Pusan National University
Computer Science and Engineering
2020 전기 졸업과제 최종보고서
2020-09-21

생성적 적대 신경망(GAN) 머신러닝을 활용한 글의 이미지화

분과 A(소프트웨어, 인공지능)
팀 번호 11
팀 명 GAN 다 박창조
지도교수 우 균

정보컴퓨터공학전공 201524461 박 성국
정보컴퓨터공학전공 201724557 장 수현
정보컴퓨터공학전공 201724480 박 창조

목 차

1. 과제 개요

1.1 과제 배경

과제를 선정한 배경은 다음과 같다. 그림 1, 그림 2는 인터넷 커뮤니티의 특정 댓글들을 캡처한 것이다. '3줄 요약'과 '길어서 패스'라는 말은 긴 글을 읽기 싫어하는 사람들이 자주 하는 말이다. 실제 긴 글을 읽고 이해하는 것이 누구에게나 쉬운 일은 아니다[1]. 2017년 국가평생교육진흥원에서 실시한 '성인 문해 능력 조사'[2](18세 이상 성인 남녀 4,004명 대상) 결과 약 22.4%로 성인 5명 중 1명이 미흡한 문해력을 가졌다고 보고했다. 글을 몰랐던 문맹 시대가 끝났지만 새로운 문맹 시대가 된 것이다. 이처럼 낮은 이해력을 가진 난독 사회에서는 '세 줄 요약'이 기본 예의가 되었다. 심지어 네이버(검색 엔진)는 기사의 핵심만 간추려 보여주는 '요약봇'이라는 인공지능(AI)까지 도입했다.

그림 1. 커뮤니티 '3줄 요약' 댓글 캡처 화면

그림 2. 커뮤니티 '길어서 패스' 댓글 캡처 화면

문화체육관광부에서 조사한 2019 국민 독서실태 조사[3]에 따르면, 그림3과 같이 성인과 학생의 독서율이 해마다 점점 감소하고 있다. 그리고 책을 읽기 어려운 이유로 성인의 경우, '책 이외의 다른 콘텐츠 이용'이 약 29.1%로 가장 많았다. 특히 19~29세 성인들의 책 이외의 다른 콘텐츠인 만화책과 웹툰 이용률이 81.7%에 달했고, 초·중·고 학생들의 만화 이용률은 약 70%에 이르는 것으로 조사되었다.

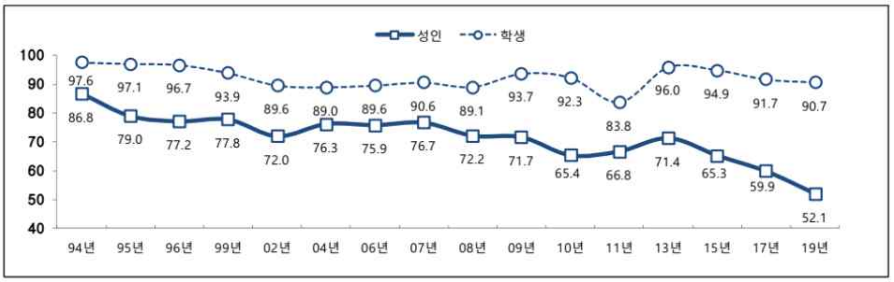


그림 3. 2019 국민 독서실태 조사에 따른 '종이책 독서율' 변화 추이(성인·학생)

영어 관용어 중 'A picture paints a thousand words.' 라는 말이 있다. 그림 한 장이 천 마디 말을 그린다는 뜻으로, 우리나라 속담 '백문이 불여일견'과 유사한 의미이다. 긴 설명과 단어들보다 그림이나 이미지와 같은 시각적인 자료가 더 중요하다는 것이다. 이것은 SNS 시장에서도 증명되었는데, 텍스트 위주의 트위터나 페이스북 같은 SNS를 이미지 위주의 인스타그램, 핀터레스트 같은 SNS가 대체해가며 급성장하였다. 결론적으로 현재 일부 세대에게는 텍스트보다는 이미지가 더 선호되고 있고, 이런 '텍스트 혐오증'에 걸린 디지털 세대를 위해 "글의 이미지화"라는 주제를 선정하게 되었다.

1.2 기존 사례 및 문제점

조사한 바로는 현재 존재하는 시스템 중 우리 과제와 완전히 부합하는 인터페이스를 제공하는 시스템은 존재하지 않는다. 다만, 관련된 기술들을 조사해보니 과제 시스템 일부에 해당하는 이미지 생성과 이미지 화풍 변환에 관한 기술들을 찾을 수 있었다.

우선, 이미지 생성에 관한 사례이다. 그림으로부터 텍스트를 추출하는 기술들은 높은 수준으로까지 발전해있다. 그러나 반대로 텍스트로부터 이미지를 생성해내는 기술은 아직 어려운 기술로 받아들여지고 있다. 사람이 텍스트를 보고 이미지를 머릿속에서 생각해내는 것과 비교하면 아주 낮은 수준에서만이 구현이 되어있다. 주로 생성적 적대 신경망(GAN[4]: Generative Adversarial Network)으로 구현이 되어있으며 CPGAN, DM-GAN, OP-GAN, MirrorGAN, StackGAN 등 종류가 다양하다. 그중에서도 우리는 AttnGAN[5]에 관심을 가지고 살펴보았다.

AttnGAN은 다른 GAN들과 비교했을 때, 더 오래되어 안정성이나 관련된 정보가 더 있었다. 그리고 생성모델의 어느 정도 객관성을 보장해주는 성능 측정 지표인 Inception score가 비교적 높은 편에 속했다. 이 AttnGAN는 Microsoft AI 연구실에서 개발된 Drawing Bot[6]이라는 가상 이미지 생성 인공지능을 지원한다. 그림 4는 Drawing Bot이 생성한 가상 이미지이다.

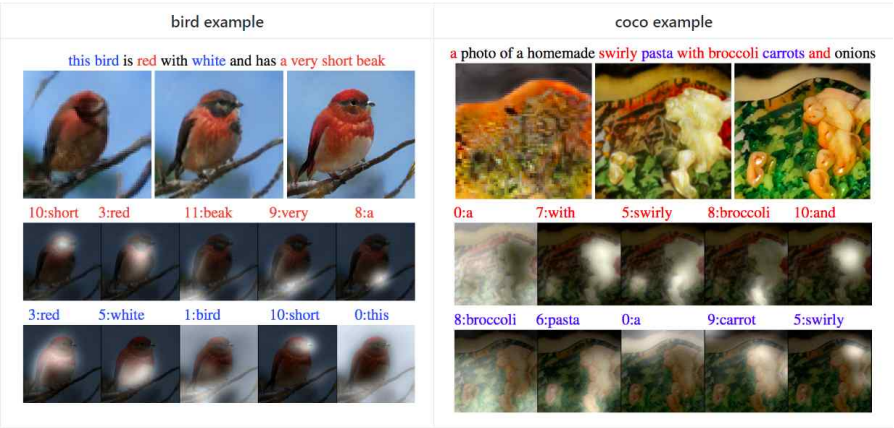


그림 4. Drawing Bot이 생성한 이미지

그러나 Drawing Bot은 단순히 Git에 코드만 제공할 뿐, 사용자가 실제 사용할 수 있도록 인터페이스를 제공하지는 않는다. 그리고 이미지 생성에 필요한 입력 텍스트로 영어만을 사용하기 때문에 한글 입력 텍스트에 대해서는 이미지 생성이 불가능하다. AttnGAN이 이미지에 대한 영어 텍스트로만 학습되었기 때문이고, AttnGAN뿐만 아니라 다른 모든 GAN에서도 마찬가지로 한글 입력을 통한 이미지 생성이 불가능하다. 그리고 사용자가 원하는 화풍으로 이미지 변환 또한 불가능하다.

그래서 두 번째로 알아본 것이 이미지 화풍 변환에 관한 사례이다. 기존의 이미지(혹은 영상)를 다른 화풍의 이미지(혹은 영상)로 변환하는 기술에도 다양한 머신러닝 알고리즘이 있고, 대표적인 예가 pix2pix[7]와 CycleGAN[8]이다. 그중에서 우리는 데이터 수집에 어려움이 있으므로, Unpaired Data Set으로부터도 높은 품질의 결과를 얻을 수 있는 CycleGAN에 집중했다.

CycleGAN 역시 Git에 코드를 제공하긴 하지만 사용자가 실제 이미지 변환을 할 수 있도록 인터페이스를 제공하고 있지는 않다. 그리고 우리 과제에서 이미지 변환 부분을 담당하겠지만 이미지 생성까지는 해주지 못한다. 표 1은 AttnGAN, CycleGAN, 과제 시스템의 차이점을 보여주는 비교표이다.

표 1. AttnGAN, CycleGAN, 과제 시스템의 비교표

	AttnGAN	CycleGAN	과제 시스템
인터페이스 제공	X	X	O
한글 사용 가능	X	무관	O
이미지 생성	O	X	O
화풍 적용	X	O	O

1.3 과제 목표

글의 이미지화라는 과제 시스템의 목표를 다음과 같이 설정했다.

1.3.1 인터페이스 제공

사용자가 원하는 문장을 직접 입력할 수 있어야 하고, 사용자가 입력한 문장으로부터 생성된 이미지와 그 이미지에 화풍 적용이 된 이미지 결과를 볼 수 있도록 웹 인터페이스를 제공해야 한다.

1.3.2 한글 사용 가능

사용자가 한글 문장을 입력했을 때 입력한 문장으로부터 추출된 객체에 대응하는 이미지 결과를 확인할 수 있어야 한다.

1.3.3 이미지 생성 및 화풍 적용

한 시스템 내부에서 이미지 생성과 생성된 이미지에 대한 화풍 적용이 모두 이루어질 수 있어야 한다.

2. 과제 배경 지식

2.1 GAN(Generative Adversarial Network)

GAN이란 '생성적 적대 신경망'의 약자로, 생성자와 식별자가 서로 경쟁(Adversarial)하며 데이터를 생성(Generative)하는 모델(Network)을 뜻한다. 흔히 GAN의 개념을 설명할 때 지폐위조범과 경찰에 비유하는데, 진짜와 구별할 수 없는 가짜 지폐를 만들어내는 지폐위조범과 가짜 지폐를 가짜, 진짜 지폐를 진짜라고 구별하는 경찰의 관계로 표현할 수 있다. 여기서 지폐위조범은 GAN에서의 생성자(Generator), 경찰은 식별자(Discriminator)에 대응된다. 예를 들어, GAN으로 인물 사진을 생성해내도록 학습한다면 생성자는 인물 사진을 만들어내고, 경찰은 만들어진 인물 사진을 식별한 후 이에 대한 피드백으로 생성자는 좀 더 진짜에 가까운 인물 사진을 만들어낼 수 있게 된다.

