|  |
| --- |
| **Text to Image AI의 저작물 무단 학습 방지 모델 설계** |
| 김진호1, 한준희2  **1, 2민족사관고등학교** |
| **Designing prevention method of**  **Text to Image AI’s unauthorized learning** |
| **Jinho Kim1, Jooney Han2**  **1, 2Korean Minjok Leadership Academy, Hoengseong-gun, Gangwon-do, Korea** |
| **요 약** 본 연구에서는 Latent Diffusion으로 대표되는 Text to Image(TTI) AI 모델의 대량 데이터 수집 과정에서 발생하는 저작물 무단 학습 문제를 해결하고자 한다. TTI 모델은 아주 많은 양의 데이터를 학습하기 위해 무분별한 웹 데이터 크롤링을 진행하는데, 이때 원작자의 동의 없이 저작물이 모델 학습에 이용되는 일이 발생한다. TTI 모델은 미동의 저작물들의 그림체를 학습하고, 이를 바탕으로 새로운 이미지를 생성시키며 원작의 가치를 훼손시킨다. 이에, 본 연구자들은 저작물들에 대한 무단 학습을 막기 위해 두 가지 방법을 제시한다. 첫째, 일반적인 웹상에서 크롤링 되지 않고 특수 뷰어만을 이용해 열람할 수 있는 이미지 파일의 설계를 제시한다. 데이터를 암호화 알고리즘을 이용해 암호화하고, 이를 뷰어에서 복호화해 원본 데이터를 확인할 수 있도록 한다. 이를 통해 자동화된 데이터 수집을 차단하지만, 사용자들에게는 원본 이미지를 배포할 수 있도록 한다. 둘째, TTI 모델의 원작 이미지 학습을 막기위해 이미지를 변형하는 방법을 제시한다. 이로써 모델이 원 그림의 그림체를 학습하는 것을 방지하는 방법을 설계했으며, 디퓨전 모델에 본 연구에서 제시한 이미지 변형법이 적용된 이미지와 일반 이미지 데이터 세트를 각각 학습시키고 동일한 조건에서 산출되는 이미지의 퀄리티를 비교하였다. 그 결과 변형된 이미지를 학습한 모델은 학습 성능이 저하되어 미동의 저작물을 온전히 학습할 수 없음을 확인했다. 이에, 본 연구에서 제시하는 두 가지 방법이 모두 AI 모델이 무단으로 저작물을 학습하는 것을 방지함을 확인한다. |
| **Abstract** In this work, we aim to solve the problem of unauthorized learning of works arising from the process of collecting large amounts of data from Text to Image (TTI) AI models represented by Latent Diffusion. The TTI model performs indiscriminate web data crawling to collect substantial number of images, and these images are used for model learning without the consent of the original author. The TTI model is capable of learning the drawing style of an image, which undermines the value of the original work. Therefore, we suggest two methods to prevent unauthorized learning of literary works. First, we present the design of a new image file extension that is protected from the mass data collection process by an encryption algorithm and can be decrypted and viewed using only a special viewer. This prevents automated data collection but allows users to distribute the original image. Second, we present a method of transforming images to deteriorate the learning accuracy of TTI models. Then, we compare the quality of original images to images processed by the modification method presented in this study. Thus, we confirm that both methods prevent AI models from learning literary works without permission.  **Key Words: Latent Diffusion, Text to Image AI, Image Format, Image Quality Metric, Encryption** |

