KIPS Trans. Softw. and Data Eng.
Vol.10, No.11 pp.473~482
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2021.10.11.473

Few-Shot Korean Font Generation based on Hangul Composability

Jangkyoung Park† · Ammar Ul Hassan†† · Jaeyoung Choi†††

ABSTRACT

Although several Hangul generation models using deep learning have been introduced, they require a lot of data, have a complex structure, requires considerable time and resources, and often fail in style conversion. This paper proposes a model CKFont using the components of the initial, middle, and final components of Hangul as a way to compensate for these problems. The CKFont model is an end-to-end Hangul generation model based on GAN, and it can generate all Hangul in various styles with 28 characters and components of first, middle, and final components of Hangul characters. By acquiring local style information from components, the information is more accurate than global information acquisition, and the result of style conversion improves as it can reduce information loss. This is a model that uses the minimum number of characters among known models, and it is an efficient model that reduces style conversion failures, has a concise structure, and saves time and resources. The concept using components can be used for various image transformations and compositing as well as transformations of other languages.

Keywords : Hangul Font Generation, Font Components, GAN, Few-shot

한글 조합성에 기반한 최소 글자를 사용하는 한글 폰트 생성 모델

박 장 경† · Ammar Ul Hassan†† · 최 재 영†††

요 약

최근 딥러닝을 이용한 한글 생성 모델이 연구되고 있으나, 한글 폰트의 구조가 복잡하고 많은 폰트 데이터가 필요하여 상당한 시간과 자원을 필요로 할 뿐 아니라 스타일이 제대로 변환되지 않는 경우도 발생한다. 이러한 문제점을 보완하기 위하여, 본 논문에서는 한글의 초성, 중성, 종성의 구성요소를 기반으로 최소 글자를 사용하는 한글 폰트 생성 모델인 CKFont 모델을 제안한다. CKFont 모델은 GAN을 사용하는 한글 자동 생성 모델로, 28개의 글자와 초/중/종성 구성요소를 이용하여 다양한 스타일의 모든 한글을 생성할 수 있다. 구성요소로부터 로컬 스타일 정보를 획득함으로써, 글로벌 정보 획득보다 정확하고 정보 손실을 줄일 수 있다. 실험 결과 스타일을 자연스럽게 변환되지 못하는 경우를 감소시키고 폰트의 품질이 향상되었다. 한글 폰트를 생성하는 다른 모델들과 비교하여, 본 연구에서 제안하는 CKFont는 최소 글자를 사용하는 모델로, 모델의 구조가 간결하여 폰트를 생성하는 시간과 자원이 절약되는 효율적인 모델이다. 구성요소를 이용하는 방법은 다른 언어 폰트의 변환은 물론 다양한 이미지 변환과 합성에도 사용될 수 있다.

키워드 : 한글폰트생성, 폰트구성요소, GAN, Few-shot

1. 서 론

폰트 디자이너가 새로운 폰트를 만들 때 알파벳을 기본으로 하는 로마자는 대소문자 52개만 디자인하면 되지만 한글은 11,172자를 디자인하여야 한다. 이는 30분에 한 글자씩 만들어도 하루 8시간 작업 시 700일이 소요되는 노동 집약적인 작업이다. 그러나 오늘날 인공지능을 이용한 딥러닝 모델을 이용하면 256개의 글자만 디자인하면 30분 만에 새로운 폰트 하나를 만들 수 있다[1].

최근에는 딥러닝을 이용하여 새로운 폰트를 생성하는 다양한 방법들이 연구, 개발되고 있다. 자동 폰트 생성의 가장 기본이 되는 연구는 GAN(Generative Adversarial Networks)[2]을 이용하여 다양한 폰트 스타일로 이미지를 변환하는 I2I (Image-to-Image translation)[3] 프레임을 사용하는 연구이다. 또한 더 빠르고 더 효율적인 방법으로 목표를 달성하기 위하여, 새로운 폰트를 생성하는 것뿐 아니라 보다 적은 글자 수를 사용하여 시간과 자원을 절약할 수 있는 방법에 대한 연구가 진행되고 있다[4, 5]. zi2zi 모델[6]은 모든 한글 11,172자를 생성하기 위하여 399개의 글자를 사용하며, SKFont 모델은 114개 글자를 사용하고[7], DM-Font 모델은 30자를

※ 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2016-0-00166).
† 비 회 원 : 숭실대학교 컴퓨터공학부 박사과정
†† 준 회 원 : 숭실대학교 컴퓨터공학부 박사과정
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : September 30, 2021
Accepted : October 11, 2021
* Corresponding Author : Jaeyoung Choi(choi@ssu.ac.kr)

사용한다[8]. 보다 적은 글자를 사용하려고 시도하는 이러한 모델들은 많은 학습 데이터를 필요로 하고 구조가 복잡하며, 일부 글자에서 스타일 정보의 부정확한 획득과 정보 손실 사례가 발생하여 글자의 품질이 저하되는 공통적인 결함이 존재한다. 본 논문은 이렇게 관찰된 개선이 필요 부분에 주목하여 아래 세 가지 목표를 정하고 연구를 진행하였다.

- ◦ 정확한 스타일 정보 획득과 정보 손실 없이,
- ◦ 얼마만큼의 최소 글자로,
- ◦ 시간과 자원을 절약하며 모든 한글을 생성할 수 있는 간결한 모델 설계

한글은 14개의 자음과 10개의 모음을 기반으로, 모든 한글은 19개의 초성과 21개 중성, 그리고 27개의 종성 등 모두 67개 (19+21+27)의 구성요소가 초성, 중성, 그리고 종성의 순서대로 조합되어 모두 11,172자 (19×21×28, 종성이 없는 경우 포함)의 조합형 형태의 글자를 생성한다. 이 67개 구성요소가 모든 한글 생성의 근본 요소이며, 이것들로 모든 글자를 생성할 수 있다. 또한 이 67개 구성요소를 모두 포함하는 최소 글자 수는 28개 글자이며 (종성이 없는 경우를 포함한 종성의 수), 이론적으로 28개의 최소 글자로부터 구성 요소들을 추출하여 모든 글자를 생성할 수 있다.