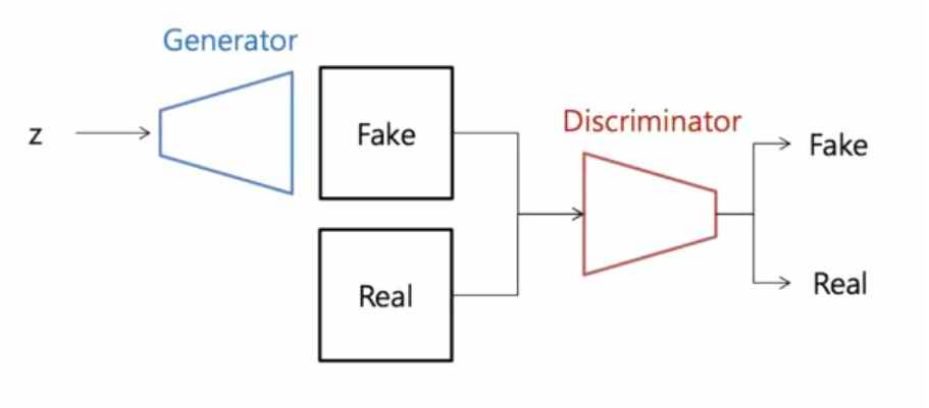


그림 5. GAN의 구조

그림 5에서 생성자는 입력 데이터의 분포를 알아내도록 학습한다. 이 분포를 재현하여 원 데이터의 분포와 차이가 없도록 하고 식별자는 진짜 데이터인지 가짜 데이터인지 구별해서 각각에 대한 확률을 추정한다. 만약 생성자가 정확히 입력 데이터의 분포를 표현할 수 있다면 거기서 생성한 표본은 실제 데이터와 구별이 불가능할 것이다. 그리고 식별자는 현재 데이터의 표본이 진짜 데이터인지, 아니면 생성자로부터 만들어진 것인지 구별해서 각각에 대한 확률을 평가한다.

생성자의 궁극적인 목적은 식별자가 거짓으로 판별할 수 없을 정도로 실제 데이터의 분포에 가까운 데이터를 생성하는 것이며, 식별자의 궁극적인 목적은 가짜와 진짜를 정확하게 구분해내는 것이다. 이렇게 생성자와 식별자는 서로 적대하며 성능이 점차 개선되고 궁극적으로는 실제에 가까운 데이터를 생성해낼 수 있게 된다.

2.2 AttnGAN(Attentional GAN)

AttnGAN은 StackGAN에 추가로 Attention 메커니즘을 도입하여 여러 단계에 걸쳐 더 섬세한 text-to-image translation을 가능하게 한 알고리즘이다. AttnGAN은 크게 두 가지 구조로 구분되는데, 첫 번째는 Deep Attentional Multimodal Similarity Model(이하 DAMSM)이다. 이 DAMSM을 통해 이미지 정보와 텍스트 정보가 잘 매칭되는지 확인하기 위해 단어 단위로 손실을 측정하여 이미지 생성을 돕는다.

두 번째는 Attentional Generative Network(이하 Attn)이다. Attn에서는 global sentence를 통해 전체적인 부분을 low-resolution 이미지로 생성하고 이후의 생성자로 개별 단어를 통해 sub region image를 생성하여 섬세한 이미지로 Upsampling한다.

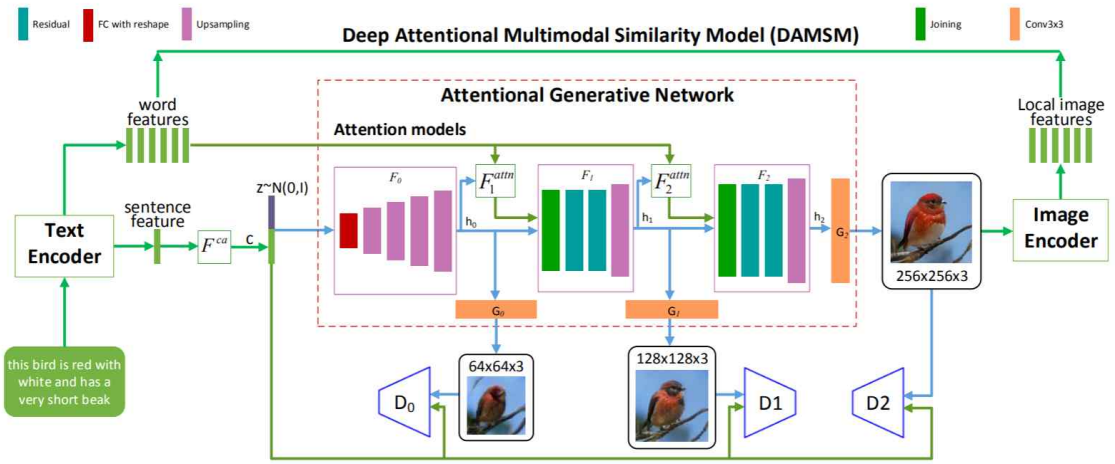


그림 6. AttnGAN의 구조도

그림 6은 AttnGAN의 구조도를 나타내고 있는데, 중앙의 Attentional Generative Network 부분을 보면 여러 개의 생성자(G0, G1, G2)와 여러 개의 식별자(D0, D1, D2)가 있다. 각각의 생성자는 저해상도에서부터 고해상도의 이미지를 생성하고, 식별자 또한 생성자가 생성한 이미지를 식별한다. G0 생성자는 초기 가장 낮은 해상도의 이미지를 생성하고, 이후 G1와 G2로 단어 단위로 섬세한 이미지를 생성한다.

Attentional Generative Network를 제외한 나머지 부분은 DAMSM에 해당한다. DAMSM에서는 Sub region의 이미지와 단어를 학습하여 그 둘을 사상한다. 이를 통해 image-text 유사도를 문장 수준이 아니라 단어 수준에서 비교할 수 있게 된다. 더불어 손실 역시 더 섬세한 부분을 고려할 수 있게 된다.

2.3 CycleGAN

CycleGAN은 image-to-image translation 알고리즘의 한 종류로, CycleGAN 이전의 image-to-image translation은 Paired image-to-image translation이었다. 하지만 Paired image-to-image translation의 경우 학습을 위해서는 원본 이미지와 해당 이미지와 짝이 되는 변환된 이미지가 필요하다는 단점이 있다.

그림 7. pix2pix를 이용한 흑백-컬러 영상 변환

그림 7은 pix2pix를 이용하여 흑백 영상을 컬러 영상으로 변환한 것인데, 학습을 위해서는 어떤 색채 이미지와 해당 이미지를 흑백으로 변환한 이미지를 짝을 지어줘야 한다는 것이다. 흑백-색채 변환에서 이는 그다지 어렵지 않다. 색채 이미지를 먼저 구한 뒤, 해당 이미지를 흑백으로 변환하는 것은 어렵지 않기 때문이다.

하지만 예를 들어 사진을 모네의 그림으로 변환하고 싶다고 하면, 사진과 해당 사진을 모네가 따라 그린 그림이 있어야 한다는 것인데, 이것은 사실상 불가능하다. 마찬가지로 세상에 있는 모든 이미지에 대해서 변환의 결과가 되는 이미지를 짝짓는다는 것 역시 불가능하다는 것이다.

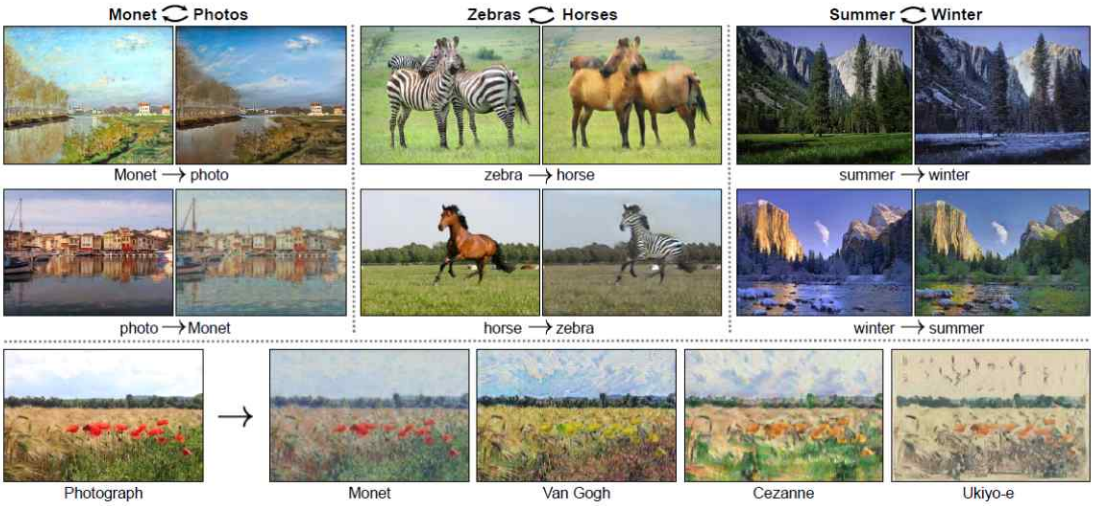


그림 8. CycleGAN으로 변환한 이미지와 원본 이미지

이러한 단점을 해결해주는 것이 CycleGAN이다. 그림 8은 원본 이미지로부터 CycleGAN으로 변환한 이미지를 보여준다. CycleGAN은 Unpaired image-to-image translation 알고리즘을 사용한다. 이는 짝지어진 예시 없이 X라는 도메인으로부터 얻은 이미지를 목표 도메인 Y로 바꾸는 것이 가능하다는 것이다.

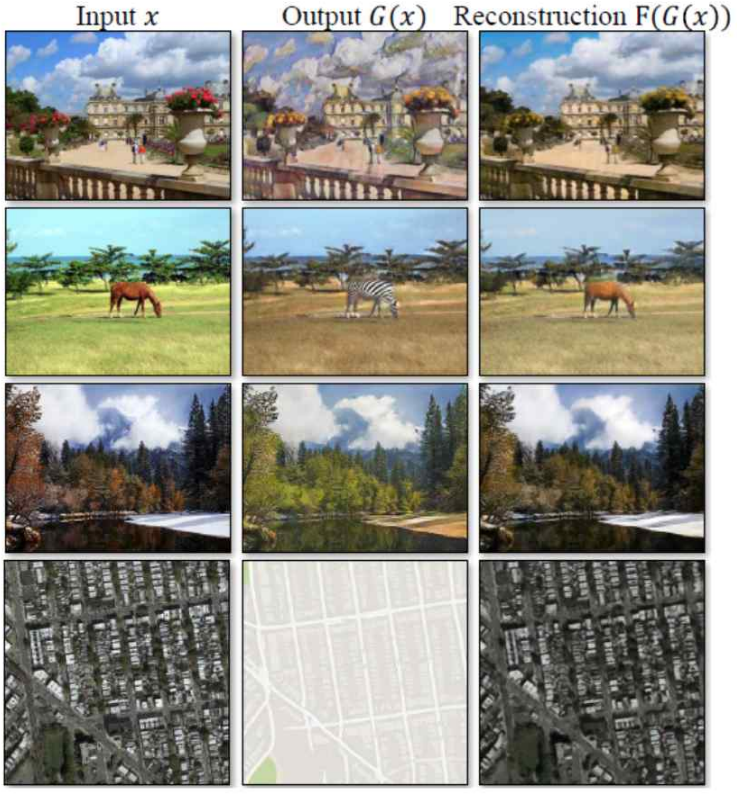


그림 9. 원본이미지 x를 변환한 G(x)와 해당 이미지를 다시 변환한 F(G(x))를 나타낸 그림. x와 F(G(x))가 꽤 유사함을 알 수 있다.