1. 서론

1.1 연구 배경

2022년 *Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models."*연구의 발표 이후로 다양한 Text to Image (TTI) AI 들이 배포되기 시작했다.[1] 기존의 GAN (Generative Adversarial Network) 모델을 이용하지 않고 새롭게 Diffusion 모델을 이용해 Stability AI사에서 개발된 Stable Diffusion AI는 150,000 GPU 시간동안 학습된 모델 전체와 소스코드가 무료로 공개되었을 뿐 아니라, 기존의 DALL-E 등의 유료 모델과 비교했을 때도 성능의 우위를 점하게 되었다.[2] 또한, 일정 라이선스와 함께 모델을 원하는 대로 정하는 Fine-Tuning 역시 허용하며, Stable Diffusion 모델을 채택한 다양한 인공지능들이 나오게 되었다. 하지만, 이러한 인공지능 모델들의 학습 과정에서 저작권 문제가 제기되었다. 학습에 사용된 그림, 삽화, 사진들이 결국 창작자의 저작물이며, 대부분의 경우 사용에 대한 동의를 구하지 않고 학습되었기 때문이다. 특히, 애니메이션 스타일(Anime Image)의 그림을 생성하는데 최적화 되어있으며 유료 서비스로 운영되는 Novel AI Image Generator의 경우, 창작 일러스트 아카이브인 Danbooru의 그림들을 학습했다고 알려져 있으며, 이 과정에서 원작자들의 동의 없이 그들의 그림을 학습에 이용했다.

이렇듯 이미지 생성 기술의 발전과는 반대로 동의없이 저작물이 학습에 이용되는 현상속에서, 본 연구에서는 원작자들의

권리와 창작물을 보호하고자 하였는데, 원작자가 자신의 작품이 학습에 이용되는 것을 원치 않는다면 간단하지만 효과적인 방법으로 무단 학습을 방지할 수 있는 방법을 개발하고자 하였다.

1.2 선행 연구

TTI AI에 관한 연구는 2022년을 기점으로 활발히 진행되고 있다. 기존의 GAN기법 기반의 생성 모델은 2014년을 기점으로 본격적인 연구가 진행되었다. *Goodfellow, Ian J., et al. "Generative adversarial networks." arXiv preprint arXiv: 1406.2661 (2014).* 에서는 GAN 모델을 수학적으로 설계, 구현하고, MNIST, TFD, CIFAR-10 데이터셋을 이용해 실험적인 생성모델을 구축하는데 성공했다.[3] 이후 DCGAN, PG-GAN, BigGAN, StyleGAN 순으로 개선된 형태의 GAN 모델들이 개발되었고, 모델의 성능은 점차 증가했다.[4-7] 하지만, GAN을 이용한 생성모델의 고질적 문제점인 불안정한 학습, 쉬운 오버피팅, 다양성 부족 등의 문제가 제기되었고, 이는 Latent Diffusion 모델의 개발로 이어졌다. [7-8]

이러한 이미지 생성 AI의 개발 과정중, 모델의 자동화 학습에서 부적절한 이미지를 학습하지 않도록 하는 기법은 개발된 바 있는데, 예를 들어 모델에서 오버피팅을 일으킬 가능성이 높거나, 불법적이고 불쾌감을 유발할 수 있는 사진들을 학습 과정에서 차단하는 것이다. 하지만, 반대로 창작자의 저작물이 학습되는 것을 막는 방법에 대해서는 연구된 바 없다. 이는 본 연구의 진행시기 기준으로 최근에 서야 Stable Diffusion의 공개와 함께 TTI AI가 널리 사용되기 시작했으며, 이 과정에서 기술적인 측면에 대해서만 연구가 진행되었을 뿐, 부작용들에 대해서는 논의가 부진했기 때문으로 사료된다.

2. 이론적 배경

2.1 Text to Image AI의 작동 원리 분석

2.1.1 GAN 모델을 이용한 Text to Image AI

GAN(Generative Adversarial Networks) 모델은 2014년 최초로 발표되어 현재까지 Text to Image AI 분야에서 널리 사용되는 인공지능 모델이다. GAN 모델은 크게 Generator(생성자, G)와 Discriminator(판별자, D)의 두 부분으로 나눌 수 있다. 생성자 G와 판별자 D는 학습 과정에서 적대적으로 작용한다. 모델에서 D는 G가 생성한 가짜 샘플과 훈련 데이터셋의 진짜 샘플을 구별하는 역할을 한다. 이때 D에는 훈련 데이터셋에 존재하는 실제 샘플 또는 생성자가 만든 가짜 샘플이 입력되며, 출력값은 입력 샘플이 실제 샘플일 확률이 된다. G는 랜덤한 숫자로 구성된 벡터를 입력으로 받아 최대한 G가 진짜 이미지라고 판별할 수 있는 샘플을 출력하게 된다. 이때, D는 G가 생성한 이미지가 실제 샘플일지 판단해 0에서 1사이의 값을 출력하고, 출력값은 다시 G에 backpropagation 되어 G는 생성 횟수가 늘 수록 더욱 실제같은 출력값을 만들게 된다.[4]