본 논문에서는 이러한 한글의 조합적인 특성을 이용하여 28개의 최소 글자와 67개의 구성요소로 모든 한글을 생성할 수 있는지를 검증하고, I2I 변환 방법의 cGAN (conditional GAN)[9]을 기반으로 새로운 한글 자동 생성 모델인 CKFont (Component-based Korean Font Generation Model)를 제안한다.

2. 관련 연구

글자의 이미지를 다른 스타일의 폰트로 변환하는 작업은 컴퓨터 성능이 획기적으로 향상되고, 동시에 딥러닝 기술이 개발되기 전까지는 거의 불가능한 일이었다[4, 5]. GPU와 같은 컴퓨터 프로세서의 성능 향상과 새로운 딥러닝 알고리즘이 접목되어, 다양한 스타일을 가진 글자의 자동 생성 및 변환이 가능해지면서, 로마자는 물론 한글, 한자를 포함한 많은 언어의 글자를 생성하는 연구들이 활발히 진행되고 있다[10-15].

글자의 스타일을 변환하여 폰트를 자동 생성할 수 있는 방법 중 I2I 방법[3]이 가장 널리 사용되고 있으며, CNN (Convolutional Neural Networks)[4], GAN[4, 5], VAE (Variable Auto Encoder)[5, 16] 등의 프레임과 접목되어 다양한 방법이 글자 이미지 변환 모델에 사용되고 있다. 이러한 글자 생성 모델은 폰트의 이미지를 대상 스타일로 변환하는 기본적인 목표와 함께 가능한 최소한의 글자를 사용하여 시간과 자원을 절약할 수 있는 우수한 성능의 모델 개발을 주된 목표로 하고 있다[8, 14, 15].

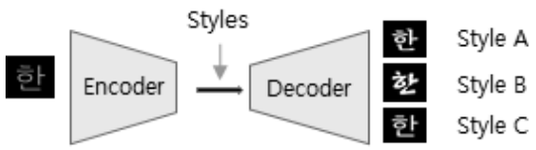

Fig. 1. zi2zi Architecture of Generator

시간과 자원을 절약할 수 있는 적은 글자 수를 사용하는 모델은 399자를 사용하는 zi2zi 모델[6]과 114자를 사용하는 SKFont 모델[7], 30자를 사용하는 DM-FONT 모델[8] 등이 있으나, 많은 데이터를 필요로 하며, 구조가 복잡하고, 시간과 자원이 많이 소요된다. 또한 스타일 손실이 발생하여 글자 품질이 저하되는 사례가 빈번히 일어나는 약점이 있다. 이러한 원인을 분석하기 위하여 위에서 언급한 3개 모델의 구조를 살펴보고 보완할 방향을 논의한다.

zi2zi 모델[6]은 pix2pix[3] 프레임워크의 일대일 스타일 변환 기능을 일대다(one-to-many) 변환이 가능하도록 기능을 개선한 모델로, DCFont[11], SCFont[12] 등 한자 폰트를 생성하는 모델에서 사용되고 있다. 한글을 생성할 수 있도록 수정된 zi2zi 모델은 399자의 한글로 모든 한글을 비교적 잘 생성하였다. Fig. 1에서와 같이 zi2zi 모델은 대상 글자의 스타일 분류 값을 별도로 입력하여, 인코더에서 생성된 글자 내용 벡터와 병합시켜 디코더로 입력하는 방식이다.

SKFont 모델[7]은 zi2zi 모델을 기반으로 한 I2I 변환 방법으로 3개의 네트워크를 직렬로 연결하여 사용한다. Fig. 2에서와 같이, 첫번째 네트워크(F2F)는 소스 폰트를 대상 폰트로 변환시키고, 두 번째 네트워크(F2S)에서는 변환된 대상 폰트에서 골격을 추출하여, 마지막 네트워크(S2F)에서 추출된 골격을 다시 대상 폰트로 변환시키는 구조이다. SKFont 모델은 대상글자의 스타일 분류 값을 zi2zi 모델과 달리 글자 내용에서 글자 스타일 벡터를 추출하여 디코더로 입력하는 방식으로, 3개의 네트워크를 사용하여 구조가 복잡하고 시간과 자원이 많이 소요된다.

DM-Font 모델[8]은 한글의 구성요소를 이용하여 한글만 생성하는 모델로 2개의 메모리를 사용하여 스타일 정보를 획득하는 구조를 가지고 있다.

Fig. 3에서 보듯이 한글의 구성요소를 학습하여 저장한(m1) 다음에, 대상글자의 구성요소별로 비교하여 정보를 획득하고(m2), 30자의 적은 글자로 모든 글자를 생성하는 모델이다. 또한 LF-Font[14], MX-Font[15] 등 한자 생성이 가능한 후속 모델이 개발되었다. 그러나 모델들이 여전히 구조가 복잡하고 많은 데이터와 시간 및 자원을 필요로 한다.

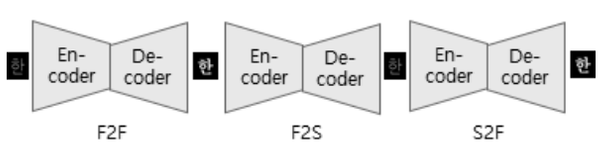

Fig. 2. SKFont Architecture

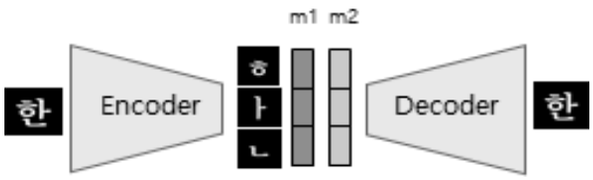

Fig. 3. DM-Font Architecture

(a) zi2zi (b) SKFont

(c) DM-Font

Fig. 4. The Upper Left (a) shows the zi2zi Model, the Right (b) shows the SKFont Model, and the Lower (c) shows the DM-Font Style Loss [6-8].

