights for beginners: https://github.com/robbiebarrat/art-DCGAN

Download a generator weight:

- Landscape: Generator Link
- Nude-Portrait: Generator Link
- Portrait: Generator Link

Save the generator in 'DCGAN.torch'

Open a terminal from DCGAN.torch and execute the following command to generate images from your chosen GAN

Using the portrait generator, linked above.
net= portrait_584_net_G_cpu.t7 th generate.lua

Optional Display UI:

Refer to Display UI section

[4] https://github.com/soumith/dcgan.torch

Training your own DCGAN

This section will guide you through the steps necessary to train your own DCGAN and produce new AI art.

First step involves gathering data to feed the neural network, and the collected data should resemble the genre/ theme/ artistic style of the expected outcome. the training data collated will focus on the Japanese portrait prints in the style of Ukiyo-e. These prints are manually sourced from a large repository, Ukiyo-e Search[5].

Automated Image Collection

There are automated processes which can help source large amount of training data. Automated data collection can yield lots of results, however inconsistent data and anomalies can affect the accuracy of the training.

An image gathering tool provided by Robbie Barrat, allowing the user to 'scrape' a genre of art data from WikiArt.org[6]

```
# replace following parameters for the desired outcomes
python3 genre-scraper.py - -genre *genre* - -num_pages 10
```

genre parameter must come from genre's listed on this page[7] for the scraper to work
num_pages parameter is set to 10 as it seems to fix a time-out issue

Manual Image Collection

The sets of training data used in this tutorial can found at ukiyo-DCGAN[8], they are manually collected based on personal preference, perception of patterns and similarities.

The images are collected from an aesthetics point of view and homogeneity:

original-ukiyo-e.zip	a collection of unfiltered or processed image data
cropped-ukiyo-e.zip	a collection of processed images and categorized based on portrait orientations

Pre-processing Images

The image data was processed using an image editing software (Adobe Photoshop/ Affinity Designer)

[5] https://ukiyo-e.org/
[6] https://www.wikiart.org/
[7] https://www.wikiart.org/en/paintings-by-genre/
[8] https://github.com/liknhe/ukiyo-DCGAN

Selected images are filtered from original images and processed; the processing focused on several key attributes:

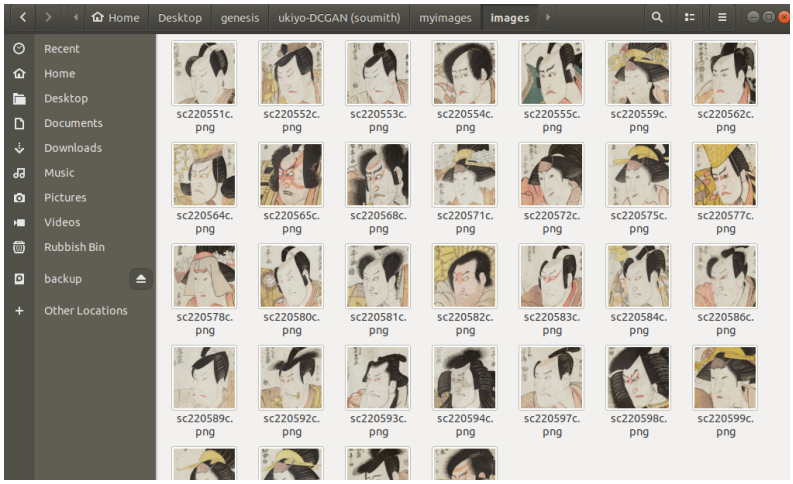
- Facial orientation
- Location of features (eyes/ nose/ mouth)
- Photo Dimension

Pre-processing includes cropping and inverting images to create a set of homogenous data.

Original	Unprocessed
	

Execution

Place the collected images within folder (e.g. ‘portrait_prints’)



Execute the following command in a terminal, within DCGAN.torch:

```
DATA_ROOT=portrait_prints dataset=folder ndf=50 ngf=150 th main.lua
```

The default epoch: 25 (Refer to ‘main.lua’ to update training epochs)
Trained checkpoints can be located in the ‘checkpoints’ folder

Resulting models:

base_100_net_G.t7 (Generator)
base_100_net_D.t7 (Discriminator)

Choose the generator file and, indicated by net_G
Execute the following command to generate an AI print:

```
net=base_100_net_G.t7 th generate.lua
```

Additional Information

main.lua

This script is used to the training neural networks from the list of specified variables.
Attributes of interest to explore:

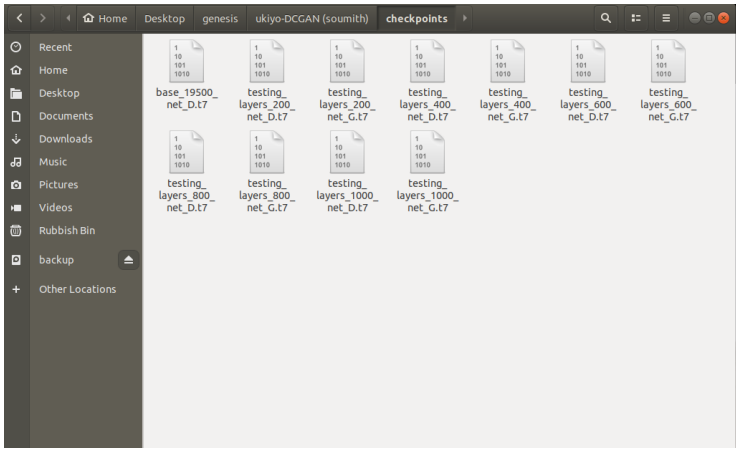
batchSize=64 -- number of images to process in 1 epoch
name=experiment1 -- change name to avoid overwrite checkpoints with existing names

Checkpoints

Checkpoints are taken at intervals during a network training, a snapshot of the generator and discriminator model at the epoch.

Name (main.lua: line 21)
This attribute should be updated before a new training to avoid overwriting existing models

Interval (main.lua: line 23)
This attribute indicates the epoch intervals between checkpoints



At each checkpoint interval, 2 files are generated and saved in checkpoints:

- neural_net_100_net_D.t7
- neural_net_100_net_G.t7
- neural_net_200_net_D.t7
- neural_net_200_net_G.t7

Resume from checkpoint

To resume from checkpoints, place the checkpoint in DCGAN.torch:
Execute the following command:
DATA_ROOT=portrait_prints dataset=folder netD=checkpoints/ base_100_net_D.t7netG= base_100_net_G.t7th main.lua

*Bug: the display UI does not seem to work when resuming from checkpoints

Missing File or Directory error

An error message will appear when new dataset is used.

> ‘Missing file or directory’

Delete the cache file in the cache folder, to remove the error and continue training.