이 과정에서, D는 참 또는 거짓을 판별하기에, 손실함수로 이진분류 함수를 사용하며, 본 연구에서 사용된 손실함수인 Binary Cross Entropy(BCE)의 식을 수식으로 정리하면 다음과 같다.



Figure 2 a. 원본 석양 사진

GAN은 최초 발표를 시작으로 계속해서 개선된 모델이 개발

되었다. 심층 컨볼루션 신경망을 이용한 DCGAN, 대규모 학습을 이용한 BIGGAN, Style-based 생성자 아키텍처를 고안한 Style GAN 등이 그 예이다. 하지만, GAN 모델을 이용한 이미지 생성 AI는 학습과정에서의 높은 오버피팅 가능성과 완전히 새로운 이미지 생성의 한계, 높은 하드웨어 리소스 비용 등으로 인한 문제에 봉착하게 되었다.[4-7]



Figure 2 b. Encoding 과정을 거친 사진

2.1.2 Latent Diffusion을 이용한 Text to Image AI

상술한 GAN 모델의 단점을 개선하고 보다 효과적인 이미지 생성을 위한 AI 모델로서 Latent Diffusion이 제시되었다. Latent Diffusion은 2022년 LMU 뮌헨 대학교와 하이델베르크 과학연산센터의 합동 연구로 개발된 이미지 생성 모델이다.[9] Latent Diffusion은 기존 GAN 모델과 매우 다른 학습 방식과 설계를 갖는다. Latent Diffusion 모델의 학습은 먼저, 원본 데이터에 노이즈를 삽입하여 시작된다. 아래는 이미지에 노이즈가 삽입되는 과정을 나타낸 그림이다.

|  |  |
| --- | --- |
| Picture a 원본 이미지 | Picture b 25% 노이즈 |
| Picture c 80% 노이즈 | Picture d 100% 노이즈 |

Figure 1 원본 이미지에 노이즈를 추가하는 과정

Latent Diffusion 모델은 [Figure 1]과 같이 원본 이미지에

서 시작해 점차적으로 전체가 모두 노이즈인 이미지를 생성한 데이터를 이용해 학습된다. 이 과정을 모델에 학습시키고, 훈련 과정에서는 모델에게 원본 이미지에 대한 정보를 입력하고, 역순으로 이미지를 복구하는 작업을 요구한다. 적은 노이즈의 이미지에서 시작해, 점차 노이즈를 늘리고, 결국 전체가 모두 노이즈인 이미지를 제공하게 되는데, 이러한 과정을 거친 모델은 최종적으로 원본 이미지에 대한 시각적 정보가 전혀 없는 노이즈에서도 원본 이미지를 추측해, 이미지를 생성할 수 있는 능력을 갖추게 된다. 노이즈 추가 과정에서는 후술할 Gaussian Noise가 이용된다. [10]

또한, 기존의 Diffusion 모델과는 차별적으로 Latent Diffusion 모델에서는 Latent Space Encoding 과정이 포함된다. 이는 이미지의 특성과 구획을 분석해 특징적인 부분을 파악, 이를 행렬로 나타내는 과정이다. 다음 사진은 Latent Diffusion 논문에서 발췌한 석양 사진이 Encoding 과정을 거친 모습이다.



Figure 2 c. Encoding 된 사진을 바탕으로 생성한 사진

Stable Diffusion 모델의 학습과정에서는 LAION-5B 데이터셋이 사용되었다. LAION-5B 데이터셋은 CLIP 되어있는 데이터셋으로, 58억개 이상의 이미지-텍스트 데이터로 구성되어있다. [11] 본 연구진은 본격적인 모델의 사용에 앞서, 데이터셋이 어떠한 데이터들로 이루어져 있는지 확인했다. 이는 LAION 자체적으로 지원하는 백엔드 URL을 이용해 사용할 수 있었다.