스타일을 변환하여 이미지를 생성하기 위해서는 폰트의 내용(content) 정보와 함께 폰트 스타일 정보를 잘 획득하고 전달하여야 한다. 위의 3개 모델은 모두 우수한 글자 생성 성능을 보임에도 불구하고, Fig. 4에서 보듯이 이미지를 생성하면서 글자 스타일에 대한 정보 손실이 나타나고 있다.

또한 3개 모델들은 입력하는 글자 수를 줄인 모델이지만 여전히 많은 입력 데이터를 사용하고 있으며, 구조가 복잡하고, 상당한 시간과 많은 자원이 소요된다. 따라서 보다 적은 수의 입력 글자로 시간과 자원이 절약되는 간결하고 효율적인 모델 개발이 요구되고 있다. 다음 장에서 우리는 먼저 한글의 구조적 특성과 구성요소에 대하여 논의하고 이러한 구조적 특성을 활용하여 최소 글자 수를 결정하고 이 최소글자로 모든 글자를 생성하는 간결한 모델에 대하여 논의한다.

3. 한글의 구조적 특징

3.1 한글의 구성요소

한글은 자음(14자)과 모음(10)의 24자로 구성된 글자이다. Fig. 5에서 보듯이 자음에서 확장된 5개의 쌍자음과 11개의 복자음, 그리고 모음에서 확장된 11개의 복모음으로 모두 51개 (14+5+11=30개 자음, 10+11=21개 모음)의 구성요소가 있다. 51개 구성요소가 조합되어 글자를 만드는데 모든 글자는 [초성 + 중성] 또는 [초성 + 중성 + 종성]의 순서대로 조합된 형태를 가진다.

이와 같이 한글은 초성에 사용되는 자음 19개 (14자음+5 쌍자음)와 중성에 사용되는 모음 21개 (10모음+11복모음), 종성에 사용되는 자음 27개 (14자음+2쌍자음+11복자음)를 순서대로 조합하여 만든 글자이다. 자음과 모음이 조합되는 규칙은 최소 한 개의 초성(자음)과 한 개의 중성(모음)이 순서로 조합되며, 여기에 종성(자음)이 없거나 붙여진 것을 말한다. 따라서 초성에는 19개 경우의 수가, 중성에는 21개, 종성에는 28개(종성이 없는 경우 포함)의 경우의 수가 있으며 이것을 조합하면 총 11,172 가지의 경우의 수(19 x 21 x 28)가 되고 이것이 모든 가능한 조합형 한글의 글자 수가 된다.

또한 Fig. 5에서와 같이 모든 한글을 이루고 있는 구성요소는 초성과 중성, 종성의 합인 67개(19 + 21 + 27) 구성요소이다. 초성과 종성이 같은 16자(14자음 + 2 쌍자음)를 제외하면 모든 한글을 구성하는 구성요소는 위에서 말한 것과 같은 51개 (30개 자음 + 21개 모음) 구성요소가 되고, 종성이 없는 경우를 포함하여 총 구성요소는 52개가 된다.

3.2 최소 글자 수

한글은 초성, 중성, 종성의 68개 구성요소로 모든 조합의 경우의 수인 11,172자의 글자가 있으며, 모든 조합된 조합형

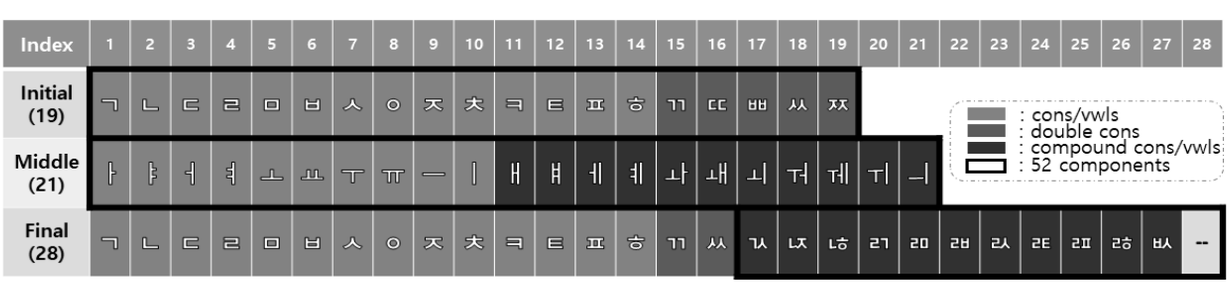

Fig. 5. Hangul's Initial, Middle and Final Components and Consonants/Double Consonants/Compound Consonants, Vowels/Double Vowels and 52 Components

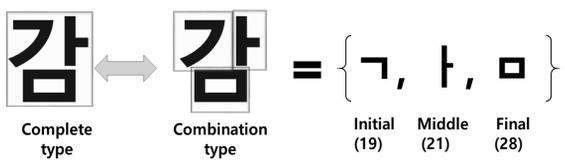

Fig. 6. Complete and Combination Type of Hangul(sample)

Fig. 7. 28 Characters Including All Components (Sample)

글자(Combination Type)는 예외 없이 초성과 중성과 종성의 합인 경우만 존재하고, 모든 글자는 초성, 중성, 종성의 순서대로 조합되는 규칙을 따르고 있다. 또한 반대로 모든 완성형 글자(Complete Type)는 초/중/종성으로 완벽하게 구성요소별로 분리될 수 있으며, 그 구성요소는 52개 구성요소로 이루어져 있다.

Fig. 6은 한글의 완성형과 조합형 그리고 구성요소가 어떻게 분해되는지를 보여준다. 여기서 우리는 모든 구성요소를 포함하는 최소 입력 글자 수는 종성의 수와 같은 28개 글자라는 것을 알 수 있다. 52개의 구성요소를 모두 포함하는 28개의 글자만 있으면 이론상 모든 한글을 생성할 수 있다. 상용 한글 2,350글자[1]중에서 모든 구성요소를 포함하는 28자를 발췌하면 Fig. 7과 같다.

