GAN 기반 폰트 생성

이세훈*, 김민재O, 권혁정*
인하공업 전문대학, 컴퓨터 시스템과O
인하공업 전문대학, 컴퓨터 시스템과*
e-mail: seihoon@inhatc.ac.kr, coldgame@nate.com

GAN based Fonts Generation

Se-Hoon Lee*, Min-Jae KimO, Hyeok-Jeong Kwon*
Dept. of Computer Systems & Engineering, Inha Technical CollegeO
Dept. of Computer Systems & Engineering, Inha Technical College*

● 요 약 ●

한글 폰트를 만드는 데는 자음+모음 조합으로 약 11,500자 정도의 글자가 필요하다. 디자이너가 글자 하나씩 전부 디자인 하는 것도 굉장한 부담요소이고, 한글폰트를 제작하는데 있어 3개월 이상의 소요 기간과 3000만 원 이상의 비용부담 또한 무시 못 할 요소이다. 게다가 카피라이트 폰트에 대한 저작권 문제 또한 골칫거리다. 그래서 이를 최소한으로 하고자 딥 러닝의 방식중 하나인 GAN(생성적 적대 신경망)을 통해서 디자이너가 399자만 작성하고 나머지는 컴퓨터가 디자이너의 폰트 디자인을 인식하고 자동으로 만들어 주는 프로그램을 고안하였다.

키워드: 폰트 생성(Font generation), 생성적 적대 신경망(Generative Adversarial Network), 폰트 저작권(Font copyright)

I. Introduction

폰트 저작권 문제가 사회적으로 여러 가지 파장을 낳고 있다. 저작권법상 보호대상이 되는 것은 '폰트 도안'이 아니라 '폰트 파일'이다. 폰트 도안은 일반적으로 동일한 모양, 형태, 크기를 갖춘 한 벌의 글꼴을 의미하며, 서체, 글꼴 디자인, 글자체 등으로 부르기도 한다. '폰트 파일'은 컴퓨터에 저장되는 전자적인 파일을 의미하며, 일반적으로 PC의 fonts폴더에 'xxx.ttf'라는 파일명이며 법원은 '폰트 도안'은 보호대상이 아니고 개별적인 '폰트 파일'이 저작권법상의 보호대상이라고 판결을 내렸다. 그리고 또한 알파벳과 다르게 한글 폰트는 자음과 모음을 조합하면 약 11,500자 정도가 나오며 디자이너가 폰트 하나를 만드는데 일일이 디자인하고 다 만들기엔 약 3개월 정도의 시간이 소모되고 또한 비용도 크다. 따라서 본 논문에서는 저작권 문제와 시간, 비용을 절약하기 위해 11,500자를 디자이너가 다 쓰는게 아니라 399자만 쓰고 나머지는 딥러닝 GAN을 이용한 나머지 한글 폰트 생성 시스템을 구현한다.

II. The Proposed Scheme

1. S/W Architecture

딥 러닝(GAN)의 모델링 작업과 백그라운드 프로세싱은 Ubuntu OS (16.04) 에서 Python으로 진행 되었으며, TensorFlow 라이브러리를 이용해 GAN 알고리즘을 구현하였다. 폰트 이미지는 png 파일을 svg 파일로 벡터화 하는 Python 라이브러리를 사용하였고, 계정관리 컨트롤러와 사용자 UI는 각각 Java Spring framework, Node.js를 사용하였다.

2. Modelling Process

폰트 생성 GAN 모델은 다중 학습 모델을 이용하였다. 다중 학습 모델이란, 학습 데이터 셋이 한 개가 아닌 여러 개의 데이터 셋을 one-to-many 관계로 학습의 과정을 거친 모델을 의미한다. 본 논문에서 사용한 모델은 다중 학습 모델링을 실현하기 위해서 카테고리 임베딩을 도입했고, 캐릭터가 디코더를 통과하기 직전에 비 훈련용 가우스 노이즈를 문자 임베딩 스타일로 연결함으로서 이를 구현했다. 이렇게 함으로서 인코더는 동일한 문자를 동일한 벡터에 매핑 하는 반면에 디코더는 대상 캐릭터를 생성하기 위해 캐릭터와 스타일을 모두 사용하게 된다.

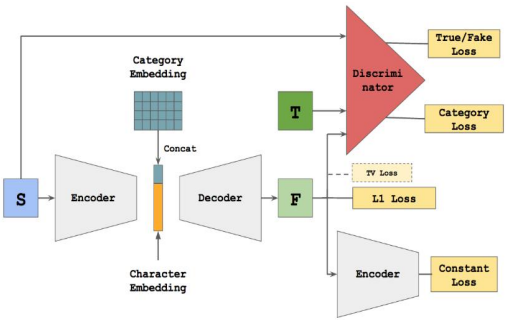

Fig. 1. First Modeling Process

본 논문에서 사용한 모델의 학습 과정은 두가지 스텝으로 진행된다. 첫 번째로, 여러 가지 소스 폰트를 이용하여 거대한 모델을 생성하고, 인코더를 동결시킨다. 인코더는 수천 개에 해당하는 캐릭터의 구조적 정보를 추출하고, 디코더는 그것을 강화하는 Transfer Learning을 수행한다. 인코더, 디코더, 구분자는 pix2pix 모델을 사용하였다. 두 번째 과정으로 1차적으로 완성된 거대한 모델 중 레이블을 섞는 셔플링 작업을 수행한다. 라벨 셔플링 이후 모델을 강화튜닝 한다. 1,2단계를 막론하고 모델 생성 시에 주의해야 할 사항은 데이터의 검증이다. 특정 문자가 누락되는지, 소스 폰트와 대상 폰트 간에 일관성이 유지되는 지의 여부를 잘 파악하고, 모델링해야 한다. 그렇지 않을 우 모델이 혼동을 일으켜 붕괴가 일어날 수 있다.

