GANdan-fontmaker: Web Service for Handwritten-Hangul Font Generation

Seongbin Yoon[1], Deukyun Nam[2], Dasol Lee[3], and Sungeun Wee[3]

[1] Sungkyunkwan University, Electronic and Electrical Engineering
[2] Sungkyunkwan University, Mathematics
[3] Sungkyunkwan University, Systems Management

Abstract. Hangul is a language that is composed of initial, medial, and final consonant. This leads to 11,172 combinations of characters in Hangul. For this reason, the current method of designing all the characters by hand is very expensive and time-consuming compared to English. We developed web-service(GANdan-fontmaker) based on Dual Memory-Augmented Font Generation Network(DM-font). Our service provides two distinctive functions; combining people's font style and re-correcting font style for satisfying user needs.

Keywords: GAN · Font Generation · Web Service

1 Introduction

These days, there is a trend that leads to customer satisfaction through personalized services. This reflects the situation in which individuals want to be expressed and respected not only on social media but also in real world. Therefore, we conducted a project to produce fonts with handwriting containing individual personality and characteristics. The goal is to overcome the time and cost limitations of existing font generation and to provide an environment in which users can easily produce their own fonts. Various trials on font generation based on Style Transfer[1] and Generative Adversarial Network(GAN)[2] have been conducted to manage the task. GAN is appropriate to generate customized fonts through few inputs, while style transfer generates fonts using whole character sets. In this project, we propose the service based on GAN font generation technique(DM-font)[3] which requires small sample of user input.

2 Related Work

2.1 Font generation models

Font generation using CNN based Style Transfer Tian, Y. (2016 [4]) proposed neural style transfer for Chinese characters, "Rewrite". The network is top-down CNN structure where font design process is formulated as a style transfer from a standard font to a target font. The network but only manages to learn one style at a time, also lots of characters (approximately 2000) are require learning style. Neural style transfer uses CNN for image style transfer. By modifying neural style transfer, Atarsaikhan et al.(2017 [5]) proposed neural font style transfer. This method transfers the style of one font image to another using the style features extracted from the intermediate layers of the CNN. In this approach, the images of various font styles of different languages, such as Arabic, Japanese, and Korean, are used to generate the font images.

GAN based font generation As generative model is advanced, lots of font generation studies are conducted using GAN framework. Hayashi et al.(2019 [6]) proposed GlyphGAN. It creates new fonts using a deep convolutional GAN (DCGAN) while maintaining style consistency over all characters. In GlyphGAN, the input vector is composed with character class vector, which is one-hot vector, and style vector, which is a uniform random vector. However, The problem is that the style of the synthesized characters cannot be controlled based on the user's preference, as the style vector is always random.

Zi2zi was proposed for generating handwritten Chinese characters[7] to solve one-to-one mapping problem on pix2pix[8]. Zi2zi adds one-hot category embedding for one-to-many modelling and includes AC-GAN architecture and a domain transfer network (DTN) to reduce category losses in multi-class. Though zi2zi can generate lots of letters simultaneously, it has a problem that the quality of the letters is poor, and the volume of learning data is too vast. To improve the model, Jiang et al.(2017 [9]) proposed DCFont: an end-to-end deep Chinese font generation system. The DCFont network consists of two networks: a network that reconstructs font features with a pretrained VGGNet for extracting a font style and a network that transmits font styles.

Font Generation in Korean Ko et al.[12] proposed SKFont: skeleton-driven Korean font generator with conditional deep adversarial networks. When given 114 target characters, the system automatically generates the rest of the characters in the same given font style by extracting the skeletons of the synthesized characters. It improves the problem of blurriness, breaking, and a lack of sophistication by using the skeleton-driven approach.

Cha et al.[3] proposed Dual Memory-Augmented Font Generation Network (DM-Font), which enables to generate a high-quality font library with only a few samples. DM-Font utilized the compositionality of Korean scripts with 68 components labels. Also, it employs dual memory structure to efficiently capture the global glyph structure and the local component-wise styles.

2.2 Font generate service

Naver "나눔손글씨" event In 2019, on the occasion of Hangul Day, Naver produced user handwriting using Clova's artificial intelligence technology and distributed 109 letters for free. Naver Clova's OCR team applied Korean handwriting font generation technology to the service to improve the recognition rate of OCR handwriting. There are the stages of recognizing handwriting in the image, analyzing and learning the characteristics of handwriting, upgrading fonts based on pre-learning models, and finally completing handwriting fonts. What user is required in this process is the 256 handwriting and 30 minutes of waiting. After the contest, the OCR team proposed a DM-font that generates a higher quality font with fewer letters. They taught the model by dividing the initial, medial, and final consonant from one letter, and understand the combination of Hangul to create the word "감" from "거" + "아" + "흠". As a result, handwriting can be created in 20 seconds even after receiving only 28 characters from the user.