이를 위한 CycleGAN만의 특징으로 역방향 대응과 Cycle Consistency Loss가 있다. 역방향 대응이란 사진을 모네의 그림으로 변환하는 것을 학습할 때, 해당 변환뿐만 아니라 모네의 그림에서 사진으로 변환하는 것을 동시에 학습하는 것을 말한다. 이 과정에서 Cycle Consistency Loss를 측정하는데, 이것은 쉽게 비유하자면 한국어에서 영어로 번역한 문장을 다시 한국어로 번역했을 때 원래의 한국어 문장과 얼마나 일치하는지를 측정하는 것이다. 사진-모네의 그림 변환에서도 사진을 모네의 그림으로 변환했다가 다시 사진으로 변환했을 때, 학습이 성공적으로 진행되었다면 원본 사진과 비슷한 결과물이 나와야 한다는 것이다. 그림 9를 보면 원본 이미지 x를 변환한 G(x)와 해당 이미지를 다시 변환한 F(G(x))를 나타낸 그림. x와 F(G(x))가 꽤 유사함을 알 수 있다. 이러한 두 가지 특징 덕분에 CycleGAN은 다른 image-to-image translation들보다 우수한 이미지 생성이 가능하다.

3. 과제 수행 내용

3.1 설계 및 수정사항

3.1.1 초기 설계 및 문제점

초기에는 한 권의 소설을 한 권의 그림책으로 만들려고 했었다. 그림 10의 초기 시스템의 개요를 살펴보면, 1단계에 해당하는 소설로부터 핵심적인 내용의 텍스트를 요약 추출한다. 그리고 추출된 텍스트별로 각각 2단계에서 이미지 생성이 진행된다. 3단계에서는 생성된 이미지들에 일관된 하나의 화풍을 적용한 뒤, 합치는 과정을 거쳐 최종적으로 하나의 그림책이 완성된다.

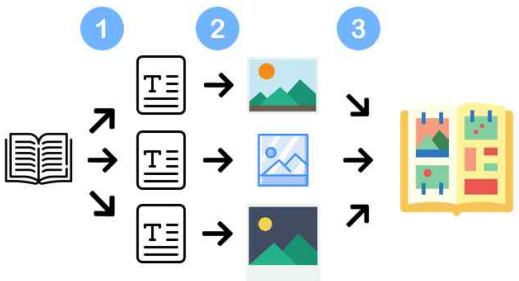


그림 10. 초기 시스템 개요도

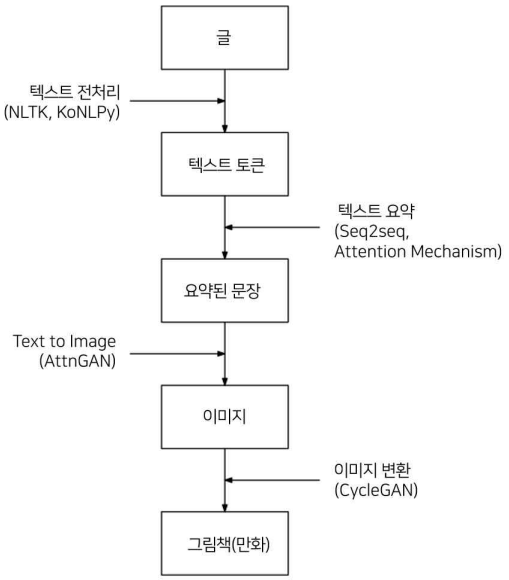


그림 11. 초기 시스템 흐름도

그림 11은 초기 시스템의 흐름도이고 각 단계에서 텍스트 전처리 및 텍스트 요약, 이미지 생성, 화풍 적용을 하는 시스템을 설계했었다. 그러나 소설 한 권을 텍스트로 요약하는 부분이 이미 큰 하나의 과제였고, 과하다고 판단되어서 이 시스템에서는 제외하기는 것으로 결정했다. 그리고 이미지 생성 부분에서 문맥에 부합하는 이미지를 생성하기에는 술어 중심의 데이터 수집과 학습이 이뤄져야 하는데 이미 데이터 수집에서도 어려움이 있고 술어를 선정하는 것에도 한계가 있었다. 그래서 과제의 완성도를 위해 문맥에 맞는 이미지 생성은 향후 발전을 위한 과제로 정했다.

3.1.2 개선된 설계

최종적으로 수정된 시스템은 객체가 묘사된 문장을 입력하면 해당 입력으로부터 이미지가 생성되고, 생성된 이미지에 화풍을 적용하는 것이다. 예를 들어 '귀가 크고, 눈과 코가 까맣고, 이마와 꼬리는 갈색 털을, 배는 하얀 털을 가진 여우가 있다.'라는 문장을 입력으로 넣으면 출력 1로는 생성된 이미지 그림 12가, 출력 2로는 화풍이 적용된 이미지 그림 13이 출력된다.

그림 12. 생성됐다고 가정한 이미지

그림 13. 동화풍이 적용된 이미지

그림 14는 최종 시스템의 설계 DFD이다. 첫 단계로 웹을 통해 클라이언트가 문장을 입력하면 해당 문장을 NLTK[9]와 KoNLPy[10]를 이용하여 토큰화한다. 두 번째 단계로 토큰화를 거친 문장은 AttnGAN에서 기존에 학습 모델을 통해 토큰에 대응되는 이미지로 변환된다. AttnGAN에서 생성된 이미지는 다시 CycleGAN의 입력으로 사용되며, 동시에 중간과정을 보여주기 위해 클라이언트의 화면에도 출력된다. 마지막 단계로 CycleGAN에서는 입력으로 들어온 이미지를 사용자가 선택한 화풍에 따라 적절한 모델을 적재하여 이미지에 화풍을 적용해준다. 화풍이 적용된 이미지는 최종적으로 클라이언트의 화면에 출력된다.

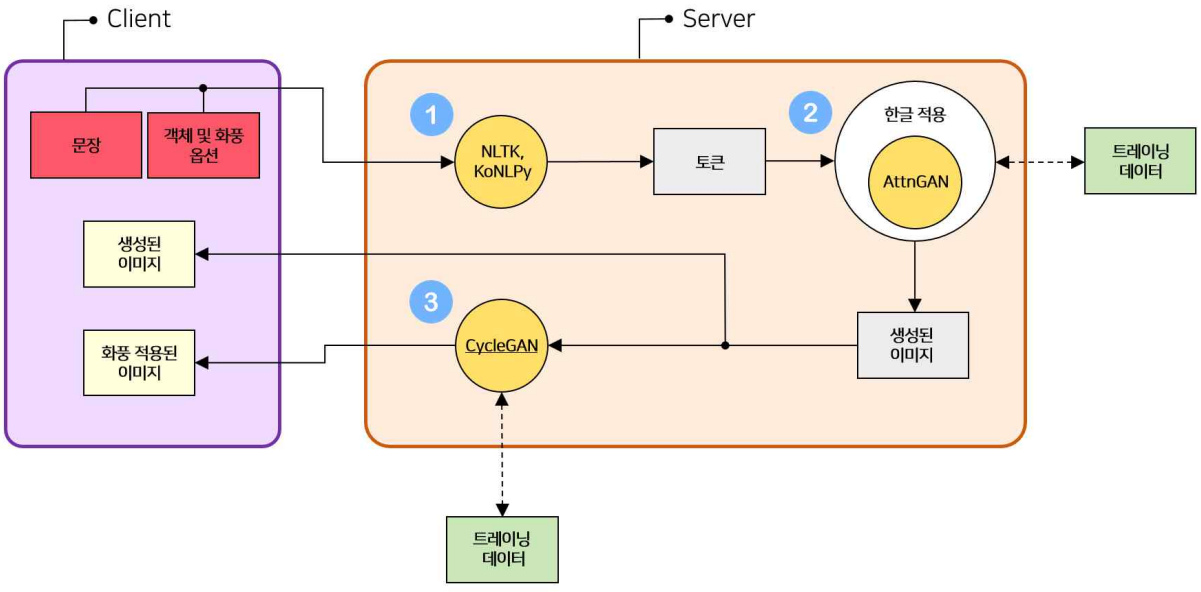


그림 14. 최종 시스템 DFD

3.2 구현 상세

3.2.1 라이브러리 테스트

과제 수행에 앞서 가장 먼저 조사했던 라이브러리들에 대해서 사용 가능 여부와 성능 테스트를 수행했다. 그러나 초기에 팀원들의 PC 환경으로는 라이브러리를 테스트할 수 없었다. 그래서 Google에서 유료로 제공하는 Colab Pro[11] 버전을 구매하여 테스트 환경을 구성하였다.

GitHub에 오픈소스로 공유되어있는 AttnGAN을 CUB_2011 데이터셋과 COCO 데이터셋을 이용해서 성능 테스트 및 검증을 했다. 이때 Colab 환경과 물리적, 시간적 제약이 존재했는데 이를 고려하여 데이터셋의 크기를 축소해 AttnGAN 라이브러리 테스트를 진행했다. CUB_2011 데이터셋에 대해서는 Text 10,000~60,000문장, Image 1,000장~6,000장으로, COCO Dataset에 대해서는 Text 15,000~60,000문장, Image 3,000장~12,000장으로 테스트를 수행했다. 학습 횟수는 DAMSM 모델과 AttnGAN 모델 모두 똑같이 600번으로 고정했다. 이렇게 진행했을 때 AttnGAN 전체 학습 시 약 23시간 정도의 시간이 소요됐다.

CycleGAN에 대해서는 테스트 전에 화풍 데이터 수집하는 과정이 먼저 필요했다. 테스트를 원했던 화풍은 브라이언 패터슨의 동화 삽화였다. 그림 15와 그림 16의 브라이언 패터슨의 동화 삽화 중 일부이다. 주로 Pinterest[12]라는 공개 이미지 SNS에서 데이터를 수집했는데, 브라이언 패터슨 삽화의 수가 학습을 위해 필요한 최소 데이터셋 1,000장을 채우기에는 한참 부족했다. 그래서 꼭 브라이언 패터슨의 삽화가 아니더라도 비슷한 느낌의 이미지를 데이터셋에 포함했다. 한 번에 대량의 이미지 데이터를 수집한 뒤 걸러내는 작업을 수행했기 때문에 데이터 수집에 꽤 많은 시간이 소요되었다. 약 1,000장으로 구성한 데이터셋에 대해 batch size 4, epoch 100으로 설정하고 학습을 수행했을 때 약 15시간 정도의 시간이 소요됐다.