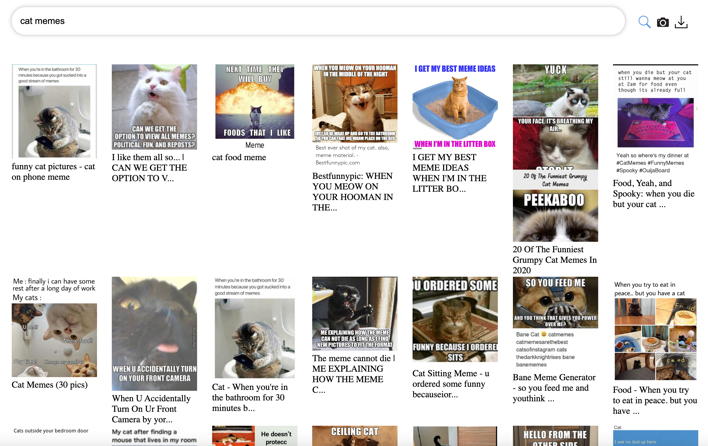


Figure 3 'Cat memes'를 키워드로 데이터셋에 검색한 결과

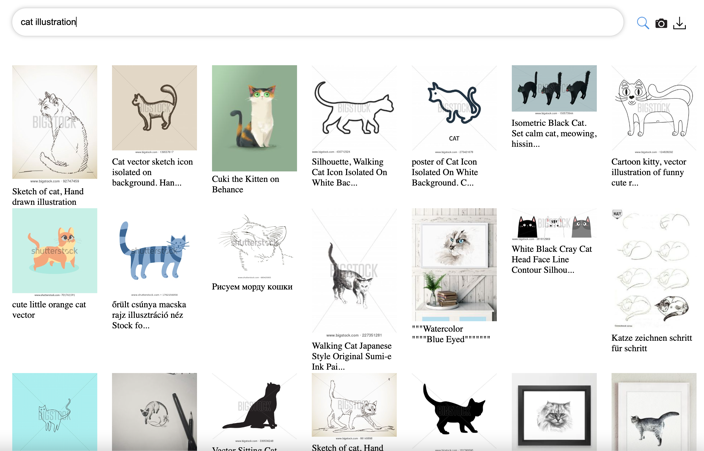


Figure 4 'Cat illustration’을 키워드로 데이터셋에 검색한 결과

상술한 바와 같이 LAION-5B 데이터셋은 다양한 웹 상의 자료들을 indiscreet한 방법으로 크롤링함을 확인할 수 있었다. 개인의 창작물인 고양이 일러스트들 뿐 아니라, 유머의 소재로 사용되는 ‘meme’ 또한 데이터셋에서 찾을 수 있었다. 또한, 일부 이미지 데이터의 Text CLIP에서 웹 주소 또한 발견할 수 있었는데, 이는 웹 상에 업로드된 자료들을 데이터 수집 과정에서 모두 저장한다는 가설에 대한 방증으로 판단하였다.

또한, Latent Diffusion 모델은 Fine-Tuning의 사용이 가능하다. Fine-Tuning이란, 사용자가 자신의 목적에 맞춰 생성하고 싶은 물체, 배경, 그림체 등을 기존 모델에 추가적으로 학습시키는 과정이다. 하지만, 이는 써드 파티로서 Latent Diffusion을 이용한 서비스를 제공하는 플랫폼에서 무단으로 웹툰, 일러스트 등의 데이터를 추가로 무단 학습시켜 자신만의 서비스를 구축 하기도한다.

상술한 특성들을 갖는 Latent Diffusion을 실제로 구현한 모델은 Stable Diffusion 이란 이름으로 배포되어 있다. 이는 미국의 AI 전문 기업인 StableAI와 데이터셋 LAION의 후원으로 구현된 모델로, 현재 가장 널리 사용되고 있는 Latent Diffusion 모델의 구현체이다.