Fontto The SW Maestro Twiiiks team developed Fontto with the aim of allowing the general public to create their own unique fonts and to sell and share them. When a user writes a given sentence up to five sentences, the remaining thousands of letters are automatically generated through machine learning technology. The Fontto service was created using three technologies: OCR, Variational AutoEncoder (VAE), and GAN. Based on the letters recognized as OCR and the general form of them configured through VAE, the user's font will be created using GAN technology. This service restored the handwriting of righteous person such as Ahn Jung Geun.

3 Proposed Service

3.1 Overall Architecture

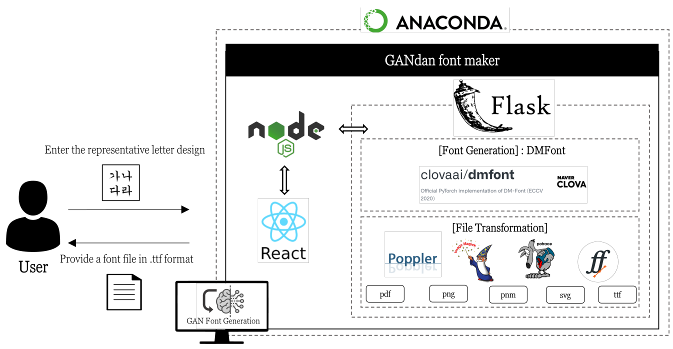


Fig. 1. Overall Architecture

3.2 Libraries and Frameworks

Front-end is developed with React.js. Web pages recently used by users are dynamic pages in which the web page structure is changed according to the user's actions. Therefore, to create a dynamic web page, it is necessary to implement a function that automatically renders the page without reloading. React fits such requirement. For example, when implementing the progress bar, the web page should handle user interaction using React's useEffect and useState. It makes easy to produce and maintain the website by reusing React components

Back-end is developed using two main technology stack. That is Node.js and Flask. Since our service is based on DM-font which is a pytorch program, we need python based web framework. Django and Flask is general option for that. We found guideline for 'when to use which framework' and we adopt Flask for out project.

When to use Flask

1. If the app is too big to fit into one codebase,
2. Or if the app is too small!
3. If anyone on the team is new to Django and Python!

We evaluate the service does not contains difficult API calls and small enough to use Flask.

3.3 DM-Font

In this section, we will explain DM-font in more detail.

Complete Compositional Scripts Compositional scripts is a glyph-rich script, where each glyph can be decomposed by several components as show in Fig. 2. A complete compositional script where each glyph can be decomposed to fixed number of sub-glyphs. For example, every Korean glyph can be decomposed by three sub-glyphs. Three sub-glyphs are each called 'cho', 'jung' and 'jong'. In Korean, three types of sub-glyph has dynamic position(See Fig. 2). This further complicates the development of a GAN by applying a bounding box to the position of each glyph. DM-font effectively solved this problem by using Dual Memory.

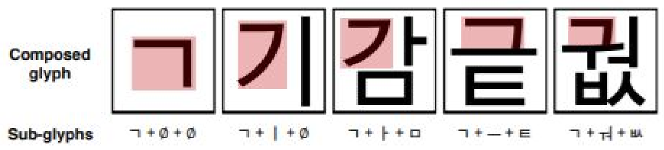


Fig. 2. Compositionality of Korean script

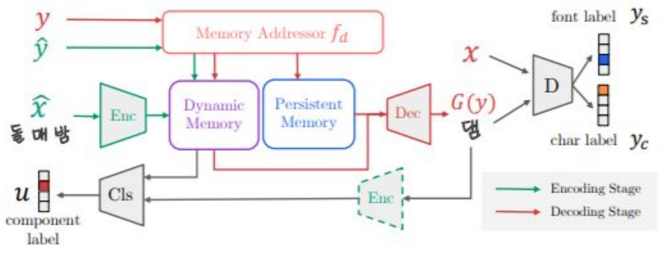


Fig. 3. DM-font overall architecture

Overall Architecture DM-font is a GAN that generates fonts for complete compositonal scripts. The overall structure is shown in Fig. 3. It consist of Generator and Discriminator. The Generator consists of Persistent Memory(PM), Dynamic memory(DM), Memory Addressor, Encoder, Decoder, and Component Classifier.