그림 15. 브라이언 패터슨의 동화 삽화1

그림 16. 브라이언 패터슨의 동화 삽화1

3.2.2 데이터 수집, 가공 및 생성 #산업체 멘토링 결과 반영

AttnGAN을 통해 객체 생성을 하기 위해서는 객체의 이미지와 해당 이미지를 서술하는 데이터가 필요하다. 그래서 새 이외의 동물을 생성하기 위해 기존의 데이터셋 이외에 또 다른 데이터 수집이 필요했다. 필요한 다른 동물 이미지에 대해서는 ImageNet[13], Kaggle[14]에서 미리 정제된 이미지 셋을 내려받거나 google[15], pinterest, instagram[16] 등에서 이미지를 수집하는 방법을 사용했다.

직접 이미지를 수집할 때는 Selenium을 이용하여 웹 크롤링 시 프로그램이 하는 것이 아닌 브라우저 동작을 제어해 사용자가 직접 이용하는 것 같이 웹 페이지를 요청하여 접근 제어를 받지 않고 이미지를 수집했다. Google에서는 원하는 동물의 단어를 검색 후 나오는 이미지를 내려받았고, pinterest와 instagram에서는 원하는 동물의 해시태그를 붙인 이미지를 검색하여 내려받았다. 내려받은 이미지들에서 동물 피사체가 제대로 보이지 않거나 정확하게 해당 동물을 인식하기 어려운 이미지들이 있어 학습에 적합하지 않기 때문에 직접 골라내 제외했다.

이미지에 대한 텍스트 데이터에 대해서는 수집한 동물 한 이미지 당 5문장으로 직접 작성하여 만들었다. 텍스트 문장은 동물이 어떤 행동을 하는지, 어떤 생김새인지 등을 묘사한다. 이 텍스트 데이터 생성을 팀원 모두가 달려들어 직접 다 작성했는데도 대량의 이미지에 대한 텍스트 파일을 생성하다 보니 상당히 많은 시간이 소요됐다.

CycleGAN을 통해서는 동화풍 외에 추가로 다양한 화풍을 학습시켰다. 동화풍 적용이 티가 많이 나지 않았던 차에 산업체 멘토링으로 수묵화나 동양화와 같이 극적인 화풍을 적용해보라는 조언을 반영해서였다. 그림 17은 수묵화 데이터셋 중 일부이고, 그림 18은 동양화 데이터셋 중 일부이다. 화풍 데이터 수집은 라이브러리 테스트 때와 같은 방식으로 수집했다.

그림 17. 수묵화 이미지

그림 18. 동양화 이미지

3.2.3 AttnGAN을 활용한 텍스트-이미지 변환 #산업체 멘토링 결과 반영

초기에는 AttnGAN 학습할 때 데이터 수를 축소하는 대신 에포크를 증가시켜 사진의 품질을 높이려고 시도했다. 당시 학습을 이어서 하는 방법을 적용하지 못하고 있었기 때문에 데이터셋을 최대한 축소하더라도 에포크는 600이 최대였다.

그림 19. 초기 AttnGAN 학습 결과(epoch: 520, 데이터 수: 5625)

그 때문에 결과 이미지의 품질은 그림 19를 보면 알 수 있듯이 새라고 생각할 수 없을 정도의 수준이었지만 이후 학습을 이어서 할 수 있게 되면서 에포크의 제한이 없어졌고, 동시에 데이터셋을 축소할 필요도 없어져 최대 데이터셋으로 학습을 진행했다.

현재 만들어진 AttnGAN 학습 모델을 학습시킬 때 사용된 데이터는 한 동물당 Image 1,000~12,000장과 Text 5,000~120,000문장이다. 새의 경우 데이터가 충분했기 때문에 약 12,000장의 이미지와 이미지 하나당 10개의 문장을 학습할 수 있었지만, 다른 동물은 데이터가 부족했기 때문에 이를 위해서 배치 크기와 학습 횟수를 조정하여 학습시켰다. 전체적으로 배치 크기는 평균적으로 약 40 정도를 잡았으며 학습 횟수는 DAMSM은 평균적으로 1500번, AttnGAN에서는 평균적으로 4000번 정도 학습시켰다.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.66       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   33C    P0    23W / 300W |      0MiB / 16130MiB |      0%      Default |
|                               |                      |                 ERR! |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

그림 20. Colab Pro의 GPU 환경 상세

배치 크기는 그림 20에서 볼 수 있듯이 Colab Pro의 GPU 환경이 그렇게 좋은 편이 아니어서 GPU가 허용하는 선에서 최대한 활용할 수 있도록 하여 학습 속도를 조금 높였고, 학습 횟수는 과적합이 되지 않는 선에서 최대한 학습하도록 하여 사진 품질을 최대한 높이도록 노력했다. 실제로 학습을 연이어 할 수 있게 된 후 최대 에포크와 함께 품질이 급상승했다.

3.2.4 AttnGAN의 한글화

AttnGAN은 원래 NLTK를 이용하여 영어 텍스트 데이터를 토큰화한 뒤 학습에 사용한다. 우리는 입력 데이터를 한글화하기 위해 우선 학습에 사용되는 텍스트 데이터를 전부 파파고 번역기를 이용하여 번역했다. 단순히 데이터만 번역해서 되는 것이 아니라 코드상에서 텍스트 데이터를 인코딩하고 디코딩하는 부분이 ASCII로 되어있는 것을 UTF-8로 모두 변환했다. 또한, 그림 21과 같이 입력 텍스트 파일도 windows 기본 메모장 인코딩이 ANSI로 되어있었기 때문에 그림 22처럼 전부 UTF-8로 변경해야 했다.

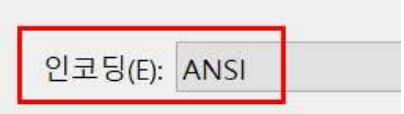


그림 21. 인코딩 변경 전.

그림 22. 인코딩 변경 후.

1차로 한글화를 마치고 학습을 시작했다. 학습 결과 한글 데이터를 입력하고 이미지를 생성하는 것에는 문제가 없었지만, 사용되는 텍스트 분석기가 NLTK였기 때문에 한글 텍스트 데이터를 분석하기에는 부적절했다. 그래서 불필요한 품사들이 이미지와 매칭되어 이미지의 품질을 떨어뜨리는 문제가 발생했다.

그래서 우리는 NLTK를 한글 형태소 분석 라이브러리인 KoNLPy로의 교체로 결정했다. 먼저 Tokenizer를 KoNLPy의 형태소 분석기 중 하나인 Okt로 교체했다. KoNLPy의 많은 형태소 분석기 중 Okt를 선택한 이유는 다른 형태소 분석기에 비교해 태깅되는 품사의 종류가 단순했기 때문이었다. 품사가 지나치게 상세하더라도 이미지의 특징을 텍스트와 대응하는 데 방해가 될 것으로 생각했다. 이렇게 Okt를 통해 분석한 품사 중 명사, 형용사, 동사만을 학습에 사용하도록 하여 학습 정확도를 조금 더 향상하려고 했다.

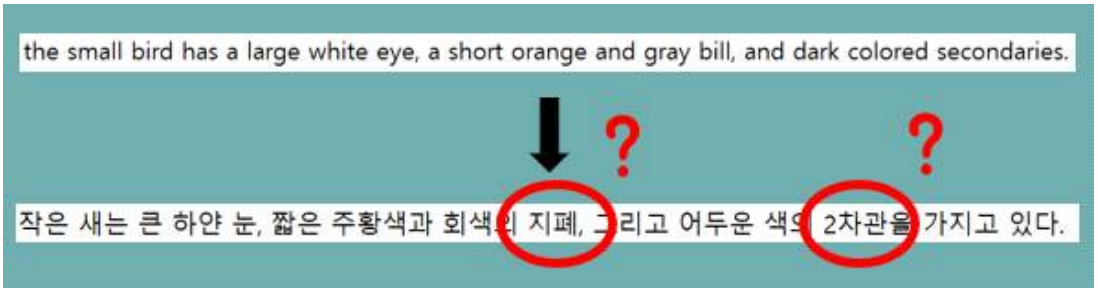


그림 23. 오역된 텍스트 예시

또한, 파파고 번역기를 통해 번역했던 데이터를 구글 번역기를 사용하여 번역한 것으로 바꿨다. 파파고가 구글에 비교해 오역이 좀 더 많았기 때문에 오역을 바로잡아 학습에 사용되는 텍스트 데이터의 품질을 높이기 위해서였다. 그림 23에서 볼 수 있듯이, 'bill'은 '부리'로 번역되어야 하지만 '지폐'로 오역되었다. 또한, '2차관'이라는 알 수 없는 번역도 존재한다.

추가로, 문장을 토큰화한 뒤 토큰들의 전체 집합에서 문장에 포함된 횟수가 일정 이하인 토큰은 오역이라 판단하고 학습에서 제외했다. 이러한 여러 가지 시도들을 통해 입력 데이터의 한글화에 성공하였고, 생성되는 이미지의 품질을 끌어올릴 수 있었다.

3.2.5 CycleGAN을 활용한 화풍 적용

CycleGAN 역시 학습을 이어서 하지 못했던 초기에는 개발환경의 한계로 인해 에포크 100 정도가 최대였지만, 학습을 이어서 할 수 있게 된 이후로는 에포크를 증가시켜가며 계속 학습한 뒤, 손실 함수를 통해 계산된 손실 값을 기록한 그래프를 관찰하여 최적의 epoch를 선정했다.

초기 CycleGAN은 학습의 부족으로 인해 이미지의 품질이 많이 떨어지고, 생성된 이미지에 화풍이 적용됐다고 말할 수 없을 정도였지만, 이후에는 오히려 학습이 과도하게 진행되어 결과 이미지가 지나치게 변형되는 경우가 생겨 오히려 더 알아볼 수 없는 경우가 생겼다. 이는 최적 epoch를 선정함으로써 해결할 수 있었다.

이후 동양화, 수묵화에 이어 추가로 세잔, 모네, 반고흐의 화풍을 수집하여 학습했다. 이를 통해 사용자의 선택지를 넓힐 수 있었다.

3.2.6 사용자 인터페이스

이미지 생성을 하는 AttnGAN과 CycleGAN에서의 성과를 어느 정도 본 뒤, 하나로 합쳐 시스템 통합을 진행했다. 사용자에게 웹 인터페이스를 제공하기 위해 웹 프레임워크 Flask[17]를 사용했다. Flask를 사용하는 것만으로도 로컬에서 웹 인터페이스를 확인할 수 있는데, 이 상태가 하나의 프로토타입이기 때문에 최종 모듈 테스트를 진행하면서 결과와 품질 비교를 많이 했다.