본 연구에서 목표하는 이미지 무단 학습 방지를 구현, 설계하는 과정에서는 Stable Diffusion 모델로 연구가 진행되었다. 이는, Stable Diffusion의 성능이 매우 강력하며, 오픈소스화 되어 이를 이용한 다양한 서비스들이 배포되고 있기 때문이다. 또한, GAN 기법의 경우 파생 기법이 너무나 다양하고, 일정한 학습 데이터셋이 존재하지 않으며, Fine-tuning 과정 또한 많은 하드웨어 리소스를 요구하기 때문에 적합하지 않다고 판단했다.

3. 연구 방법

3.1 Stable Diffusion 모델의 저작물 학습 방지법 설계

3.1.1 Stable Diffusion과 DreamBooth의 사용

Stable Diffusion 모델은 현재 오픈소스로써 완전히 공개되어 있다. 본 연구에서는 Stable Diffusion을 활용한 연구의 효율성을 높이고, Fine Tuning이나 생성 과정에서 파라미터를 자유롭게 조작하기 위해 웹 상에서 Stable Diffusion을 활용할 수 있는 Stable Diffusion WebUI 를 활용했다.

또한, 연구 과정 중 Fine-Tuning을 활용하기 위해 DreamBooth 기법을 활용했다. DreamBooth는 *Ruiz, Nataniel, et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation." arXiv preprint arXiv:2208. 12242 (2022).* 논문에서 발표된 기법으로, Stable Diffusion 모델 환경 상에서 Fine-Tuning을 보다 적은 수의 추가 데이터로도 효과적으로 실행할 수 있도록 한다.[12]



Figure 5 선글라스의 사진을 이용해 Fine-Tuning한 결과물

[Figure 4] 에서는 선글라스 사진 4장을 이용해 선글라스 객체에 라는 ‘class name’을 부여하고, ‘A sunglasses in the jungle’ 과 같은 prompt 로 이미지를 생성한다. 이에, 본 연구에서는 Fine-Tuning 을 활용, 추가 데이터를 변형한 후에도 그림체나 화풍, 또는 일러스트 내 요소들을 생성모델이 학습할 수 있는지 확인한다.

3.1.2 이미지 변형을 통한 학습률 저하

상술했듯, 본 연구에서는 크게 두 가지 방법을 이용해 이미지가 무단 학습되는 것을 막고자 한다. 이중 첫번째로는 Stable Diffusion의 특성을 역이용하여 학습을 막는 방법을 제시한다. 이 과정에서 이미지를 허용된 사용자는 인식하는데 어려움이 없지만, 학습 과정에서 AI 모델은 어려움을 겪도록 이미지를 변형하고자 한다. 이를 위해 본 연구진은 다양한 가설을 세우고, 이에 대응하는 이미지 변형 프로그램을 개발하였다.

먼저, 사람이 이미지를 인식할 때 어떤 과정을 거쳐 이미지에 대한 인상을 갖게 되는지 연구를 진행하였다. 선행 연구들에 의하면, 이미지에 소멸 또는 생성 효과가 적용되는 경우에는 비전형적 (Atypical) 시선을 통해 사물을 인식하게 된다. 또한, 색체의 대비가 키거나, 착시를 일으키거나, 처음 보는 사물, 또는 미술관의 미술작품 등 의식적으로 자세히 이미지를 확인하려고 하는 경우에는 이미지를 전체적으로 훑는 과정을 거치게 된다. 반대로 사람의 얼굴을 인식하는 경우, 큰 어색함이 없는 경우 무의식적으로 뇌에서 자세한 인식 과정을 포기하고 미간에 시선이 집중되어 인물을 인식하게 된다. [13-17] 본 연구에서 다루는 디지털 일러스트의 경우, 연구들에 따르면 미술 작품들과는 달리 사람이 의식적으로 자세히 확인하려 하지 않고, 상술했듯 미간을 응시하며 대략적으로만 사물을 인식한다. 이러한 점을 인지한 후, 사람이 일러스트에서 인물과 물체를 인식하는데 어려움을 최소화 시키되, Stable Diffusion 모델에서 학습될 때 일러스트가 정상적으로 학습되는 것을 방지해 이미지를 보호할 수 있도록 변형하고자 하였다.