Encoding phase The encoder extracts the component-wise features and stores them into the dynamic memory using the component label u_i^c and the style label $\hat{y}_s$. Hence DM learns unique local styles depending on each font. Memory Addressor determines the location where embedded feature is to be saved. In Fig. 4, we can see that 3 times of DM access occurs in the process of encoding one Korean character. Multi-head structure is adopted for the structure of encoder. Since it adopted with one head per one component type, there are 3 heads for Korean language encoder.

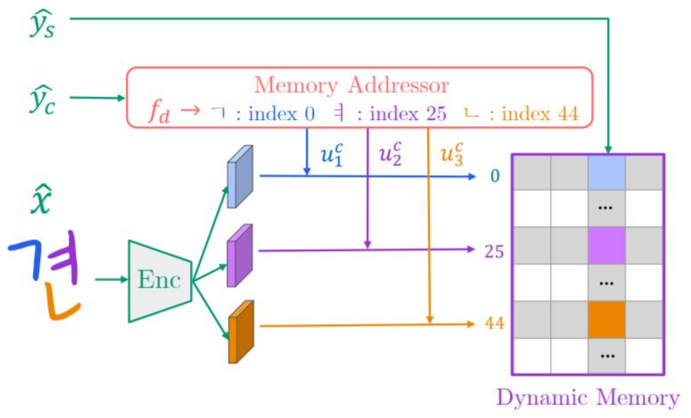


Fig. 4. Encoding phase detail

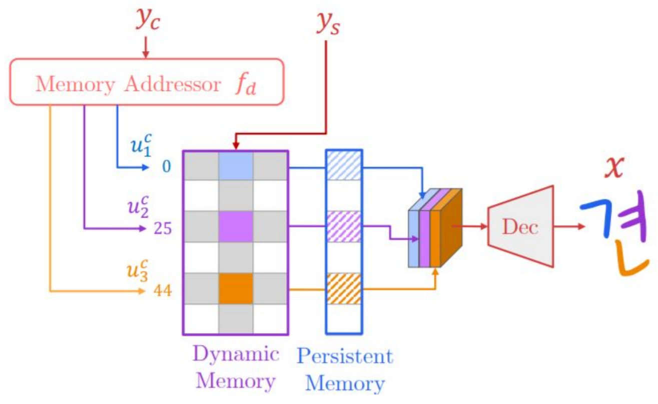


Fig. 5. Decoding phase detail

Decoding phase In decoding phase, the memory addressor loads the component features by the character label y_c and feeds them to the decoder. Both DM and PM is applied in this process. Persistent memory (PM) is a component-wise learned embedding that represents the intrinsic shape of each component and the global information of the script such as the compositionality. Therefore, the dual memory plays different roles in the decoding process. PM has global information such as glymph shape. The information includes global embeddings obtained from data. DM is responsible for simply retrieving the encoded information, but includes information such as the style of text because it distinguishes font styles.

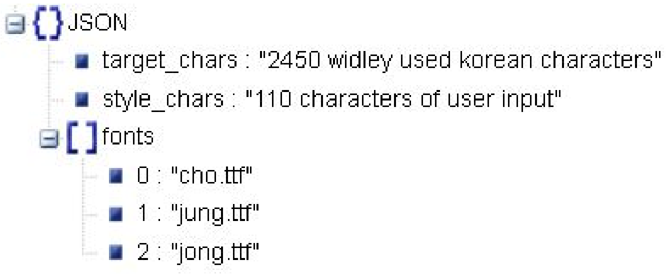
Learning

$$\underset{G,C}{min}\underset{D}{max}L_{adv(font)} + L_{adv(char)} + \lambda_{l1}L_{l1} + \lambda_{feat}L_{feat} + \lambda_{cls}L_{cls} \tag{1}$$

equation (1) is the objective function of DM-font. It consist of two **adversarial losses** on font and character, a $\mathbf{L_1}$ **loss**, a **feature matching loss** and a **component-classification loss**. Please refer to the appendix for the formula and meaning of each loss.

Implementation and environment
DM-font is implemented with pytorch, and it receives a font file as input and provides image

files for each character as output. A target character in output MUST be a combination of sub-glyphs in the character of input font file. We configure 110 characters to be default size of input font file. And configure output as 2450 characters that are most widely used.



```
{ } JSON
    target_chars : "2450 widley used korean characters"
    style_chars : "110 characters of user input"
    [ ] fonts
        0 : "cho.ttf"
        1 : "jung.ttf"
        2 : "jong.ttf"
```

Fig. 6. Configured json file for font generation : /dmfont/meta/split.json

3.4 Challenges

In this section, we explain about the challenges we faced while implementing the service and how we overcame them.