그리고 외부에서도 접근할 수 있게 웹 호스팅 서비스 Pythonanywhere[18]를 이용해 호스팅을 진행하였다. 이 과정에서 파이썬 버전 불일치, 이미지에 대한 경로 문제, 용량 부족 그리고 GPU 사용 불가 등의 어려움을 겪었다. 해결을 위해 파이썬 3버전으로 코드를 수정했고, 이미지 파일에 대한 경로는 재지정했다. 부족한 파일 서버 용량은 구매하였으며, GPU를 사용하지 않고 CPU를 사용하도록 코드 수정하는 것으로 문제들을 해결했다. 다만, GPU를 사용하는 로컬 환경에 비교해 CPU를 사용하기 때문에 실행, 반응 속도와 품질이 비교적 더 떨어졌다.

그림 24는 최종 시스템의 구조를 나타낸다. 먼저 사용자가 웹을 통해 문장과 객체, 화풍을 입력으로 주면, pythonanywhere와 flask를 통해 내부 코드에 전달이 된다. 전달된 입력 데이터는 이미 학습되어 생성된 모델을 기반으로 이미지가 생성과 화풍 적용되어 결과를 만들어낸다. 결과물들은 다시 flask와 pythonanywhere를 통해 웹에 전달이 되고 사용자에게 최종적으로 출력이 된다.

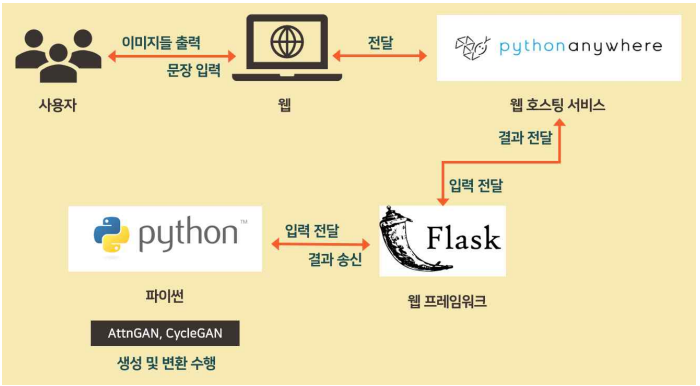


그림 24. 통합된 시스템의 구조

3.3 과제 결과물

3.3.1 웹 페이지

최종으로 만들어진 웹 페이지의 주소는 아래와 같다. 그림 25는 해당 페이지로 접속하면 과제 수행한 결과물을 올려놓은 것을 확인할 수 있는 UI이다.
GANdaParkChangJo.Pythonanywhere.com

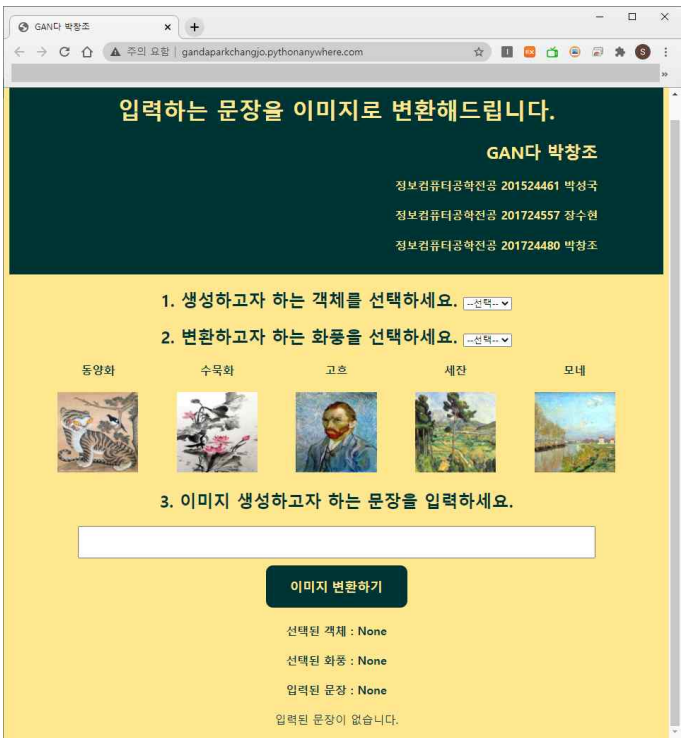


그림 25. 최종 시스템이 제공되는 UI

3.3.2 웹 수행 결과물

객체와 화풍을 선택한 뒤 문장을 입력하면 생성된 이미지와 화풍이 적용된 이미지가 출력된다. 그림 26과 그림 27은 결과 중 일부이다. 그림 26은 동양화로 변환되었고, 그림 27은 세잔의 화풍으로 변환되었다.

그림 26. 웹 페이지에서 확인할 수 있는 결과물 1

그림 27. 웹 페이지에서 확인할 수 있는 결과물 2

3.3.3 AttnGAN 결과물

한글 텍스트를 입력하면 문장에 대응하는 이미지를 출력할 수 있다. 현재 새에 대해서는 정확도 높은 이미지를 생성할 수 있지만, 토끼와 호랑이는 아직 정확도가 낮아 학습을 계속 진행 중이다. 그림 28은 토끼 이미지 생성에 대한 학습 과정에서 출력된 결과물 중 품질이 높은 편에 속하는 이미지이다.

그림 28. 토끼 이미지 생성 중간 결과

새 이미지 생성은 비교적 높은 정확도를 보인다. 아래 그림 29~33은 실제 학습이 완료된 모델에 한글 텍스트를 입력하여 나온 결과물 중 비교적 높은 정확도를 보이는 예시를 모아놓은 것이다.

이 새는 회색의 큰 날개와 길고 구부러진 검은 부리, 하얀 몸 깃털을 가지고 있다

그림 29. 새 이미지 생성 예시 1

오리

그림 30. 새 이미지 생성 예시 2

노란색 몸을 가지고, 머리가 빨간색이고, 부리가 긴 새

그림 31. 새 이미지 생성 예시 3

이것은 파란색 꼬리 깃털과 노란색 얼굴을 가진 아름답고 작은 새다

그림 32. 새 이미지 생성 예시 4

이 새는 주로 검고 몸 전체와 날개 전체에 노란색이 두드러진다

그림 33. 새 이미지 생성 예시 5

3.3.4 CycleGAN 결과물

최종적으로 총 5가지의 화풍을 학습하였으며, 화풍별 최적 에포크는 손실 그래프를 기반으로 선정했다. 그림 34를 보면 윤곽의 변화는 적지만, 색감의 차이가 두드러지는 것을 확인할 수 있다.

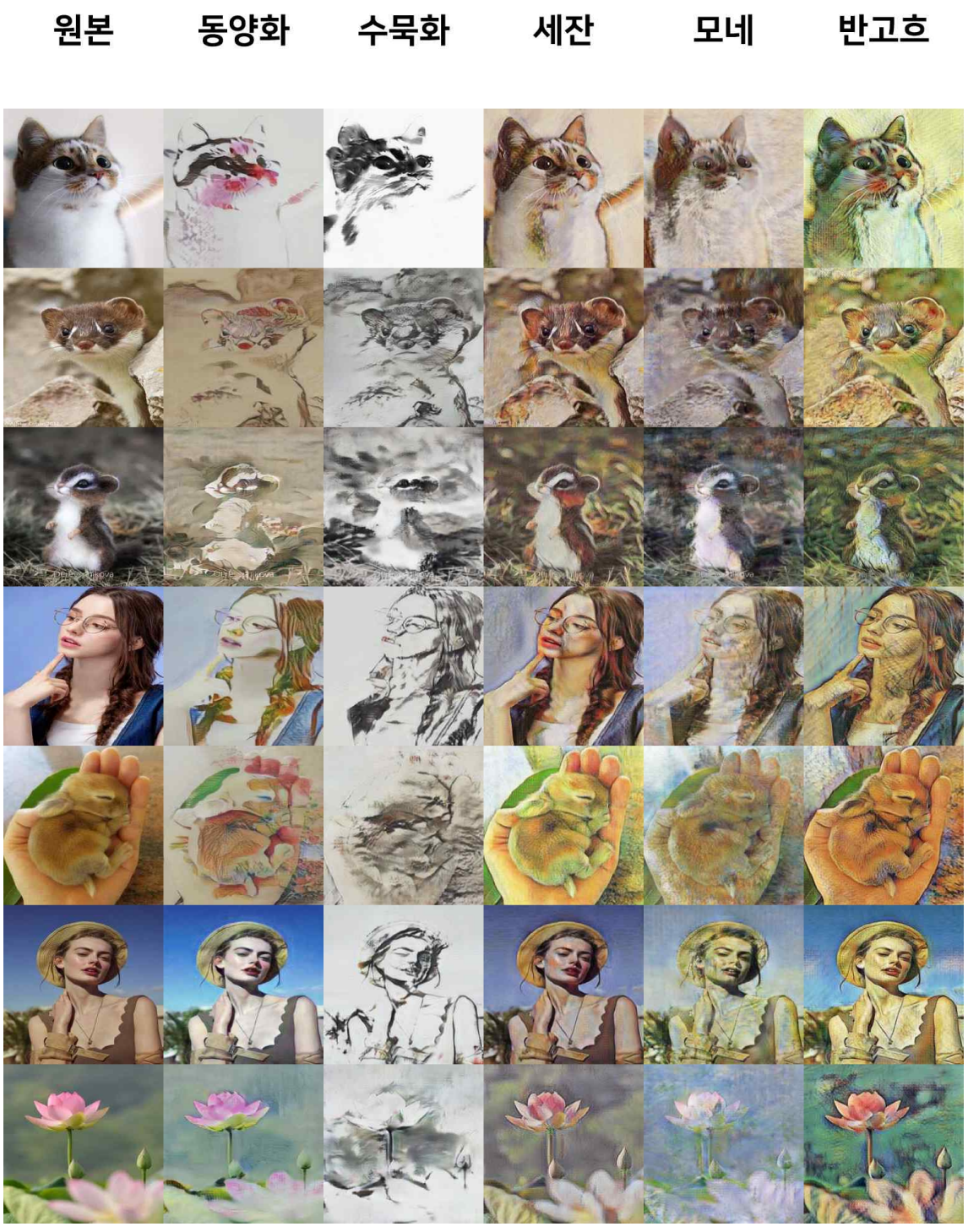


그림 34. 여러 사진에 각각 5가지 화풍을 적용한 결과.