많은 28개 입력 글자 세트가 있을 수 있으며, Fig. 7은 그 중 하나의 예이다. 이 28개 글자로부터 52개의 구성요소를 학습하고 규칙에 따라 조합하면 모든 한글을 만들어 낼 수 있다.

우리는 이러한 한글의 조합적 특성을 활용하고 초/중/종성의 모든 구성요소를 포함하는 최소 글자 수인 28개의 입력 글자로 모든 한글을 다양한 스타일로 자동 생성하는, GAN을 기반으로 한 새로운 한글 자동 폰트 생성 모델을 개발하였다.

4. CKFont 모델

모든 한글은 한 글자의 예외도 없이 규칙에 따라 조합되는 조합성에 근거하여 변환 대상 글자를 초/중/종성으로 분리하여 분리된 구성 요소로부터 상세한 스타일 정보를 획득하는 새로운 구조의 CKFont 모델을 설계하였다. 또한 52개 구성요소를 모두 포함하는 28개 글자만으로 모든 글자를 생성하여 상당한 시간과 자원을 절약할 수 있는 간결한 모델로 설계하였다.

4.1 모델 구조

최소 글자로 우수한 품질의 폰트를 생성해 내기 위한 방법은 대상 스타일 벡터 값을 정확히 추출하여, 딥러닝 모델이 학습을 진행하는 동안 정보의 손실을 최소화하도록 하는 것이다. 우리는 이를 목표로 두 개의 인코더를 사용하였다. 하나는 폰트 내용 정보를 생성하고, 다른 하나는 대상 폰트 스타일 정보를 생성하는 구조를 사용하였다. 또한 대상 폰트 스타일 정보를 대상 폰트 구성요소에서 바로 획득하는 것으로 설계하여, 보다 더 정확한 스타일 정보를 얻을 수 있도록 하였으며, 손실을 최소화하기 위하여 구성요소별 이미지를 지속하도록 하였다. 실험을 통해 스타일 정보가 잘 획득되고 전달되고, 시간과 자원이 상당히 절약되는 매우 효과적인 구조의 모델이라는 것을 확인하였다. Fig. 8은 일반적인 GAN 구

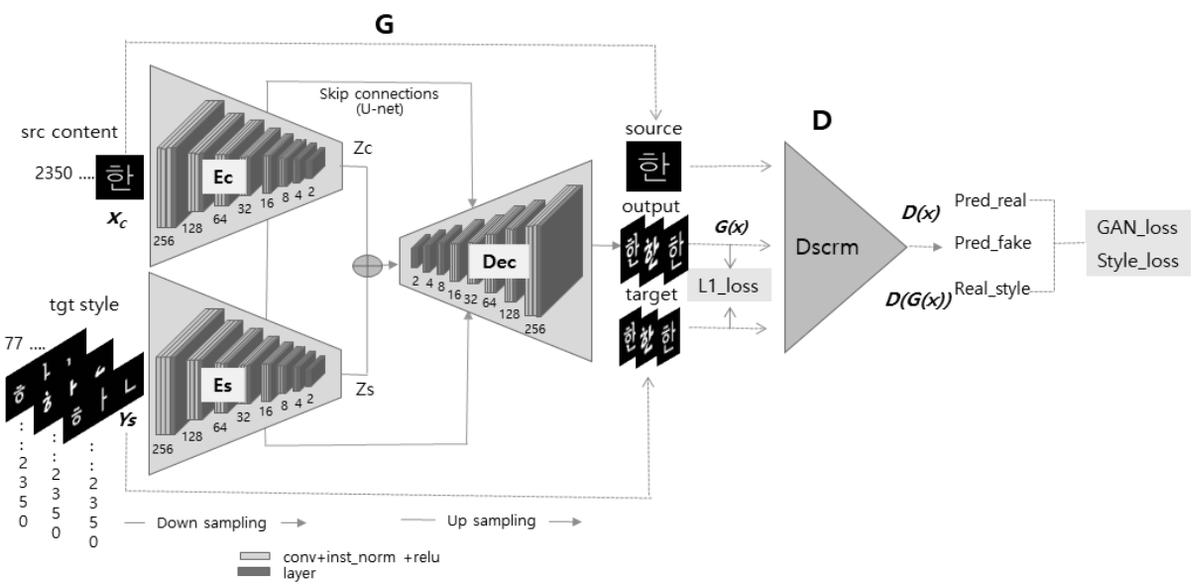

Fig. 8. CKFont Architecture

1) 한글 표준코드체계(KS X 1001).

조 프레임에 두 개의 인코더를 사용하는 것으로 수정된 CKFont 모델 구조이다.

CKFont는 Fig. 8에서와 보는 바와 같이 일대다 프레임으로 한글의 조합성과 구성요소를 활용한 간결한 모델이다. 입력으로 내용(Xc) 이미지와 변환 대상 글자 구성요소의 스타일(Ys) 이미지를 사용하였다. 이를 분리하여 입력하기 위해 2개의 인코더 구조(Ec, Es)를 가지는 cGAN 구조를 가지며, 디코더의 입력으로 Zc와 Zs를 병합한 벡터를 사용한다.

각각의 인코더 Es와 Ec는 다운 샘플링되며 6개 계층을 가지고 있다. 첫 번째 계층 (kernel=7, stride=1)을 제외한 각 계층은 kernel=4, stride=2를 사용한 convolution과 Instance Normalization, Leaky ReLU 함수를 거치며, 모든 계층은 skip connections 구조를 가진다. 결과물 잠재벡터 Zc와 Zs는 병합되어 디코더 입력으로 사용되며 다시 업샘플링되어 변형된 스타일로 내용을 생성해낸다.

디코더도 6개 계층을 가지고, 각 계층은 kernel=5, stride =1을 사용한 deconvolution과 Instance Normalization, ReLU 함수를 거친다. 모든 계층은 인코더 계층과 내용/스타일이 업샘플링된 벡터와 병합되는 skip connection 구조를 가지며, 마지막에 convolution과 tanh 함수를 거쳐 결과물 이미지($G(x)$)를 생성한다.