How to convert PNG to SVG Fundamentally, two types of file format conversion are required for the Backend of the service: pdf to png and png to ttf.

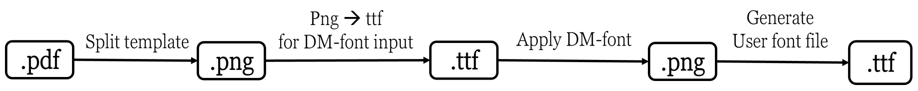


Fig. 7. required file conversion process

We could easily figure out that we need a file format called svg in the middle of the png to ttf conversion process. But we couldn't find an easy method to convert png to svg. The first method we found left a background in the converted svg, and the subsequent process did not proceed normally. Also, other known methods were mostly applicable in Linux environment. We found that we could convert png to svg in Windows environment without leaving a background via a not widely known file format called pnm. Therefore, the final necessary file conversion process was decided as shown in Fig. 8.

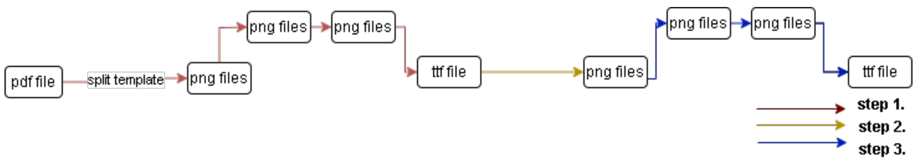


Fig. 8. final file conversion process

Here, we introduced three steps for final file conversion process. Here, we introduced three steps in file conversion.

1. pdf template → pngs files → ttf file : split template and generate font file.
2. ttf file → png files : apply DM-font
3. png files → ttf file : generate font file for user download.

More details will be dealt with in 5.2.Backend.

How to improve performance Performance improvement in font generation means the improvement of output image that are blurry or do not capture the input's style well. In the proposal phase of the project, we proposed performance improvement using Ada-Boost. At first, we thought that the Ensemble was suitable for our case referred to Boosting in GAN[13]. Like other boosting-type performance improvement methods, the idea is to increase the learning rate related to incomplete part.

However, We later realized that the meaning of bad performance in AdaGAN and DM-font are different. The referenced paper was about the generation ratio of the target labels in the unconditional GAN, not the conditional GAN like DM-font. Therefore, we investigated a method for improving the performance(quality of output) of the unconditional GAN, but it was hard to find the appropriate method rather than parameter tuning. However, parameter tuning in glyph-wise takes too much time, and also we were not confident in performance improvement.

We applied an easier approach to improve performance; increase the amount of data. We found that better performance can be obtained by applying the data-driven method on AS service.

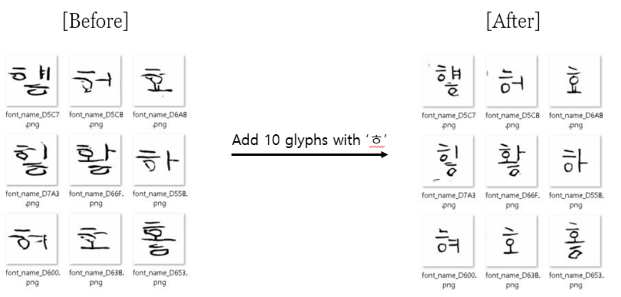


Fig. 9. AS service example

4 Design

4.1 DM-Font

We mostly used the original structure of DM-font, and added one function. That is to take handwritten-template from multiple users and mix other users' styles. To implement such a function, the structure of the encoding process has been changed as shown in Fig. 10.