4. 과제 결과 분석 및 평가

4.1 AttnGAN

AttnGAN을 구성하는 두 가지 모델 중 DAMSM은 문장 수준에서 이미지와 얼마나 매칭되는지를 측정할 수 있는 문장 레벨 손실(s_loss)과 단어 수준에서 이미지와 얼마나 매칭되는가를 측정하는 단어 레벨 손실(w_loss)로 손실을 나누어 측정할 수 있는데, s_loss와 w_loss를 이용하여 DAMSM의 전체 손실을 계산할 수 있다.

$$\mathcal{L}_{DAMSM} = \mathcal{L}_1^w + \mathcal{L}_2^w + \mathcal{L}_1^s + \mathcal{L}_2^s.$$

수식 1. DAMSM 손실 계산 수식

수식 1에서, L_1^w과 L_2^w를 합하여 w_loss를 계산할 수 있고, L_1^s과 L_2^s를 합하여 s_loss를 계산해낼 수 있다. 우리는 L_{DAMSM}을 바탕으로 DAMSM의 최대 학습 에포크를 약 2000으로 설정했다.

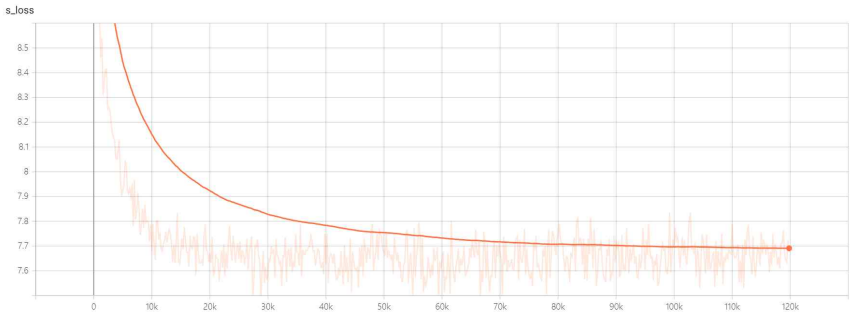


그림 35. DAMSM의 s_loss를 측정한 그래프

그림 35은 DAMSM의 문장 레벨 손실(s_loss)을 측정한 그래프이다. 그래프를 보면 약 80k 지점부터 손실이 수렴하기 시작한다는 것을 관찰할 수 있다.

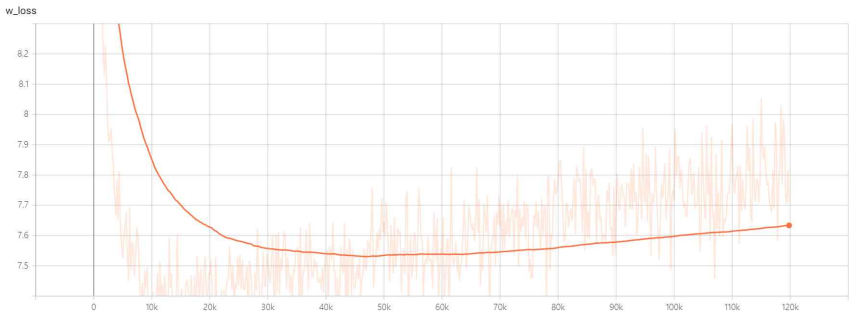


그림 36. DAMSM의 w_loss를 측정한 그래프

그림 36는 DAMSM의 단어 레벨 손실(w_loss)을 측정한 그래프이다. w_loss의 경우 s_loss와는 다르게 약 40k 지점부터 수렴하기 시작한다. L_{DAMSM}은 s_loss와 w_loss의 합이므로, 우리는 최적 step을 두 지점의 중간값인 60k 지점으로 설정했다. 1번의 epoch는 약 30번의 step으로 이루어져 있으므로, 최적 epoch는 2000으로 결정되었다.

이어서 우리는 AttnGAN을 구성하는 나머지 부분인 Attentional Generative Network에서도 비슷한 방법으로 최적 epoch를 결정하려고 했으나, 약간의 문제가 있었다. Attentional Generative Network에서는 생성자가 얼마나 실제와 같은 이미지를 생성해내는지를 측정할 수 있는 Loss_G와 식별자가 얼마나 이미지를 정확하게 식별하는지를 측정할 수 있는 Loss_D가 존재하는데, 그림 37과 그림 38을 보면 현재 epoch가 2000인 상황에서 손실 값이 수렴하지 않고 있다. 그래서 최적 epoch를 최대 epoch로 설정했다. 그림 39를 보면 에포크가 증가함에 따라 생성된 새 이미지가 점점 정확해지고 있음을 알 수 있다.

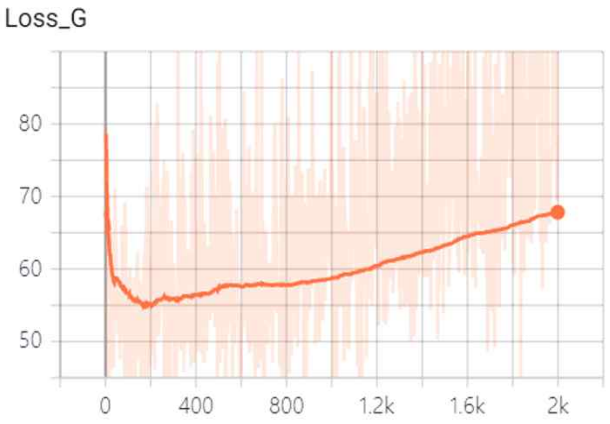


그림 37. Loss_G를 측정한 그래프

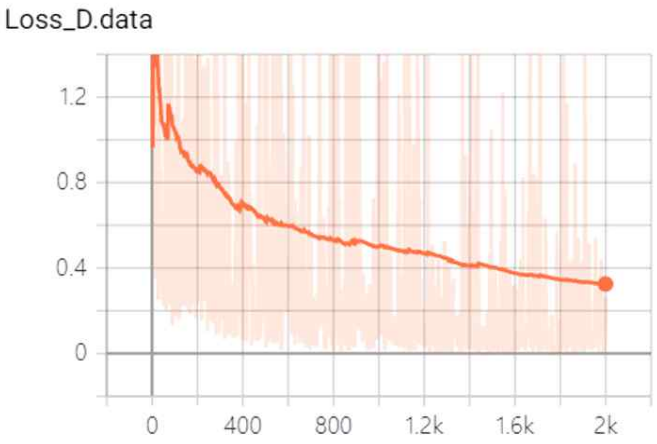


그림 38. Loss_D를 측정한 그래프

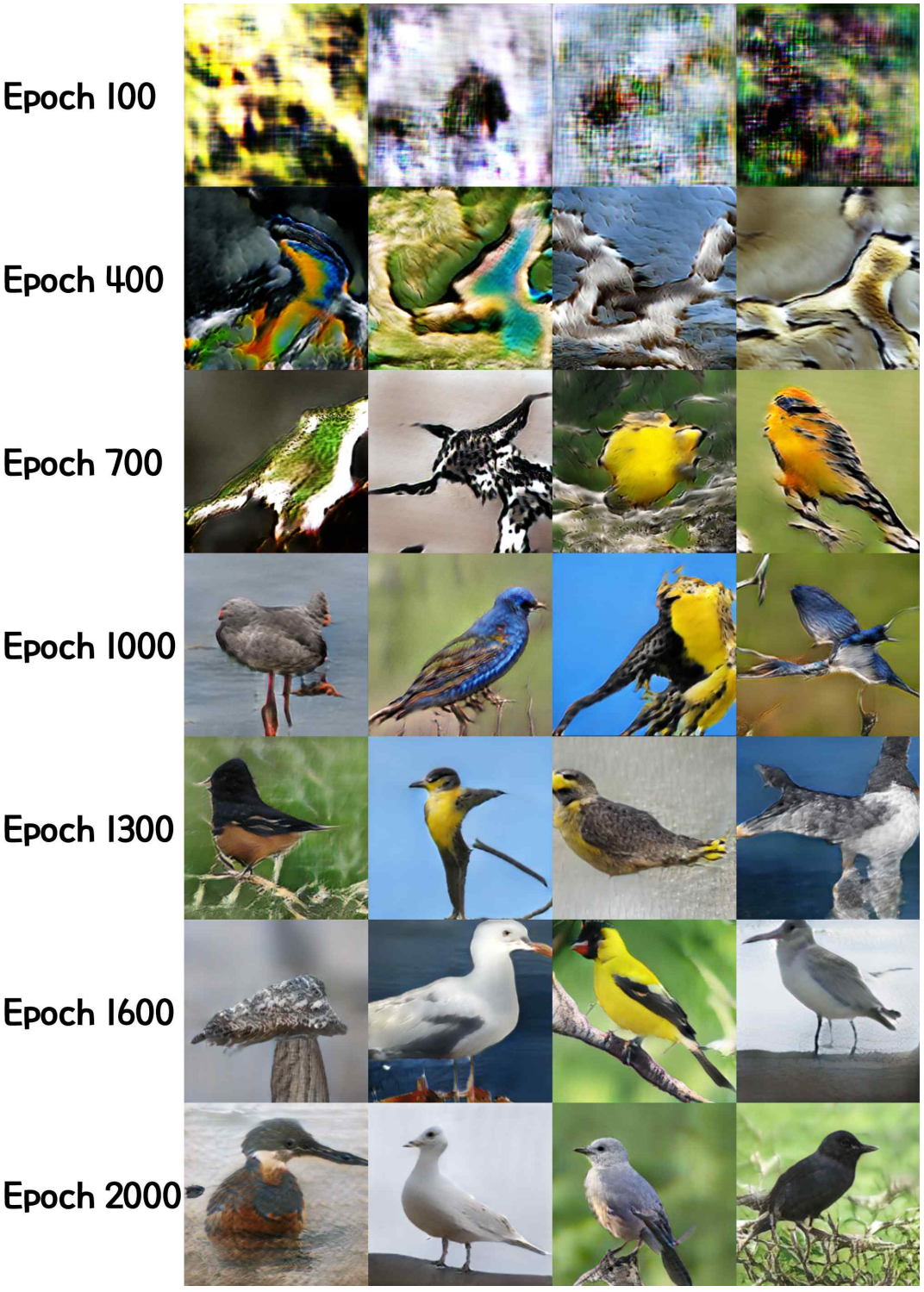


그림 39. 에포크에 따른 새 이미지 생성 결과물 변화

4.2 CycleGAN

우리는 CycleGAN을 이용하여 총 5가지의 화풍(동양화, 수묵화, 세잔, 모네, 반고흐)을 학습하는 데 성공했다. 각 화풍을 학습할 때의 최적 epoch는 도메인 A에서 B로 변환한 뒤 다시 A로 변환했을 때 원본 이미지와의 차이를 측정한 loss_cycle_a와 반대로 B에서 A로, 다시 B로 변환했을 때의 손실을 측정한 loss_cycle_b를 이용하여 결정했다.