이 생성된 결과물이 판별기 D (Discriminator)에 입력되며 소스 이미지와 생성된 결과물 이미지를 비교하여 GAN 손실을 계산한다. 소스 이미지에서 생성된 벡터와 생성된 결과물 이미지 벡터를 비교하여 스타일 손실, 그리고 대상 이미지와 생성된 결과물 이미지를 비교하여 $L1$ 손실을 계산하고 갱신을 반복하며 생성된 이미지의 품질을 향상시킨다 (code참조: http://github.com/xjnpark/ CKFont).

4.2 목적 함수(Objective Function)

모델의 목적함수(Objective Function, G)는 Equation (1)과 같이 Adversarial 손실 (L_{ADV}), 스타일 손실(Ls), L1 손실 (L_{L1}) 등 손실함수의 합으로 표시되며, λ는 학습할 때 Hyper parameter로 손실의 가중치로 작용한다.

$$G = \arg\min_G \max_D L_{ADV}(G,D) + \lambda_s L_S(G,D) + \lambda_{L1} L_{L1}(G) \quad (1)$$

Adversarial 손실 (L_{ADV}): cGAN의 L_{ADV}는 Equation (2)와 같이 D는 입력받은 가짜 이미지($G(x)$)가 진짜(True:1)인지 가짜(False:0)인지를 판별($D(y)$)하여 최대화 ($D(y) =1$) 한다. 생성기 G (Generator)는 가짜 이미지를 생성하고($G(x)$), D는 진짜 이미지를 판별하기 위하여 ($D(G(x))=1$) $G(x)$가 최소가 되도록($G(x)= 0$) 한다.

$$\min_G \max_D L_{ADV}(G,D) = E_y[\log D(y)] + E_x[\log(1-D(G(x)))] \quad (2)$$

스타일 분류 손실 (L_S): 일대다의 스타일 변환된 이미지를 생성하기 위해서 D는 진짜와 가짜를 판별함과 동시에 스타일이 대상 스타일과 같은지를 판별($D(Ys)$) 한다. 이미지의 폰트 스타일을 유지하기 위하여 D는 폰트 스타일을 예측하고, G로 피드백하여 손실을 줄이고 G가 올바른 스타일의 폰트를 생성하도록 한다. 이는 Adversarial 손실과 같은 개념으로 $D(y_S)$를 최대화하고 $G(x_S)$를 최소화하는 Equation (3)과 같다.

$$\min_G \max_D Ls(G,D) = E_y[\log D(y_S)] + E_x[\log(1-D(G(x_S)))] \quad (3)$$

L1 손실(L_{L1}): L_{L1}는 G가 가짜 이미지를 생성하고(G(x)) 대상 이미지(Y)와 픽셀별로 비교한 MAE(Mean Absolute Error)를 감소시켜 두 이미지가 같게 되도록 하며 Equation (4)와 같이 나타낼 수 있다.

$$L_{L1} = E_{x,y}[\,||\, y - G(x) \,||\,] \quad (4)$$

5. 실 험

5.1 사전학습(Pretraining)

원하는 폰트 스타일로 새로운 폰트를 생성하기 위해서는 먼저 글자 이미지 자체를 생성하기 위한 사전학습(pre-training)을 하여야 한다. 가능한 많은 글자 데이터를 이용하여 글자의 구성요소를 학습하고 새로운 스타일의 글자를 생성하는 방법을 미리 학습한다.

사전학습에 사용할 학습과 테스트 데이터 셋을 만들기 위해 상용 한글 2,350자에서 학습용으로 2,000자, 테스트용으로 350자로 분리하고 대상 폰트 스타일은 77개의 다양한 폰트스타일을 발췌[2]하여 60개를 학습에, 17개를 테스팅에 사용하였다. 한글을 초성과 중성, 종성으로 분리하기 위해 유니코드[3] 기반 python의 'jamo' API를 활용하여 모든 글자를 초/중/종성으로 분리하고 각 글자가 가지는 구성요소가 초성과 중성만으로 이루어진 경우와 초/중/종성으로 이루어진 글자를 식별하여 글자 조합 때 사용하고, 스타일 정보를 대상 글자의 구성요소에서 추출하기 위해 대상 스타일별로 구성요소를 분리하였다. 소스와 스타일별 대상글자와 대상글자 구성요소 이미지를 병합하여 256×256×5 (256×1,280) 사이즈 이미지 총 180,950 (2,350×77) 개의 이미지 데이터 셋을 만들었으며 Fig. 9는 샘플 데이터 셋 이미지이다.

이 데이터 셋으로 사전학습하고 테스트 한 결과물은 Fig. 10과 같다. 시각적인 판단으로 사전 학습한 결과는 만족할 만한 수준임을 확인할 수 있으며 이를 기반으로 28개 글자에 대한 학습을 진행하였다.

5.2 전이학습(Fine-Tuning)

처음 보는(unseen) 스타일로 글자를 생성하기 위하여 사전 훈련된 모델을 이용한 전이학습이 필요하다. 28개 글자를 처음 보는 17개 스타일로 모델을 전이학습 하였으며, 결과는 Fig. 11과 같다.

---

2) https://Hangul.naver.com/2011/font

3) https://github.com/SSLAB-SSU/hangul-font-dataset

Fig. 9. Dataset Images (Sample)

Fig. 10. Pretraining Sample Output

Fig. 11. Fine-tuning Output Sample

5.3 글자 생성

새로운 폰트 스타일로 모든 한글을 생성할 수 있는 가능성을 확인하기 위하여 처음 보는 17개 폰트 스타일로 256개 글자를 생성하였다.

결과는 Fig. 12에서와 같이 매우 우수한 품질의 글자를 생성하였으며 스타일 변화에도 잘 적응하여 고품질의 글자를 생성하였다. 그러나 일부 폰트 스타일에서 약간의 정보 손실이 나타나는 것을 Fig. 12에서 관찰할 수 있다.

Fig. 13은 글자 품질이 저하된 사례를 발췌한 샘플 이미지로 일부 폰트 스타일과 종성이 없는 경우에 집중해서 나타나고 있다. 이는 특정 폰트에 반응이 적절하지 못하고 종성이 없는 부분을 빈(blank) 이미지로 학습한 이유로 판단되며 향후 개선이 필요한 부분이다.