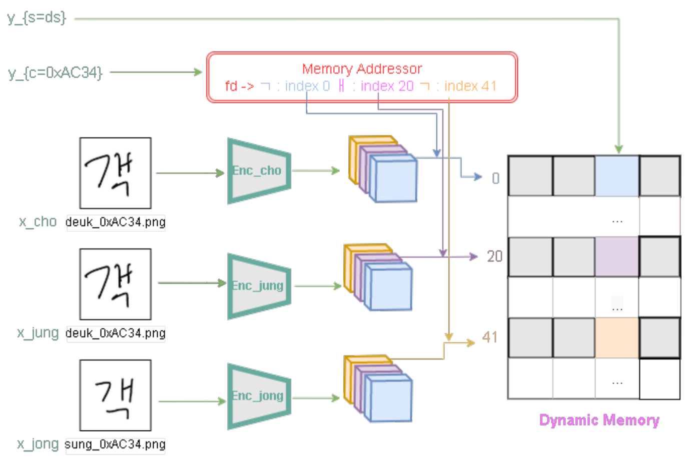


Fig. 10. modified encoding phase

We encoded a character using a encoder for each 3 component types, 'cho', 'jung', and 'jong'. The range of the target label for each encoder is the index range of the corresponding component type. By doing so, we were able to implement the function to mix the fonts of multiple users. Fig. 11. is a example of font mixing. Middle font is generated with the embedding of initial from Sengbin's font and medial and final from Hynenni's font. As you can see, the middle font represents both users style well as in configured consonant.

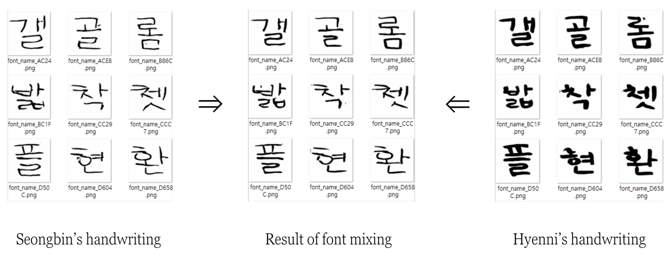
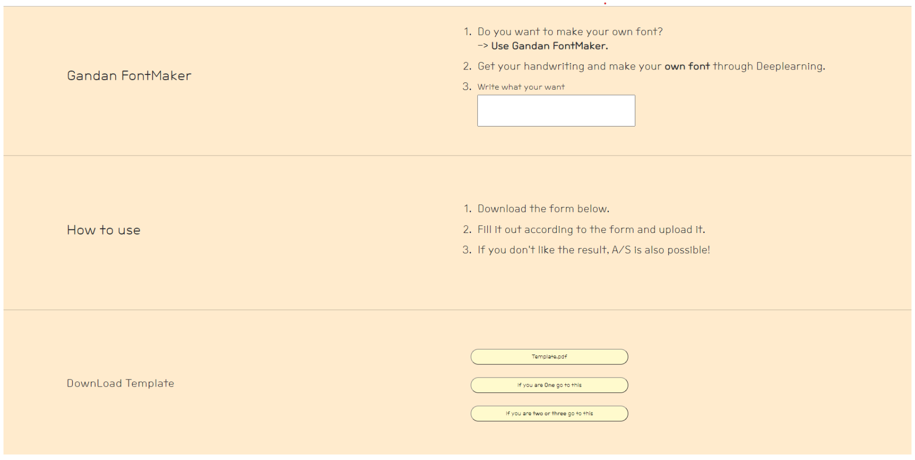


Fig. 11. Font mixing example

4.2 UI/UX

The web page consists of six pages; Main page, Singleuser page, Multiuser page, Progress page, As page, Download page.

Main page("/") The Main page was divided into three parts by adding a horizontal line for user to understand the contents of the page at a glance.



Gandan FontMaker

1. Do you want to make your own font?
   -> **Use Gandan FontMaker.**
2. Get your handwriting and make your **own font** through Deeplearning.
3. Write what your want

How to use

1. Download the form below.
2. Fill it out according to the form and upload it.
3. If you don't like the result, A/S is also possible!

DownLoad Template

Fig. 12. Main page

1. GANDAN Fontmaker has a brief explanation and the user can use an example of the font generated by the model.
2. Explain how the user can use the our website's services.
3. The function of downloading a template that allows users to fill their handwriting, page movement when there is one user, and page movement when there are many users are implemented in button format. The user can download a template that fills the user's handwriting and click one of two buttons according to user's number.

Single-user page("/one") If the user upload and submit font's name, email, and template, the user moves to the Progress page. If the user didn't upload the file, the alert that 'please upload the file' under the submit button will appear without going to the Progress page.

Multi-user page("/multi") It is a page for mixing handwriting styles of multiple users. We additionally implemented several checkboxs to reflect each user's style by receiving initial, medial, and final consonants and number of users.

Progress page("/user/:id") The user can view a progress bar configured in three steps for the progress of font generation. At the end of step 2, the Progress page displays various letters written in generated fonts on the screen so that the user can see them and decide by pressing the button whether to receive AS service or not. When the user presses the AS button, the user goes to the AS page, otherwise the Progress page will proceed with the step3, and when it ends, a link to download page will be created.