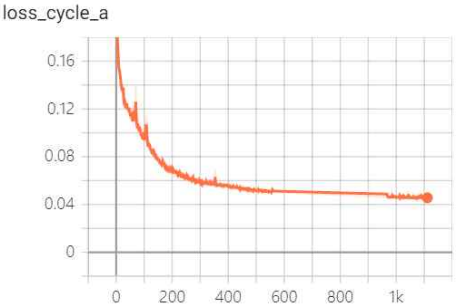


그림 40. 수묵화의 loss_cycle_a 그래프

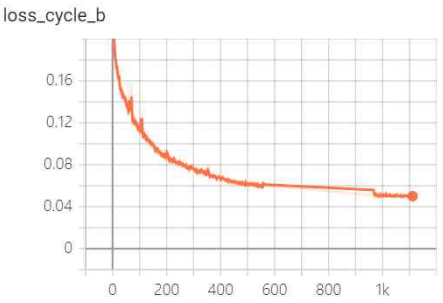


그림 41. 수묵화의 loss_cycle_b 그래프

그림 40을 보면 약 800~900에서 값이, 그림 41을 보면 마찬가지로 800~900에서 값이 수렴함을 알 수 있다. 그 때문에 우리는 최적 epoch를 900으로 결정했다. 다른 화풍도 마찬가지 방식으로 최적 epoch를 결정했고, 동양화는 epoch 300, 모네는 epoch 1200, 반고흐는 epoch 300으로 최종 결정되었다.

4.3 KoNLPy

기존의 입력 텍스트 데이터를 분석하여 토큰화하는 형태소 분석기로 NLTK를 사용했지만, 한글 데이터를 토큰화하기에는 적절하지 않다고 판단되어 최종적으로 한국어 형태소 분석기인 KoNLPy를 사용했다.

또한, 텍스트와 이미지를 사상할 때 객체의 특징을 표현하는 것과는 관련이 없는 단어가 이미지와 대응되는 것을 방지하기 위하여 객체의 특징을 표현하는 것과 관련 없을 만한 품사들을 사전에 걸러내는 작업을 추가했다.

우리가 사용한 형태소 분석기는 KoNLPy의 여러 분석기 중 Okt 분석기인데, 해당 분석기는 한글 문장을 입력받으면 아래 표 2와 같이 품사를 구분한다.

표 2. Okt 형태소 분석기의 품사 태깅 표[19]

기호	품사	기호	품사	기호	품사	기호	품사
Adjective	형용사	Determiner	관형사	Hashtag	해시태그	Number	숫자
Adverb	부사	Eomi	어미	Josa	조사	PreEomi	선어말 어미
Alpha	알파벳	Exclamation	감탄사	Korean-Particle	(ex:ㅋㅋ)	Punctuation	구두점
Conjunction	접속사	Foreign	외국어	Noun	명사	Screen-Name	트위터 아이디
Suffix	접미사	Unknown	미등록	Verb	동사		

학습에 사용되는 한글 텍스트 데이터를 분석한 결과 전체 토큰 중 약 70%가 명사, 형용사, 동사였고, 나머지 30%는 숫자, 접속사, 접미사 등 객체의 특징을 표현하기에는 부적절한 품사들이라고 판단되어 걸러냈다. 명사와 형용사를 포함한 이유는 명사의 경우 새의 부위를 지칭하는 '날개', '머리', '가슴', '배' 등의 단어를 포함하기 때문이고, 형용사의 경우 '빨간색의', '화려한', '긴', '작은' 등 새 부위를 수식하는 단어들을 포함하기 때문이다. 아래의 코드 1은 불필요한 품사를 제외한 나머지를 학습 데이터에 추가하는 코드 일부이다.

```
def load_captions(data_dir, filenames, split2):
   all_captions = []
   for i in range(len(filenames)):
       cap_path ='%s /%s /text/%s .txt'% (data_dir, split2, filenames[i])
       with codecs .open(cap_path, "r",encoding ='utf8') as f:
           captions = f .read().split('\n ')
           cnt =0
           for cap in captions:
               if len(cap) ==0:
                   continue
               cap = cap .replace("\ufffd\ufffd ", " ").replace('\ufeff ','')
               tokenizer = Okt()
               tos = tokenizer .pos(cap)
               tokens = []

               for t in tos:
                   if t[1]=='Noun'or t[1]=='Verb'or t[1]=='Adjective':
                       if t[0]!='이'and t[0]!='있습니다'and t[0]!='입니다':
                           if not '니다'in t[0]:
                               tokens .append(t[0])

               if len(tokens) ==0:
                   continue

               tokens_new = []
               for t in tokens:
                   t = t .encode('utf-8', 'ignore').decode('utf-8')
                   if len(t) >0:
                       tokens_new .append(t)
               all_captions .append(tokens_new)
               cnt +=1
               if cnt ==10:
                   break
   return all_captions
```

코드 1. 불필요한 품사를 제외한 나머지를 학습 데이터에 추가하는 코드 일부

동사를 포함한 이유는 조금 다른 이유인데, 일반적으로 동사는 객체의 행동을 나타내어 객체를 생성하는 데는 불필요한 정보이지만, 학습 데이터가 '새'에 관한 것이다 보니 특정 문장에서 필수 단어가 동사로 잘못 분류되는 경우가 생겼기 때문이다. 예를 들어, '이 새는 하얀색의 머리와 검은색의 몸, 회색빛의 날개를 가졌다.'라는 문장이 있다고 해보자. 이때 '새는'이라는 단어는 명사로 분류되어야 하지만, '물이 새는', '기름이 새는' 등에 사용되는 '새다'라는

동사로 분류되어버리는 것이다. 만약 동사를 제외하게 된다면, 새를 표현하기 위한 문장에서 '새'라는 객체가 빠져버리는 치명적인 문제가 발생하게 되기 때문에 약간의 불필요함을 감수하고서라도 동사를 포함하게 되었다.

또한, 우리는 기존에 영어로 만들어진 텍스트 데이터를 일일이 번역기를 통해 번역했는데, 이때 구글 번역기와 파파고 번역기를 둘 다 사용하여 그 결과를 비교해보았다. 기존 데이터는 11,788개의 텍스트 파일이지만, 실험에서는 2,903개의 텍스트 파일을 사용했다. 코드 2는 실험에 사용된 코드 일부이다.

```
def build_dictionary(train_captions, test_captions, split2):
    word_counts = defaultdict(float)
    captions = train_captions + test_captions
    for sent in captions:
        for word in sent:
            word_counts[word] +=1

    keys =list(word_counts .keys())

    total_token_num =0.0
    mis_token_num =0.0
    for key in keys:
        v = word_counts[key]
        total_token_num +=v
        if v <=4.0:
            mis_token_num +=v

    print('===============<%s >==============='%split2)
    print('total_token_num: %d '%int(total_token_num))
    print('mis_token_num: %d '%int(mis_token_num))

    print('전체 토큰에 대한 비율: %.3f%%'%((mis_token_num /total_token_num)*100.0))
    print()

    vocab = [w for w in word_counts if word_counts[w] >=0]

    ixtoword = {}
    ixtoword[0] ='<end>'
    wordtoix = {}
    wordtoix['<end>'] =0
    ix =1
    for w in vocab:
        wordtoix[w] = ix
        ixtoword[ix] = w
        ix +=1

    train_captions_new = []
    for t in train_captions:
        rev = []
        for w in t:
            if w in wordtoix:
                rev .append(wordtoix[w])
        # rev.append(0)  # do not need '<end>' token
        train_captions_new .append(rev)

    test_captions_new = []
    for t in test_captions:
```

```
        rev = []
        for w in t:
            if w in wordtoix:
                rev .append(wordtoix[w])
        # rev.append(0)  # do not need '<end>' token
        test_captions_new .append(rev)

    return [train_captions_new, test_captions_new, ixtoword, wordtoix, len(ixtoword)]
```

코드 2. Papago와 Google 번역기 비교를 위한 코드 일부

```
===============<google_text>===============
total_token_num: 283181
mis_token_num: 3361
전체 토큰에 대한 비율: 1.187%

===============<papago_text>===============
total_token_num: 311586
mis_token_num: 4805
전체 토큰에 대한 비율: 1.542%
```

그림 42. 구글 번역기와 파파고 번역기 비교 결과

그림 42는 같은 영어 데이터를 구글 번역기와 파파고 번역기를 통해 각각 번역한 뒤, 토큰화를 거친 뒤 오번역의 비율을 간접적으로 측정하기 위해 진행한 실험이다. 결과를 보면 전체 토큰의 수는 google 번역기로 번역한 데이터의 경우가 더 적은데, 이는 같은 의미이지만 다르게 번역된 단어의 중복이 줄었기 때문으로 추측된다. 예를 들어, 'bill'과 'beak'는 모두 우리말로 '새의 부리'를 나타내지만, 파파고에서는 '부리'보다 '지폐'로 번역되는 경우가 눈에 자주 띄었다. 이러한 오역이 구글 번역기에서는 더 적게 발생하는 것이다.

두 번째로 알 수 있는 것은 전체 토큰에서 일정 횟수 이하로 등장한 토큰의 비율이다. 위의 실험에서는 5회 이하로 등장한 토큰의 수를 세어봤는데, 전체에서 차지하는 비율이 google 번역기로 번역한 한글 데이터에서는 1.187%, 파파고 번역기를 통해 번역한 한글 데이터에서는 1.542%로 파파고 번역기가 구글 번역기보다 약 1.3배 더 높은 비율을 차지했다. 오역된 데이터는 여러 번 등장하지 않을 가능성이 크다는 것을 고려해봤을 때, 파파고 번역기의 오역 가능성이 그만큼 더 크다는 것으로 생각할 수 있다.

5. 결론

5.1 활용 방안

이미지-텍스트 변환에 비교해 텍스트-이미지 변환은 지금까지 갈 길이 먼 분야이다. 입력한 문장에 정확하게 대응되는 이미지를 생성할 수 있다면 활용 분야는 무궁무진하다. 우선, 희귀병 환자에게 도움이 될 수 있다. 아판타시아[20]라는 병은 상상이라는 것이 불가능한 희귀병인데, 이 병을 앓고 있는 환자들이 이 기술을 활용한다면 상상이라는 것을 대신할 수 있을지도 모른다.

또한, 텍스트로 이루어진 글을 이미지가 포함된 글 혹은 만화책으로 변환할 수 있다. 이는 단순히 글을 그림으로 변환한다는 것뿐만 아니라 실질적 문맹이 점점 많아지고 난독증이 증가하는, 글을 읽기 싫어하는 사람이 증가하는 사회에서 글을 이해하는 데 큰 도움이 될 수 있을 것이다.

소소하게는 작가에게도 도움이 될 수 있다. 그림을 잘 그리지 못하지만, 글을 잘 쓰는 작가가 있다면 이 기술을 이용해 혼자서 그림책 혹은 만화책을 만들 수도 있을 것이다.

5.2 향후 과제

이하지만 위의 활용 방안에 기술한 과제를 달성하기 위해서는 개선되어야 할 사항들이 많다. 첫 번째로 개선되어야 할 것은 현재 텍스트-이미지 변환은 객체 중심에 치중되어 있다. 예를 들어, '귀가 길고 눈이 빨간색이고 다리가 짧은 토끼가 있다'라는 문장은 이미지로 변환할 수 있지만, '토끼가 달리고 있다'라는 동작을 묘사하는 문장은 이미지로 변환할 수 없다.