이렇게 28 글자로 256자를 생성하였으며 같은 방법으로 2,350 글자는 물론 모든 11,172 글자 원하는 스타일로 쉽게 생성할 수 있다.

Fig. 12. Sample Output of 256 Characters Created with Unseen 17 Font Styles

Fig. 13. Failure Cases(Sample)

6. 평 가

6.1 CKFont 와 zi2zi / SKFont 비교

우리의 모델 성능을 평가하기 위해 zi2zi와 SKFont 모델을 같은 2,000자로 학습하고 28자로 전이학습 하였으며, 처음 보는 17개의 스타일로 256개의 글자를 모두 같은 조건에서 동일하게 생성하였다.

정량적 평가로 생성된 이미지와 대상 이미지와의 L1, L2 손실, 유사도 (Structural Similarity Index, SSIM)의 통계적 값과, FID(Frechet Inception Distance) 점수[17]를 포함한 모든 데이터 값을 Table 1에 표시하였다.

Table 1에서 통계적 분석 값을 보면 L1, L2 손실 평균값에서 CKFont 모델은 대등하거나 우수하며 흩어짐의 정도를 나타내는 분산은 각각 0.0004~0.0006이나 zi2zi 모델의 L2 분산 값은 0.0016으로 3배 이상 차이를 보였다. 유사도의 분산은 CKFont 0.0006, zi2zi 0.0052로 8배 이상의 성능 차이를 나타내고 있으며, SKFont와 CKFont는 대등하고 안정적이다.

실제 이미지의 특징 벡터와 가짜 이미지의 특징 벡터 (생성기에서 생성) 사이의 거리를 계산하는 성능 지수인 FID점수는 점수가 낮을수록 생성기에 의해 생성 된 이미지의 품질이 실제와 비슷하고 더 우수하다는 것을 나타내는 것으로 CKFont가 71점으로 가장 우수하며 그다음은 zi2zi로 127점, SKFont 162점 순으로 나타났다. 이는 CKFont의 생성 이미지가 다른 두 모델에 비해 탁월함을 말해주고 있다.

정성적인 판단으로는 Fig. 14에서 보듯이 시각적으로도 비교 판단이 가능하며. zi2zi와 SKFont 모두 폰트 스타일 8, 9, 10, 11 등에서 글자 품질이 급격히 저하되는 것을 확인할 수 있다. 이에 비해 CKFont는 16 번째 폰트 스타일에서 약간의 품질 저하가 발생하였을 뿐 대체적으로 품질이 양호하다. 전체적으로 zi2zi 〈 SKFont 〈 CKFont 순으로 성능 차이가 나며 CKFont가 가장 우수한 것으로 나타났다.

Table 1. Values of Loss / SSIM / FID

Index		zi2zi	SFont	CKFont
L1 Loss	min	0.2520	0.2406	0.2524
	max	0.3364	0.3256	0.3306
	mean	0.2916	0.2725	0.2859
	var	0.0005	0.0006	0.0004
L2 Loss	min	0.2603	0.2419	0.2475
	max	0.4075	0.3141	0.3258
	mean	0.3081	0.2721	0.2814
	var	0.0016	0.0005	0.0005
SSIM	min	0.6865	0.8332	0.7968
	max	0.9061	0.9063	0.8961
	mean	0.8478	0.8700	0.8562
	var	0.0052	0.0005	0.0006
FID		127.19	162.91	71.52

Fig. 15-17은 세 모델의 성능을 종합적으로 비교할 수 있도록 시각화하여 그래프로 나타낸 것이다.

Fig. 15는 유사도와 Loss 값을 한 그래프로 나타낸 것으로 파란색은 CKFont, 초록색은 SKFont, 빨간색은 zi2zi 모델을 각각 나타낸다. Fig. 15에서 유사도(선그래프)는 zi2zi의 폰트 스타일 9, 12, 14 (붉은색 화살표)에서 매우 낮음을 나타내고 있으며, L1, L2 Loss(막대그래프) 또한 zi2zi 모델이 폰트 스타일 14, 15(빨간색)에서 크게 나타나고 있다. 이는 zi2zi 모델이 폰트 스타일의 변화에 일관성 있게 적응하지 못하고 있음을 말하며, 상대적으로 CKFont와 SKFont는 비교적 안정적이며 일관성이 있음을 보여준다.

Fig. 16은 유사도만 비교한 그래프로 폰트 스타일에 따른 모델별 유사도의 변화를 나타낸다.

Fig. 17은 Loss 값만 비교한 그래프로 zi2zi는 폰트 스타일 14, 15(붉은색 화살표)에서 Loss 값이 매우 높아 글자 내용 자체가 정확하지 않음을 보여준다. 또한 SKFont(초록색)