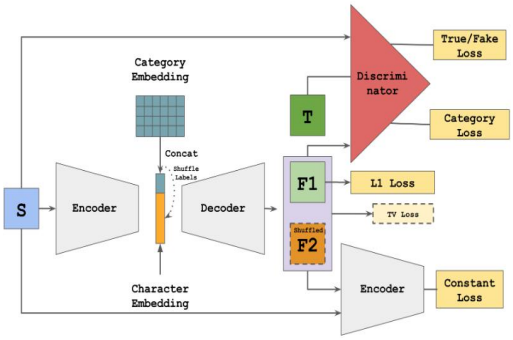

Fig. 2. Label Shuffling Proess

벡터 이미지를 다루기 위해 사용한 컨벌루전 레이어의 표준화 함수로는 ReLU와 Leaky ReLU를 사용하였으며, 생성자와 구분자의 최적화 함수로는 안정적으로 최적화를 위한 하강이 가능한 Adam을 사용하였다. 본 논문에 사용된 GAN의 하이퍼 파라미터는 Fig.3에서 확인할 수 있다.

Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_x(z)$ – 110 × 1 × 1 input						
Linear	N/A	N/A	384	×	0.0	ReLU
Transposed Convolution	5 × 5	2 × 2	192	√	0.0	ReLU
Transposed Convolution	5 × 5	2 × 2	96	√	0.0	ReLU
Transposed Convolution	5 × 5	2 × 2	3	×	0.0	Tanh
$D(x)$ – 32 × 32 × 3 input						
Convolution	3 × 3	2 × 2	16	×	0.5	Leaky ReLU
Convolution	3 × 3	1 × 1	32	√	0.5	Leaky ReLU
Convolution	3 × 3	2 × 2	64	√	0.5	Leaky ReLU
Convolution	3 × 3	1 × 1	128	√	0.5	Leaky ReLU
Convolution	3 × 3	2 × 2	256	√	0.5	Leaky ReLU
Convolution	3 × 3	1 × 1	512	√	0.5	Leaky ReLU
Linear	N/A	N/A	11	×	0.0	Soft-Sigmoid
Generator Optimizer	Adam ($\alpha = [0.0001, 0.0002, 0.0003]$, $\beta_1 = 0.5$, $\beta_2 = 0.999$)					
Discriminator Optimizer	Adam ($\alpha = [0.0001, 0.0002, 0.0003]$, $\beta_1 = 0.5$, $\beta_2 = 0.999$)					
Batch size	100					
Iterations	50000					
Leaky ReLU slope	0.2					
Activation noise standard deviation	[0, 0.1, 0.2]					
Weight, bias initialization	Isotropic gaussian ($\mu = 0$, $\sigma = 0.02$), Constant(0)					

Fig. 3. Convolution Hyper Parameter

3. Templates

미리 제공되는 템플릿에 유저가 탬플릿에 있는 399자를 적어서 업로드 하면 파이썬 Pillow 라이브러리를 이용해 자동으로 각 글자를 잘라낸 후 각 유닛들을 한글 유니 코드에 매핑 한다. 변화하는 작업은 svgicons2svgfont , svg2ttf를 사용하였다. 이렇게 잘라낸 유닛들은 각자의 유니 코드 값을 들고 쪼개지고, 해당 유니 코드들은 각자의 레이블 값이 된다.

4. Training & Result

템플릿에서 잘라져 나온 캐릭터들을 모델에 넣고, 모델을 트레이닝 시키면서 나머지 11000자를 생성한다. 모델의 학습 과정은 총 30 에폭으로 구성되며 Learning_late 초기 값은 0.001, 배치 사이즈는 16이며, 강화 학습을 위해 10 에폭마다 Learning_late가 절반으로 줄어들게 디폴트 값으로 세팅했다.

Fig. 4. Training

모델링이 끝나고 모든 캐릭터가 완성되면 지정한 경로로 모델이 완성한 나만의 폰트파일인 ttf file이 생성되고, 쉽게 설치하여 사용할 수 있다. 탬플릿에 들어갈 손글씨 데이터 셋이 저작권에 걸리는 데이터 여도, 카피레프트 폰트들로 미리 학습된 모델을 통해 나온 데이터라 저작권 문제에 관해서는 안심하고 사용할 수 있다.

III. Conclusions

본 논문에서는 GAN을 이용해 한글 폰트를 생성하였다. 모델을 생성할 때, 정적인 소스 폰트 데이터 셋을 통해 모델링을 진행하는데, 이후에 진행할 연구에서는 모델링 과정에서 차가운 데이터 셋, 따듯한 데이터 셋, 장난스러운 데이터 셋 등. 다양한 성격을 가진 데이터 셋을 각각의 모델로 만들어서 사용자가 원하는 폰트를 제공해 줄 수 있게 하는 플랫폼을 개발할 예정이다.

REFERENCES

[1] Jae-Jon Yoo, "Generative Adversarial Nets," http://jaejunyoo.blogspot.com/2017/01/generative-adversarial-nets-1.html January 2017.

[2] Augustus Odena, Christopher Olah, Jonathon Shlens, Conditional Image Synthesis With Auxiliary Classifier GANs https://arxiv.org/pdf/1610.09585.pdf