두 번째로, 현실에 존재하는 객체만을 생성할 수 있다는 문제점이 있다. 텍스트-이미지 변환도 결국 학습에 사용되는 사진을 기반으로 이미지를 생성하는 것이기 때문에, '팔과 다리가 달린 물고기가 식탁에 앉아 수프를 먹고 있다' 같은 현실에 존재할 수 없는 이미지는 생성할 수 없다.

세 번째는 현재 우리 과제는 문장 혹은 구 수준의 텍스트를 입력으로 받아 이미지를 생성하는데, 초기 설계처럼 책 한 권을 만화책으로 바꾸는 수준이 되려면 먼저 장문의 글을 정확한 몇 개의 문장으로 요약하는 기술이 필요할 것이다.

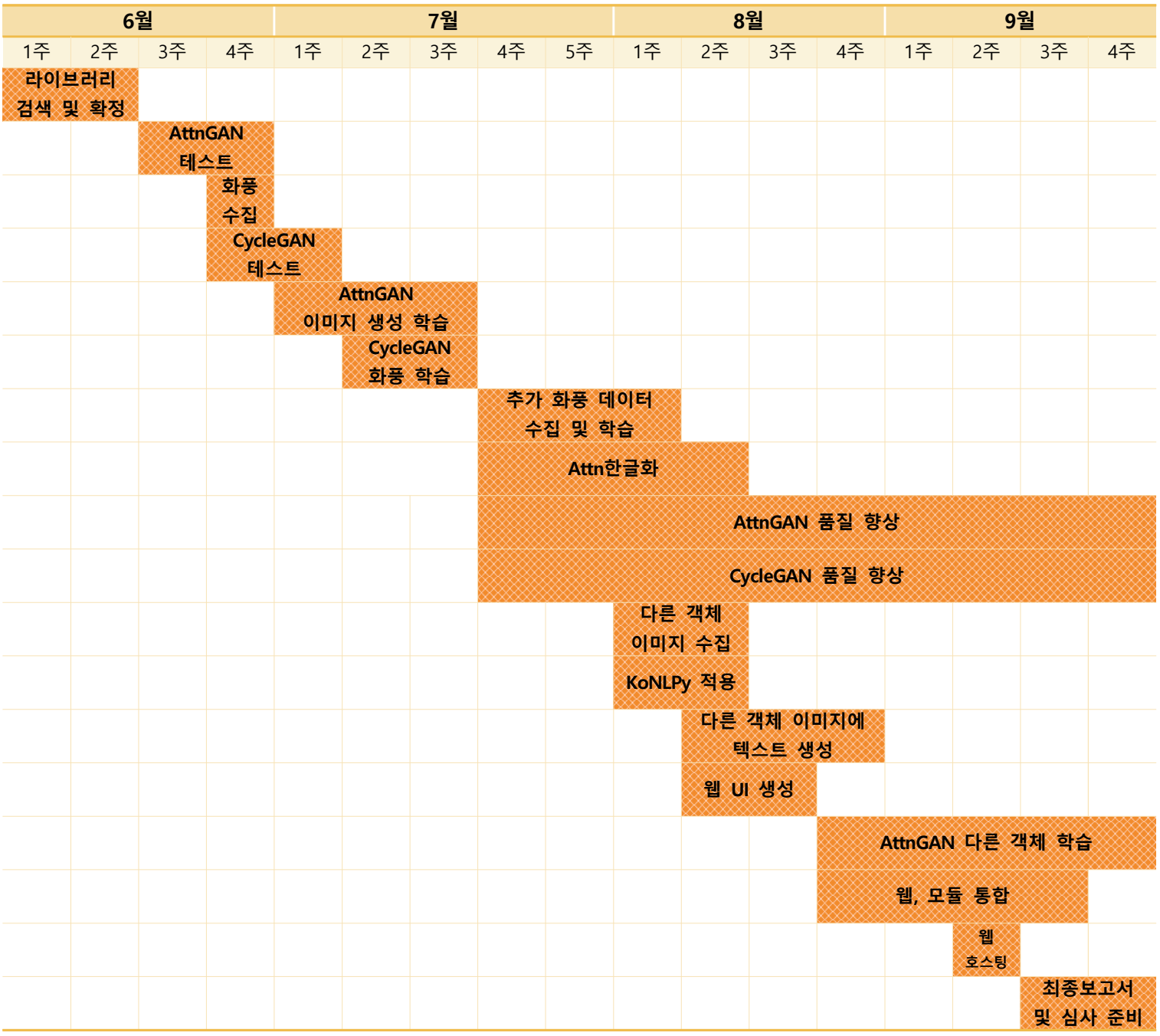
마지막으로, CycleGAN 기술 자체의 한계점이 있다. CycleGAN은 이미지의 색상을 바꿀 수는 있지만, 윤곽선을 바꿀 수는 없다. 다시 말해, 사진의 객체를 색을 변경하여 그림처럼 보이게 할 수는 있지만, 실제 그림처럼 선이 울퉁불퉁해진다든지 하는 변환은 불가능하다. 해당 변환이 가능해진다면 좀 더 사람이 그린 그림의 느낌을 낼 수 있을 것이다.

6. 개발 일정 및 역할분담

6.1 개발 일정

과제 수행을 진행했던 프로그램 개발 일정은 표 3과 같다. 주로 AttnGAN과 CycleGAN의 학습과 품질 향상에 많은 소요가 되었다. 그리고 예상외로, 다른 객체 AttnGAN 학습을 위한 이미지 데이터 수집과 텍스트 데이터 생성에서도 상당히 많은 시간을 소요했다. 본 보고서를 작성하고 있는 시점에도 품질 향상을 위한 학습이 진행 중이다.

표 3. 프로그램 개발 일정표



6.2 역할분담

6.2.1 박 성국

- CycleGAN 화풍 변환 작업
  - CycleGAN 라이브러리 테스트
  - 이미지 화풍 변환 테스트
  - 화풍 이미지 데이터 수집
  - CycleGAN 화풍 데이터 학습
  - CycleGAN 세잔, 모네, 고흐 화풍 적용
- AttnGAN 이미지 생성 작업
  - AttnGAN 이미지 품질 향상
- AttnGAN 한글화 작업
  - 영어 텍스트 데이터 한글화
  - AttnGAN 한글화
  - AttnGAN 다른 동물 한글 텍스트 데이터 생성
  - AttnGAN 동물 데이터 한글 학습
  - Konlpy 라이브러리 확인 및 변경

6.2.2 장 수현

- CycleGAN 화풍 변환 작업
  - 화풍 이미지 데이터 수집
  - CycleGAN 동물 이미지 데이터 수집
  - CycleGAN 화풍 데이터 학습
  - CycleGAN 수목화, 동양화 화풍 적용
  - CycleGAN 화풍 품질 향상
- AttnGAN 이미지 생성 작업
  - AttnGAN 다른 동물 이미지 데이터 수집
- AttnGAN 한글화 작업
  - AttnGAN 다른 동물 한글 텍스트 데이터 생성
- 웹 인터페이스 생성 및 모듈 통합 작업
  - 웹 인터페이스 생성
  - 웹 인터페이스, 모듈 통합
  - 서버 개설 및 웹 호스팅

6.2.3 박 창조

- AttnGAN 이미지 생성 작업
  - AttnGAN 라이브러리 테스트
  - AttnGAN 최소 데이터 set 측정
  - AttnGAN 기존 동물 이미지 생성 테스트
  - AttnGAN 다른 동물 이미지 데이터 수집
  - AttnGAN 동물 데이터 영어 학습
  - AttnGAN 이미지 생성 품질 향상
- AttnGAN 한글화 작업
  - AttnGAN 다른 동물 한글 텍스트 데이터 생성
  - AttnGAN 동물 데이터 한글 학습
- 환경 개선을 위한 작업
  - Google Cloud Platform 환경 구동 확인

7. 참고 문헌

[1] 표태준. (2018, January 19). "세 줄만 넘어가도 어찔… 혹시 당신도 긴글 까막눈?" [Online]. Available: https://news.chosun.com/site/data/html_dir/2018/01/18/2018011801596.html

[2] 통계청. (2020, January 4). 성인문해능력조사 [Online]. Available: http://kostat.go.kr/portal/korea/kor_pi/8/6/1/index.board?bmode=read&aSeq=379799&pageNo=7&rowNum=10&amSeq=&sTarget=&sTxt=

[3] 문화체육관광부. (2020, February 2). 국민독서실태조사 [Online]. Available: http://www.mcst.go.kr/kor/s_policy/research/researchView.jsp?kid=20031113581999862

[4] 장준혁. (2018, August 28). 새로운 인공지능 기술 GAN [Online]. Available: https://www.samsungsds.com/global/ko/support/insights/Generative-adversarial-network-AI-2.html

[5] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang and Xiaodong He. "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks", Proc. of the Computer Vision and Pattern Recognition(CVPR) Conference, 2018

[6] 김정은. (2018, January 19). "MS, 텍스트 설명을 그림으로 구현하는 AI 시스템 발표" [Online]. Available: https://www.thedailypost.kr/news/articleView.html?idxno=59916

[7] Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros and Alexei A. "Image-to-Image Translation with Conditional Adversarial Networks", Journal of Computer Vision and Pattern Recognition(CVPR), 2017

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.", IEEE International Conference on Computer Vision (ICCV), 2017.

[9] Bird, Steven, Edward Loper and Ewan Klein. "Natural Language Processing with Python". O'Reilly Media Inc., 2009

[10] 박은정, 조성준. "KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지", 제 26 회 한글 및 한국어 정보처리 학술대회 논문집, 2014.

[11] 구글 코랩. (2020, July 18). Colaboratory [Online]. Available: https://colab.research.google.com/signup

[12] 구글 핀터레스트. (2020, July 18). Pinterest [Online]. Available: https://www.pinterest.co.kr/

[13] 이미지넷. (2020, July 23). ImageNet [Online]. Available: http://www.image-net.org/

[14] 캐글. (2020, August 4). Kaggle [Online]. Available: https://www.kaggle.com/

[15] 구글. (2020, August 4). Google [Online]. Available: https://www.google.co.kr/

[16] 인스타그램. (2020, August 5). Instagram [Online]. Available: http://www.instagram.com/

[17] flask. (2020, August 26). "Flask web development, one drop at a time" [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/

[18] 파이썬애니웨어. (2020, August 27). PythonAnywhere [Online]. Available: https://www.pythonanywhere.com/

[19] 데이터사이언스스쿨. (2016, June 14). "KoNLPy 한국어 처리 패키지" [Online]. Available: https://datascienceschool.net/view-notebook/70ce46db4ced4a999c6ec349df0f4eb0/

[20] 이슬기. (2015, October 8). "보아도 상상할 수 없는 '아판타시아'" [Online]. Available: https://www.sciencetimes.co.kr/news/보아도-상상할-수-없는-아판타시아/