AS page("/as") In this page, the user can click several checkboxs which part of the generated font the user wants to fix, and press the button below that shows which letter to write additionally. Then the user modify and upload the pdf and press the submit button to the Progress page.

Download page("/download/:id") The user can download generated font in ttf format by clicking the download link.

5 Implementation

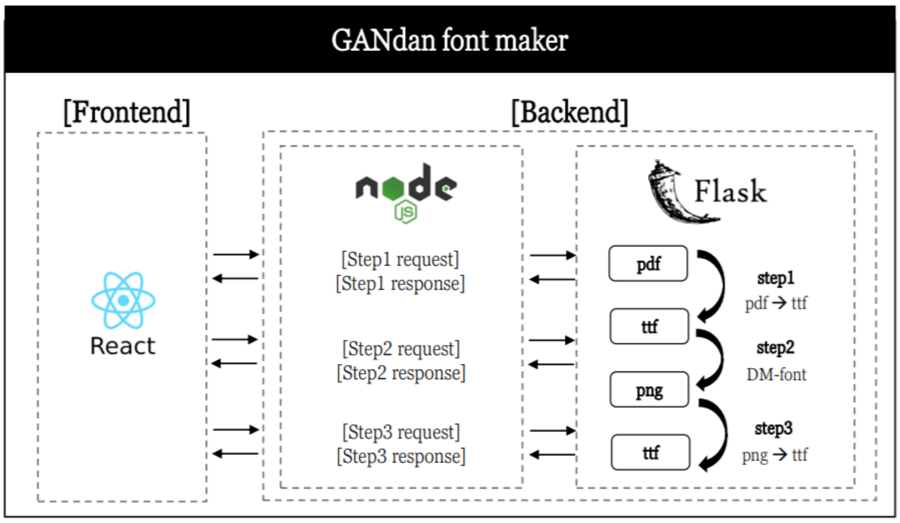


Fig. 13. Server Communications

5.1 Frontend

Send the user information React uses port 3000 and interacts with Backend using the Axios library.

1. Text We used React's useState to store user's information such as the number of users, font's name, email and consonant. When the user presses the submit button, the information is sent to Node.js server in json format.
2. File In the case of file sending, files are stored in an object called FormData.

Progress-bar Implementation When the user moves to the Progress page, an Axios regarding step1 is executed. When it receives the call meaning that step1 is finished, the step progresses and the variable related to the progress bar is increased. Accordingly, the Progress page is newly rendered, and then the Axios related to the step2 is executed. To implement this, we used the React's useEffect. When step2 is completed, the several images are generated and button to choose whether AS or not is shown. If the user doesn't choose AS, the Axios related to the step3 is executed. When step3 is completed, the progress-bar is filled, and the download link is created as the variable showing the link to the download page changes from false to true.

download user's generated font After communicating with Node.js through Axios, the download page will receive the font name from Node.js and use it for the user to download the user's generated font.

5.2 Backend

Backend is implemented using Node.js using port 8080 and Flask using port 5000. Node.js connects between React and Flask and Flask receives information from nodejs and executes its step. We use Express.js to facilitate implementation of Node.js.

Store the user's information

1. Text When Node.js receives the information from React, text information in form of json is stored and sent to Flask without any processing through Axios.
2. File File store requires additional process because it is different from the text. We use multer which is an Express library to modify received file in FormData. Next, upload function of multer stores the files in public folder of React.

As discussed in 3.4.'How to convert PNG to SVG', the operation of Flask Backend is divided into 3 steps. That is

1. pdf template → pngs files → ttf file : split template and generate font file.
2. ttf file → png files : apply DM-font
3. png files → ttf file : generate font file for user download.

We used many open sources for file format conversion. Table. 1. is the list of file conversion and open source that we used. Flask sends a response to Node.js whenever each step is finished.

Conversion	Used Open Source	Related File, Class	Used steps
pdf->png	pdf2image	Template.py	1
png->pnm	ImageMagick	Png2Svg.py	1,3
pnm->svg	potrace	Png2Svg.py	1,3
svg->ttf	FontForge	Svg2ttf.py	1,3
ttf->png	DM-font	/dmfont/*	2

Table 1. File Conversion Details

6 Limitation

6.1 Assumptions

Validation GANdan-fontmaker will interact with users with templates, and we assume that users strictly follow the guidelines we suggest. We check the existence of a input, not the validity of the input. User may behave incorrectly such as writing different letters on template, or uploading invalid file.