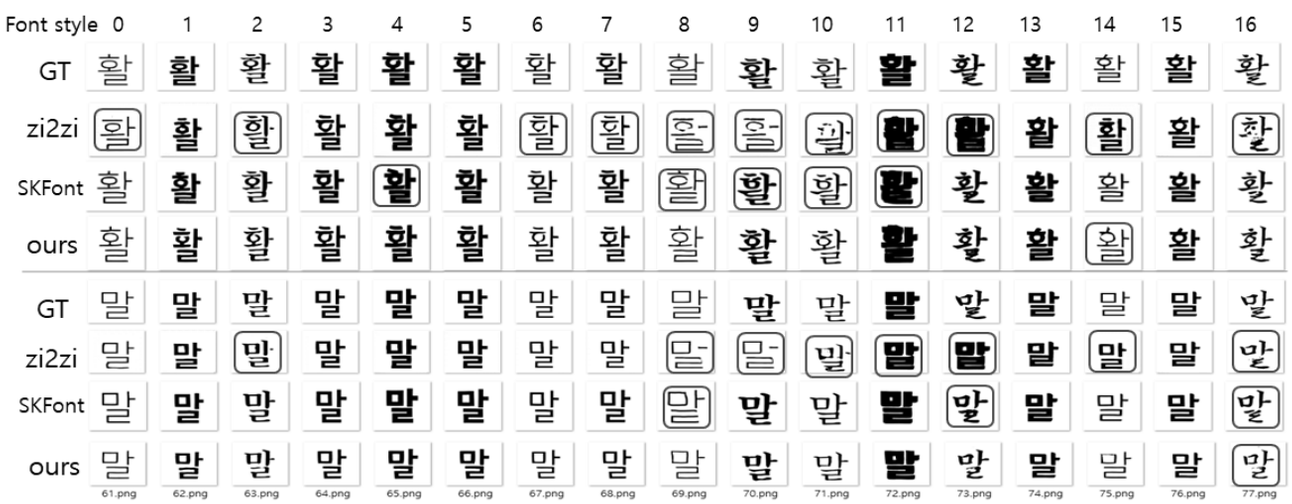

Fig. 14. Generated Output Comparison(Sample)

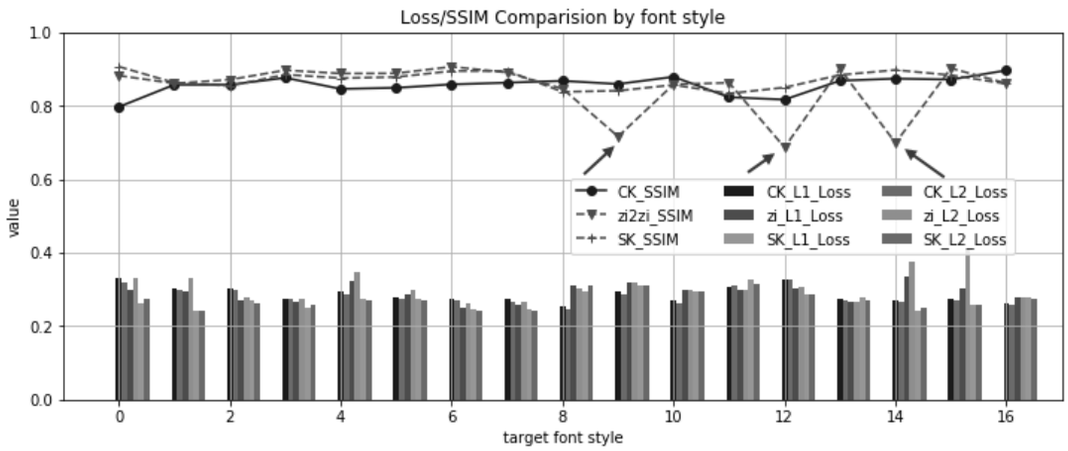

Fig. 15. SSIM / Loss Comparison by Models

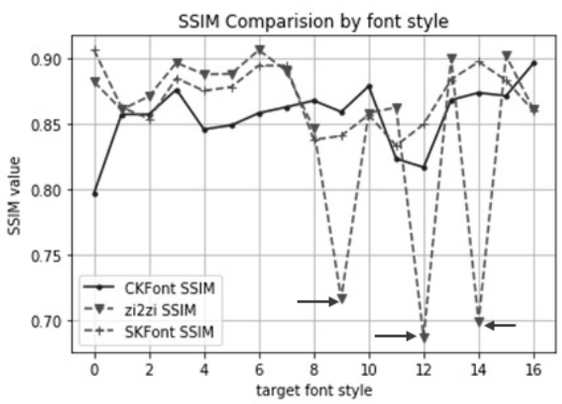

Fig. 16. SSIM Comparison

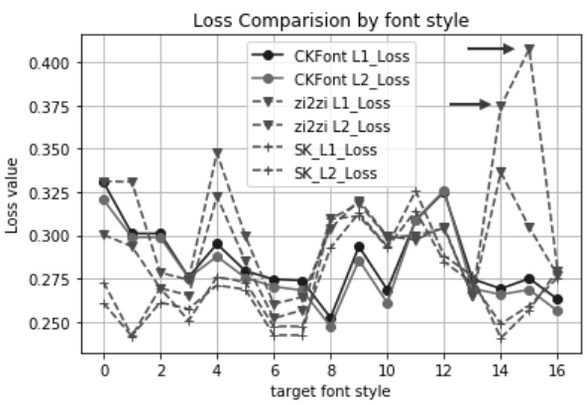

Fig. 17. Loss Comparison

는 폰트 스타일 9, 11에서 비교적 큰 품질 저하가 나타나고 있으나 CKFont(파란색)는 스타일 0과 12 외에 폰트 스타일의 변화에 잘 적응하며 일정한 성능을 나타내고 있다.

결론적으로 CKFont 모델이 모든 결과 값에서 zi2zi와 SKFont 보다 우수하며 zi2zi와 SKFont는 대등한 성능을 가진 것으로 나타났다. 시간과 자원의 비교는 측정이 불가할 정도로 엄청난 시간 차이를 보여주었다. CKFont가 약 24시간, zi2zi 50시간, SKFont 90시간 이상이 소요되었으며 메모리 사용도 시간에 비례하였다(우리는 NVIDIA GTX-2080Ti GPU 12GB-Memory Size 사용하였다).

이상에서 본 바와 같이 CKFont 모델은 서론에서 언급한, 최소글자로서, 스타일 손실을 최소화하며, 시간과 자원을 절약하는 간결한 모델 구축의 3가지 목표를 다 충족하였다고 할 수 있다. 그러나 유사도 값 86%와 26%대의 손실 값은 여전히 개선될 부분이 있음을 말하고 있으며, CKFont 역시 이미지 전환에 실패한 사례가 발생하고 있다. 이는 특정 대상 폰트 스타일(0, 12)과 종성이 없는 경우가 관련이 있어 보이며, 향후 추가적인 연구가 필요한 부분이다.

6.2 CKFont 의 확장

CKFont 모델을 한글 손글씨와 한자 생성에도 시험적으로 적용해 보았다. Fig. 18와 Fig. 19는 각각 한글 손글씨와 한자의 생성 결과로 시각적인 판단으로도 만족스러운 결과임을 확인할 수 있다. 이는 본 모델의 확장성을 보여주는 것으로 여러 분야에 유용하게 사용될 수 있을 것이다.