이에, 본 연구자들은 고안한 방법을 통해 이미지를 변형시키고, 해당 방법으로 변형된 이미지가 Stable Diffusion 모델에서 얼마나 학습효율을 감소시키는지 확인하였다. 이미지를 변형한 방법은 다음과 같다.

앞서 설명한 원활한 Latent Space Encoding을 차단하기 위해 이미지 내 객체들의 테두리를 변형시킨다. 객체의 테두리를 인식하고, 특정한 부위의 테두리를 늘리거나, 두껍게 하는 등 이미지를 변형시켜 encoding 과정을 어렵게 만든다.

가장 먼저, 이미지의 테두리를 인식하기 위해, Canny Edge Detection 알고리즘을 활용하였다.

모서리를 구분하는 Edge Detection 알고리즘 중 Canny Edge detection 알고리즘은 모서리 탐색에 방해가 되는 노이즈를 제거하기 위해 가우스 필터로 이미지의 노이즈를 줄이는 작업을 진행한다. 이후 이미지의 전체적인 강도를 계산하고, 강도가 급격히 변하는 지점을 minimum threshold, maximum threshold에 따라 모서리로 판별한다. [18-19]

이렇게 Canny Edge Detection을 통해 이미지의 요소별 테두리를 확인한 후, 해당 테두리의 두께를 랜덤하게 변경하거나, 지우는 등의 과정을 거쳤고, 이렇게 얻은 이미지를 Fine Tuning을 통해 Stable Diffusion 모델에 학습시켜, 이미지의 재생성에 어색함이 있는지 확인했다.



Figure 6 사진의 테두리를 인식한 사진

블러 효과를 가하는 프로그램을 구현하기 위해 블러 효과를 가하는 방법에 대해 연구를 진행하였다. 일반적으로 블러 효과는 특정한 parameter로 N이 주어졌을 때, 한 픽셀의 값을 주변 픽셀의 값의 평균을 계산해 정하게 된다. 본 프로그램에서 의도한 바와 같이 물체의 테두리를 인식하고, 그 테두리 주변에 블러 효과를 가하기 위해서는 2가지 변수를 관리하고자 이 과정에서 두가지 parameter를 관리하고자 하였다. 먼저, kernelSize 변수를 도입했다. kernelSize 변수는 한 픽셀의 주변의 몇개의 픽셀의 값을 평균 계산 과정에서 포함하는지에 대한 변수이다. 상술했듯, 기준 픽셀 주변의 픽셀의 값의 평균을 계산해, 기준 픽셀의 값을 정하기 떄문에 kernelSize 변수를 통해 얼마나 넓은 범위의 픽셀 값을 평균치로 계산해 적용할지 정하게 된다. 이때문에 kernelSize parameter 값을 키우면 이미지가 보다 부드럽게 변형되므로, PSNR과 SSIM 수치를 상승시키는 작용을 한다. 두번째로, nearbyBlurSize 변수를 도입하였다. nearbyBlurSize 변수는 테두리를 기준으로 몇개의 픽셀에 블러 효과를 적용하는지에 대한 변수이다. 따라서 nearbyBlurSize 값을 키우면 PSNR과 SSIM 수치를 낮추는 기전으로 작동한다.

아래는 이미지의 테두리에 블러 효과를 적용해 변형시키는 역할을 하는 Python 코드의 일부이다.

|  |
| --- |
| (…)  kernelSize = 5  nearbyBlurSize = 5  term = (kernelSize//2) + (nearbyBlurSize//2)  for x in range(term, cannyimg.shape[0]-term):  for y in range(term, cannyimg.shape[1]-term):  if cannyimg[x][y] == 255:  for t in range(x-(nearbyBlurSize//2), x+(nearbyBlurSize//2)):  for c in range(y-(nearbyBlurSize//2), y+(nearbyBlurSize//2)):  for p in range(3):  sum = 0  for i in range(kernelSize):  for j in range(kernelSize):  sum += orig\_img[t+i-term][c+j-term][p]  orig\_img[t][c][p] = sum // (kernelSize\*kernelSize)  (…) |

