Using Deep Learning To Implement Domain Generation Algorithm

Student: Chia-Ruei Lee

Advisor: Chia-Mu Yu

CONTENTS

3

Botnet

Introduction

4

Domain Generation Algorithm

Introduction

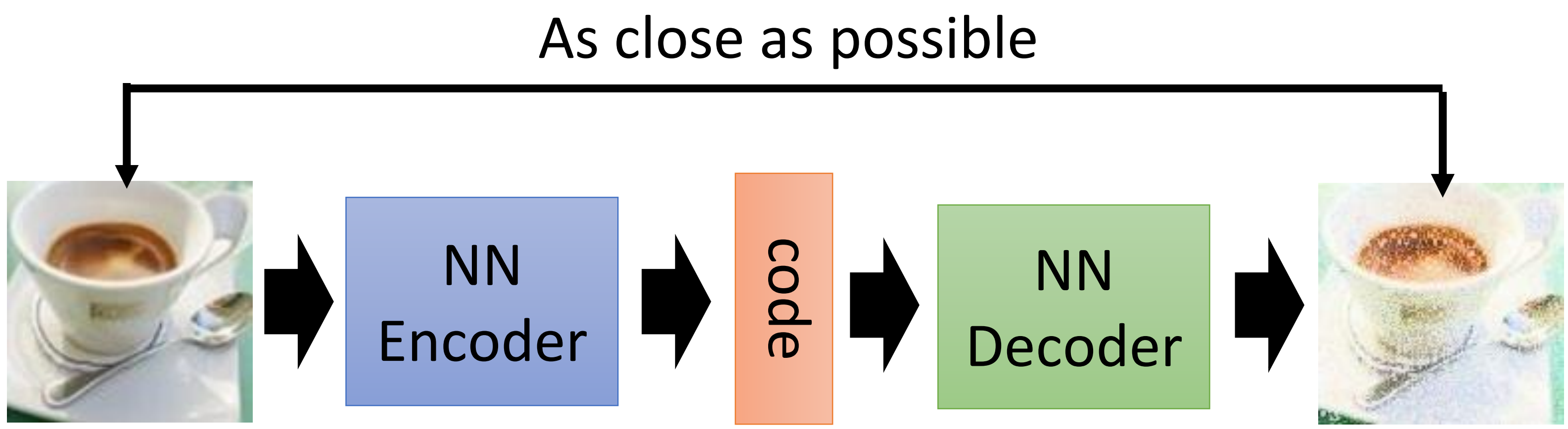
5 Neural Network

Deep Learning

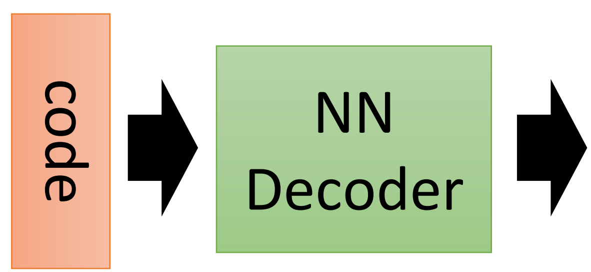
6 Auto-Encoder

Deep Learning

AUTO-ENCODER

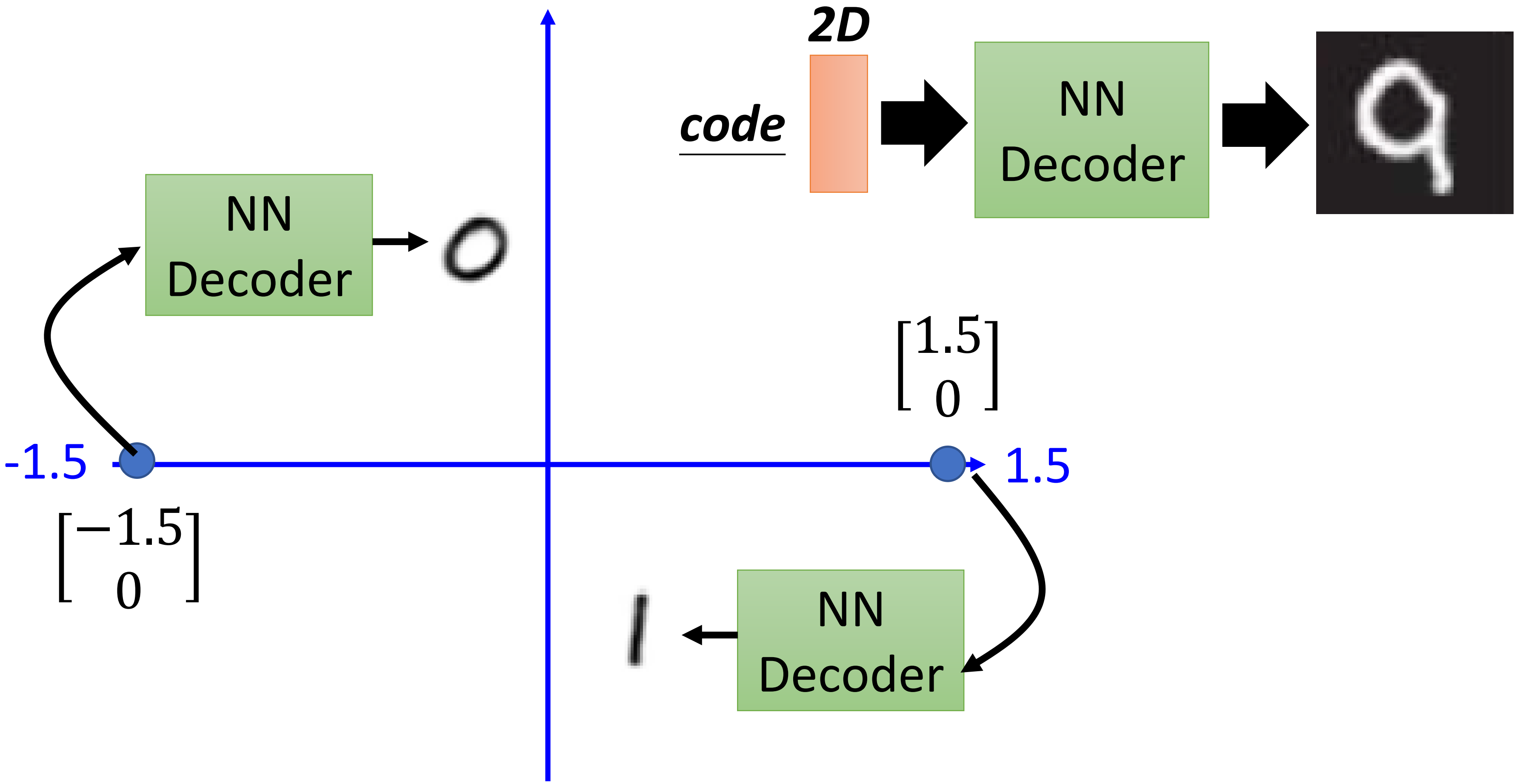


Randomly generate
a vector as code



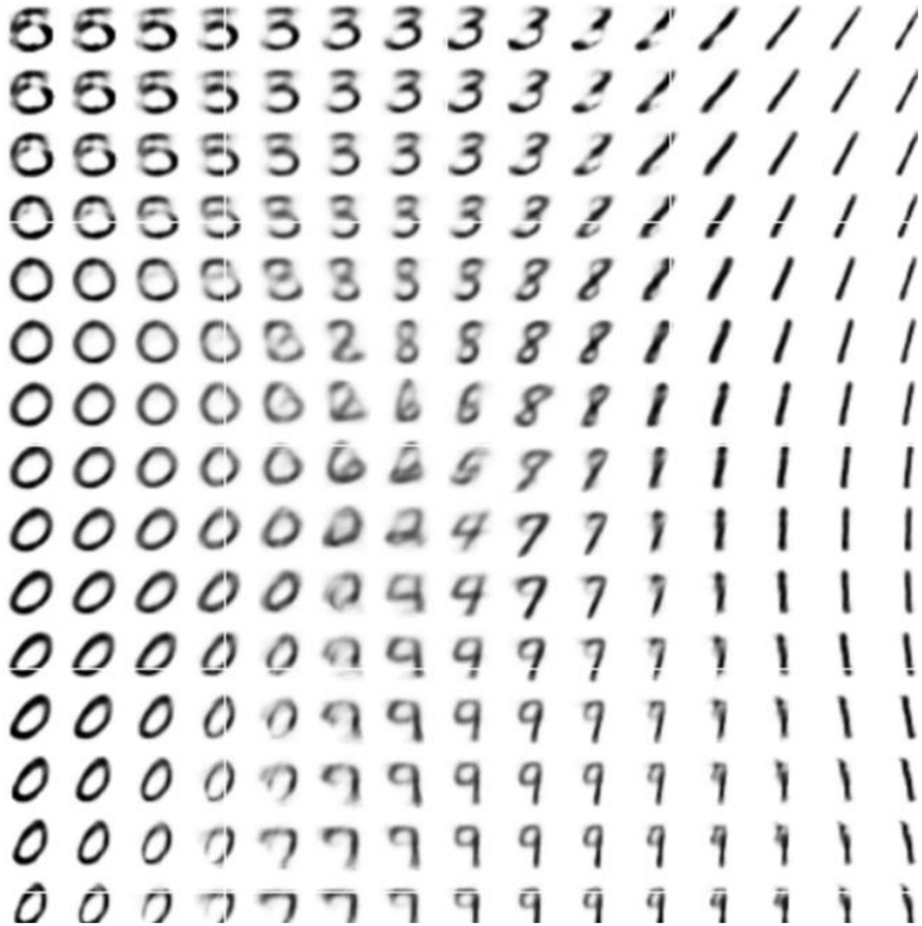
From Hung-yi Lee, "Generative Adversarial Network/Basic Idea/Machine Learning and having it deep and structured (2017,Spring)"
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html

AUTO-ENCODER



From Hung-yi Lee, “Generative Adversarial Network/Basic Idea/Machine Learning and having it deep and structured (2017,Spring)” http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html