Fig. 18. Hangul Hand Writings Sample Output with Our Proposed Model CKFont

Fig. 19. Chinese Characters Sample Output with Our Proposed Model CKFont

7. 결 론

폰트 생성에 관한 연구는 다양한 폰트를 손쉽게 생성할 수 있는 방법을 제시하며, 보다 효과적인 모델 개발을 위해 많은 논문이 발표되었고 여전히 활발한 연구가 진행 중이다.

본 논문에서는 최소한의 입력 글자로, 정확한 스타일 정보를 획득하여 손실을 최소화하면서 정보를 전달하며, 시간과 자원을 절약할 수 있는 간결한 모델을 제시하였다. 특별히 한글의 구조적 특성을 분석하고 한글의 조합성에 근거하여 51개의 구성요소와 28자의 최소 입력 글자가 요구되는 과정을 설명하였고, 이를 본 연구에서 제시한 딥러닝 모델에서 입력으로 사용하여 새로운 한글 폰트를 생성하였다.

우리가 제안한 CKFont 모델의 결과를 zi2zi와 SKFont 모델의 결과와 비교 평가하여 CKFont 모델의 결과가 가장 우수함을 확인하였다. 본 모델은 정형화된 한글 폰트뿐만 아니라 한글 손 글씨에도 적용하여 우수한 결과를 보였다. 또한 여러 부수로 구성되는 한 자에 시험적으로 적용시켜 보았을 때도 아주 우수한 품질의 결과를 생성하는 것을 확인하였으며, 이는 구성요소를 이용한 다양한 변환 모델에 적용할 수 있는 가능성을 확인하였다. CKFont 모델 성능 개선은 물론 향후 여러 방면으로 구성요소 개념을 적용하는 연구를 계속 진행할 계획이다.

References

[1] J. B. Cha, "Few-shot handwriting copycat AI," *DEVIEW 2020 Session*, 2020. [Internet] [https://deview.kr/data/deview/session/attach/1400_T4, Jun. 10. 2021.

[2] I. Goodfellow, et al., "Generative adversarial networks," In *Advances in Neural Information Processing Systems*, ArXiv Preprint arXiv: 1406.2661, 2014.

[3] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*, (CVPR), 2016.

[4] S. Weidman, *Deep learning from scratch*, O'Reilly Media, Inc. 2019.

[5] D. Foster, *Generative deep learning*, O'Reilly Media, Inc. 2020.

[6] Y. Tian, "zi2zi: Master chinese calligraphy with conditional adversarial networks," [Internet] https://github.com/kaonashi-tyc/zi2zi, Mar. 22. 2021.

[7] D. H. Ko, A. U. Hassan, J. Suk, and J. Choi, "SKFont: Skeleton-driven Korean font generator with conditional deep adversarial networks," *International Journal on Document Analysis and Recognition (IJDAR)*, 2021.

[8] J. Cha, S. Chun, G. Lee, B. Lee, S. Kim, and H. Lee, "Few-shot compositional font generation with dual memory," *European Conference on Computer Vision (ECCV)*, 2020.

[9] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784. 2014.

[10] D. H. Ko, H. Lee, J. Suk, A. U. Hassan, and J. Choi, "Hangul font dataset for Korean font research based on deep learning," *KIPS Transactions on Software and Data Engineering*, Vol.10, No.2, pp.73-78, 2021.

[11] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "DCFont: An end-to-end deep chinese font generation system," *SIGGRAPH Asia 2017, Technical Briefs*, 2017.

[12] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "SCFont: Structure guided Chinese font generation via deep stacked networks," *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, 2019.

[13] Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao. "Artistic glyph image synthesis via one-stage few-shot learning," *Association for Computing Machinery (ACM) Transactions on Graphics*, 2019.

[14] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim. "Few-shot font generation with localized style representations and factorization," *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[15] S. Park , S. Chun, J. Cha, B. Lee, and H. Shim, "Multiple heads are better than one: Few- shot font generation with multiple localized experts," *ArXiv abs/2104.00887*, 2021.

[16] D.P. Kingma and M. Welling, "Auto-encoding variational bayes," *Proceedings of International Conference on Learning Representations*, 2014.

[17] G. Parmar, R. Zhang, and J. Y. Zhu, "On buggy resizing libraries and surprising subtleties in FID calculation," 2021, [Internet] https://github.com/bioinf-jku/ TTUR, Sep. 2021.

박 장 경

https://orcid.org/0000-0001-6446-4590
e-mail : xjnpark@soongsil.ac.kr
1981년 공군사관학교 항공공학(학사)
1987년 미국 해군대학원 OR/SA(석사)
2019년 ~ 현 재 숭실대학교 컴퓨터공학부 박사과정
관심분야 : 딥러닝, 자동 폰트 생성, 최적화

Ammar Ul Hassan

https://orcid.org/0000-0001-6744-507X
e-mail : ammar.instantsoft@gmail.com
2013년 International Islamic University Islamabad, Pakistan 컴퓨터공학부(학사)
2018년 숭실대학교 컴퓨터공학부(석사)
2018년 ~ 현 재 숭실대학교 컴퓨터공학과 박사과정
관심분야 : 딥러닝, 자동 폰트 생성, 폰트 데이터셋

최 재 영

https://orcid.org/0000-0002-7321-9682
e-mail : choi@ssu.ac.kr
1984년 서울대학교 제어계측공학과(학사)
1986년 미국 남가주대학교 전기공학과 (컴퓨터공학)(석사)
1991년 미국 코넬대학교 전기공학부 (컴퓨터공학)(박사)
1992년 ~ 1994년 미국 국립오크리지연구소 연구원
1994년 ~ 1995년 미국 테네시 주립대학교 연구교수
1995년 ~ 현 재 숭실대학교 컴퓨터학부 교수
관심분야 : 디지털 타이포그라피, 시스템소프트웨어, 병렬/분산처리, 고성능컴퓨팅