6.2 Constraints and Limitations

AS in mixed font However, according to the modified DM-font structure, each font shape is stored by dividing the initial, medial, and final consonants, but the number of inputs for initial, medial, and final consonants could not be different. For example, if the initial consonant part needs to be modified, the inputs of not only initial, but also medial and final parts are required. It did not fit the purpose of the AS service as we planned. Alternatively, we tried to input the empty template, but an error occurred in DM-font stage due to empty character in ttf file. But this problem was not solved within the project period also.

7 Evaluation

In fact, the satisfaction and assessment of the generated font depends on individual taste. Considering it as a subjective area, it is hard to set a objective evaluation criteria and formulas. As an alternative, user satisfaction evaluation was conducted through a survey.

User satisfaction Survey Through Everytime, Sungkyunkwan University community service, we recruited students who wish to participate in this project. A total of 19 people participated and responded to the survey. The results of this survey are as follows.

1. The amount of handwriting requested from the user

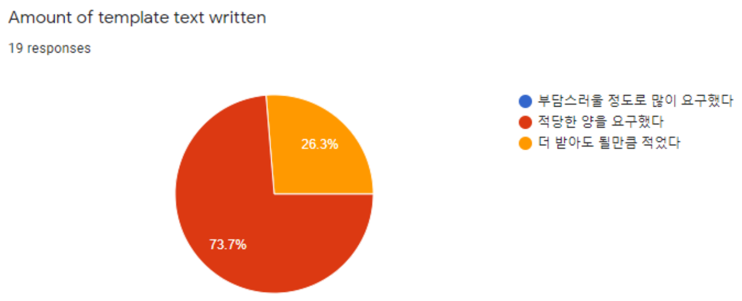


Fig. 14. the result of survey question 1

The users have to write 110 Korean letters and submit it to generate a font using GANdan fontmaker. Although it performs better as the amount of input data increases, it is important to select the appropriate amount of letters that satisfy user experience without burdensome. As shown on Fig. 14 above, 26.3% said the amount of writing was little enough, and 73.7% said it was suitable. Nobody answered it was burdensome, indicating that our project asked users for a proper amount of handwriting.

2. Satisfaction of users with the font
   The users choose one of the numbers from 1 to 5 according to the satisfaction of the generated font. As shown on Fig. 15 above, 26.3% showed the highest satisfaction, and 63.1% of the total evaluated positively by choosing 4 and 5. Although 21% were dissatisfied with the font's design, more than half were satisfied with the result of GANdan fontmaker.

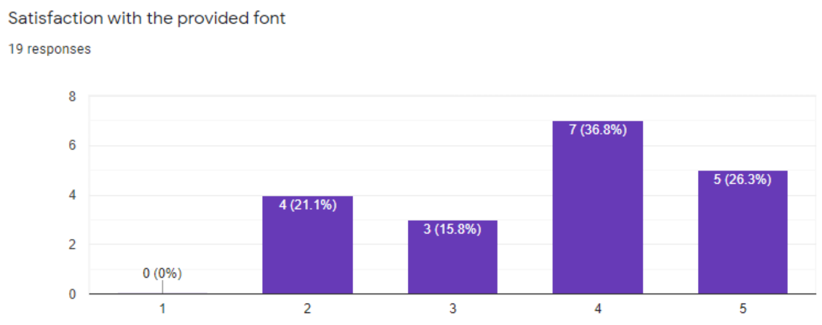


Fig. 15. the result of survey question 2

3. Users' review
   We asked users for personal opinions on font production, and received lots of answers. There was a review that pointed out the flaws and disappointment in the generated font. In addition, there was an answer that was satisfied with the result and was happy with the experience of producing own fonts. In general, users responded positively to the fonts and these kind of new experiences. Through the user satisfaction evaluation, we confirmed that the project has been successfully carried out.

8 Conclusion

Our project utilizes DM-Font model specialized for the comprehensive characteristics of Korean and develops a website that can provide font production services. Given the conventional way of making fonts, it is time-consuming and expensive to produce every single 2,448 letters or more by humans. In addition, edg services that produce fonts based on AI are not only limited, but also in high costs. Therefore, GANdan fontmaker allows users to easily produce fonts for free. Moreover, the website was developed so that users could intuitively access to the service. Users can obtain the only font in the world in about ten minutes by providing 110 handwriting letters.