AUTO-ENCODER



From Hung-yi Lee, “Generative Adversarial Network/Basic Idea/Machine Learning and having it deep and structured (2017,Spring)”
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS17.html

10

Generative Adversarial Networks (GAN)

Deep Learning

ALGORITHM

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

From “Generative Adversarial Networks”
https://arxiv.org/abs/1406.2661

12

Wasserstein GAN (WGAN)

Deep Learning

ALGORITHM

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

From "Wasserstein GAN"
https://arxiv.org/abs/1701.07875

14

Improved Wasserstein GAN (WGAN-GP)

Deep Learning

ALGORITHM

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

```
while θ has not converged do
    for t = 1, ..., n_critic do
        for i = 1, ..., m do
            Sample real data x ~ P_r, latent variable z ~ p(z), a random number ε ~ U[0, 1].
            x̃ ← G_θ(z)
            x̂ ← εx + (1 − ε)x̃
            L^(i) ← D_w(x̃) − D_w(x) + λ(‖∇_x̂ D_w(x̂)‖_2 − 1)^2
        end for
        w ← Adam(∇_w (1/m) Σ_{i=1}^m L^(i), w, α, β_1, β_2)
    end for
    Sample a batch of latent variables {z^(i)}_{i=1}^m ~ p(z).
    θ ← Adam(∇_θ (1/m) Σ_{i=1}^m −D_w(G_θ(z)), θ, α, β_1, β_2)
end while
```

Where:

$$\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$$

$$\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$$

$$L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$$

$$w \leftarrow \text{Adam}\left(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$$

