Université de Fribourg

Very Deep Learning

DEEPDRAW

Ismaïl SENHAJI, Gil CLAVIEN
Noé ZUFFEREY, Michaël DIATTA
Igor DUNDIC

Responsable : Liwicky LIWICKY

20 décembre 2017

Département d'Informatique — Departement für Informatik
Université de Fribourg — Universität Freiburg
Boulevard de Pérolles 90, 1700 Fribourg, Suisse

1 INTRODUCTION

For the very deep learning project, we decided to build, train and manage a neural network that can draw pictures of simple concept. We use the Quick Draw Dataset, a google's game, that provide a huge dataset of labelised simple drawing.

2 ABOUT QUICKDRAW-DATASET

The Quick Draw Dataset [1] is a collection of 50 million drawings across 345 categories, contributed by players of the game Quick, Draw ! [2]. The drawings were captured as timestamped vectors, tagged with metadata including what the player was asked to draw and in which country the player was located.

The Quick Draw game principle is to draw specific picture given by the computer. This one try to recognize, with machine learning, your drawing.

FIGURE 1 – Quick Draw example.

The Quick Draw Dataset provides preprocessed dataset. We choose to use the **Simplified Drawing files** preprocessed dataset. They've simplified the vectors, removed the timing information, and positioned and scaled the data into a 256x256 region. The data is exported in ndjson format with the same metadata as the raw format. We use a binary version of this format, for efficiency purpose.

3 DEEPDRAW CODE

The final goal of this project is to reverse the principle given in the previous section : with give a class (a number) to the computer and this one try to draw the corresponding drawing.

4 CODE

Présenter les choix, les NN et tout ce merdier

1. https://github.com/googlecreativelab/quickdraw-dataset
2. https://quickdraw.withgoogle.com/#

4.1 Get the data

The size of the dataset presented an interesting challenge : at 7.1GB in the packed binary format, and even more once parsed into Python data types, it barely fits into memory. To solve this, we build an index file containing each drawing's id, class (and by extension the file where it can be found), and its offset within the file. The size of the resulting `index.csv` file is 1.6GB, small enough to be loaded into memory in a reasonable time.

Drawings are provided as a sequence of strokes, where each stroke is a sequence of (x, y) coordinates. This format isn't particularly suited for a Neural Network, thus the need to transform them. We implemented a `Rasterizer` class, responsible for converting such a sequence of strokes to a 256 by 256 rasterized image, using Bresenham's line algorithm [3]. This class is compatible with Torch's transormation framework and can be used in conjunction with other transformations, such as resizing or normalizing the image.

Another transormation class was implemented, `Sequencer`, which converts the drawing into a sequence of state vectors, for use in a Recurrent Neural Network [4]. This approach wasn't used, however.

4.2 First NN : GAN

Générateur...

4.3 Second NN : DCGAN

Discriminative...

5 PROBLEMS

Les problèmes qu'on a eu, outre les problèmes de natel de Mick...

6 RÉSULTAT

On présente les résultats et on nique des mères... histoire de fêter ça...

7 FURTHER WORK

Warsserstein

WaffenSS

3. https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm
4. https://arxiv.org/abs/1704.03477

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

RÉFÉRENCES