Through a personalized font production project, we have developed a program that can provide convenience and draw attention from users. It increased the user's enjoyment either by producing fonts alone or by mixing handwriting characteristics of multiple users. It is a highly accessible project that even people who do not know artificial intelligence can participate in deep learning projects with fun. As can be seen from the user satisfaction survey, it provided results that can elicit user interest and satisfaction. It positively increased the user's usage experience through AS service.

References

1. Gantugs Atarsaikhan, et al. ”Neural Font Style Transfer.” International Conference on Document Analysis and Recognition (IAPR) (2017).
2. A. Odena, C. Olah, and J. Shlens, “Conditional Image Synthesis with Auxiliary Classifier GANs,” Proceeding of the 34th International Conference on Machine Learning, ArXiv Preprint arXiv:1610.09585, 2017.
3. Cha, Junbum, et al. ”Few-shot compositional font generation with dual memory.” Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16. Springer International Publishing, 2020.
4. Tian, Yuchen. ”Rewrite: Neural style transfer for chinese fonts, 2016.” Retrieved Nov 23 (2016): 2016.
5. Atarsaikhan, Gantugs, et al. ”Neural font style transfer.” 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). Vol. 5. IEEE, 2017.
6. Hayashi, Hideaki, Kohtaro Abe, and Seiichi Uchida. ”GlyphGAN: Style-consistent font generation based on generative adversarial networks.” Knowledge-Based Systems 186 (2019): 104927.
7. Tian, Yuchen. ”zi2zi: Master chinese calligraphy with conditional adversarial networks.” Internet] https://github. com/kaonashi-tyc/zi2zi (2017).
8. Isola, Phillip, et al. ”Image-to-image translation with conditional adversarial networks.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
9. Jiang, Yue, et al. ”Dcfont: an end-to-end deep chinese font generation system.” SIGGRAPH Asia 2017 Technical Briefs. 2017. 1-4.
10. Freund, Yoav, Robert Schapire, and Naoki Abe. ”A short introduction to boosting.” Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.
11. Tolstikhin, Ilya, et al. ”Adagan: Boosting generative models.” arXiv preprint arXiv:1701.02386 (2017).
12. Ko, Debbie Honghee, et al. ”SKFont: skeleton-driven Korean font generator with conditional deep adversarial networks.” International Journal on Document Analysis and Recognition (IJDAR) (2021): 1-13.
13. Tolstikhin, Ilya, et al. ”Adagan: Boosting generative models.” arXiv preprint arXiv:1701.02386 (2017).

Appendix. A. learning
Data set for DM-font is font sets (x, y_c, y_f) ˜ D, where x is a target glyph image, y_c and y_f is a character and font label, respectively.

Adversarial loss helps the model generate plausible images.

$$L_{adv} = \mathbb{E}_{x,y}[logD_y(x))] + \mathbb{E}_{x,y}[log(1 - D_y(G(y_c, y_f)))] \tag{2}$$

There are two types of adversarial $L_{adv(font)}$ and $L_{adv(char)}$.

L_1 **loss** adds supervision from the ground truth target x.

$$L_1 = \mathbb{E}_{x,y}[||x - G(y_c, y_f)||]_1 \tag{3}$$

Feature matching loss helps the stability of the training.

$$L_{feat} = \mathbb{E}_{x,y}[\frac{1}{L}\sum_{l=1}^{L}||D_f^{(l)}(x) - D_f^{(l)}(G(y_c, y_f)))||_1] \tag{4}$$

It stabilize training by considering layer-wise output of discriminator.

Lastly, **component-classification loss** is used to fully utilize the compositionality.

$$L_{cls} = \mathbb{E}_{x,y}[\sum_{u_i^c \in f_d(y_c)} CE(Enc_i(x), u_i^c))] + \mathbb{E}_y[\sum_{u_i^c \in f_d(y_c)} CE(Enc_i(G(y_c, y_f)), u_i^c))] \tag{5}$$

Component-classification loss train Component classifier C with extracted component-wise features $(Enc_i(x))$ using cross-entropy loss(CE). Here u_i^c is the component labels for given character label y_c.

The final objective function to optimize the generator G, the discriminator D, and the component classifier C is defined as the sum of above losses :

$$\min_{G,C}\max_{D} L_{adv(font)} + L_{adv(char)} + \lambda_{l1}L_{l1} + \lambda_{feat}L_{feat} + \lambda_{cls}L_{cls} \tag{1}$$

where λ_1, λ_{feat} , λ_{cls} are control parameters to importance of each loss function. The default is $\lambda_1 = 0.1$, $\lambda_{feat} = 1.0$, $\lambda_{cls} = 0.1$.