$$\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2\right)$$

From “Improved Training of Wasserstein GANs”
https://arxiv.org/abs/1704.00028

16

Requirement

Using GAN To Implement DGA

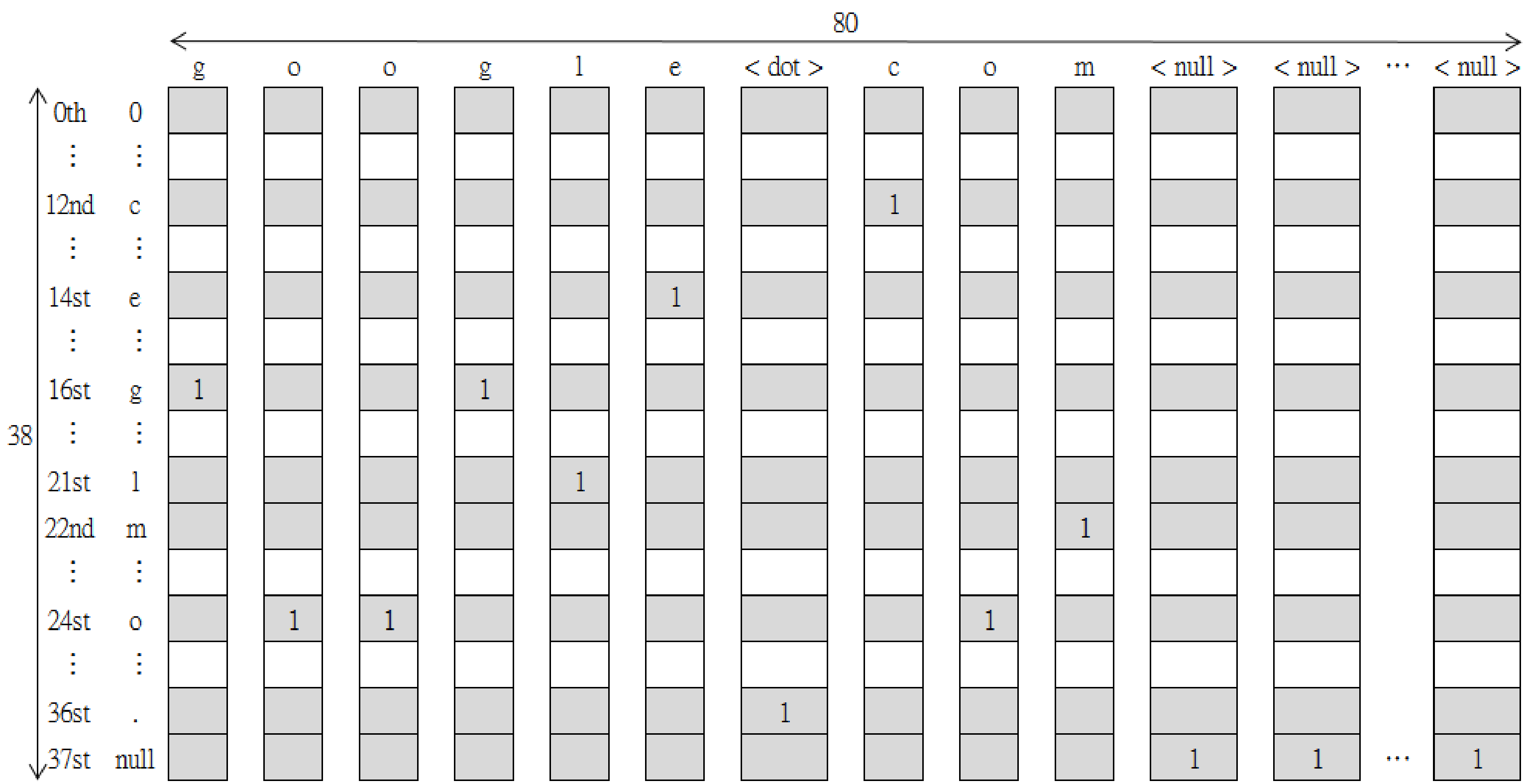
REQUIREMENT

- Q: Where is my training data?
- A: From Alexa Top 1-million sites.

	A	B
1	1	google.com
2	2	youtube.com
3	3	facebook.com
4	4	baidu.com
5	5	yahoo.com
6	6	wikipedia.org
7	7	amazon.com
8	8	twitter.com
9	9	qq.com
10	10	google.co.in
11	11	live.com
12	12	taobao.com
13	13	bing.com
14	14	google.co.jp
15	15	msn.com
16	16	yahoo.co.jp
17	17	linkedin.com
18	18	sina.com.cn
19	19	weibo.com
20	20	vk.com
21	21	instagram.com
22	22	google.ru
23	23	yandex.ru
24	24	google.de
25	25	hao123.com
26	26	ebay.com
27	27	reddit.com
28	28	google.co.uk
29	29	amazon.co.jp
30	30	t.co
31	31	google.com.br
32	32	mail.ru
33	33	google.fr
34	34	pinterest.com
35	35	netflix.com

REQUIREMENT

- Q: How to use this data?
- A: I deal these domain names as 38x80 pictures.



REQUIREMENT

- Because we don't have the "code". We have to train an Auto-encoder model.