Code 1 이미지 테두리에 블러 효과를 적용하는 코드

Code 1은 이미지의 테두리를 인식하고, 블러효과를 적용시키는 Python 코드이다. kernelSize 변수를 높게 설정하면, 테두리를 기준으로 블러효과의 영향을 받는 주변 픽셀수가 증가하게 된다. 이 때문에 픽셀 수가 많은 이미지들의 경우kernelSize 변수의 값을 높이면 블러 효과를 확인할 수 있다. 다음 사진은 Figure 1 에서 사용된 사진을 Canny Edge Detection을 통해 테두리를 인식시키고, 블러효과를 가한 사진이다.



Figure 7 [Figure 6]의 사진 테두리에 블러 효과를 적용한 사진

[Figure 7]에서 확인할 수 있듯, 테두리에 일정한 파라미터를 이용해 테두리를 변형시켰을 때, 그 차이는 맨눈으로 확인하기 쉽지 않다. 하지만, 원본 그림과 변형된 그림을 각각 확대하면, 그 차이는 명확히 확인 가능하며, 이러한 차이를 통해 인공지능의 무단 학습을 막고자 하였다. 아래에서는 원본 사진과 변형된 사진을 확대해 비교한다.

|  |  |
| --- | --- |
| Picture a 원본사진을 확대한 모습 | Picture b 변형된 사진을 확대한 모습 |

Figure 8 원본이미지와 변형된 이미지를 확대해 비교한 모습

[Figure 8]의 [Picture b]는 Code 1에서 kernelSize를 5, nearbyBlurSize를 3으로 설정해 변형한 사진이다. Figure 2와 Figure 8에서 볼 수 있듯, 이미지를 변형했을 때, 원본 상태에서는 일반적으로 사용자가 이미지의 변형 여부를 판가름하기 어려우며, 어색함을 느끼기 힘들다고 정성적으로 분석할 수 있었다.

원본 이미지와 변형된 이미지의 품질을 정량적으로 분석하기 위해 PSNR (Peak Signal-to-Noise Ratio) 수치와 SSIM (Structural Similarity Index Measure) 수치를 도입하였다.

PSNR은 화질의 손실 정도를 파악하기 위해 사용된다. PSNR의 계산을 위해서는 MSE(Mean Squared Error) 값이 활용된다. MSE는 예측 값 와 parameter 간의 차이를 제곱해 더하고, 이 값의 평균을 계산한 것인데, 이미지에서는 각 픽셀간 값의 차이를 계산해 사용된다. MSE와 PSNR 수치를 구하는 식은 다음과 같다.[20-22]

식에서 확인할 수 있듯, PSNR 수치는 MSE가 최소, 즉 PSNR 값이 커질 수록 이미지가 원본 사진과 유사하다는 것을 뜻하며, 완전히 동일한 두 사진은 무한대의 PSNR 값을 가지게 된다.

SSIM은 시각적인 화질 차이와 유사도를 파악하기 위한 방법으로, 휘도 (luminance), 대비(contrast), 구조적 차이(structure) 값을 활용한다. PSNR 수치는 보다 데이터, 수치적으로 계산하는데 반해 SSIM은 휘도, 대비 등의 시각적인 요소를 활용해 계산한다는데 그 차이가 있다. [21-23]

앞서 설명한 PSNR 수치와 SSIM 수치를 이용해 변형된 사진의 품질에 대한 정량적 평가를 진행하였다. 이를 비교한 값은 다음과 같다. 유효숫자는 소수점 아래 세 자리로 설정하였으며, 실험에 사용된 값은 19801080 픽셀의 [Figure 1] 그림이다. 해당 실험의 결과를 정리할 때, 변형도는 Code 1의 kernelSize와 nearbyBlurSize 의 순서로 이루어진 의 튜플로 표기한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 변형도 | (3, 3) | (5, 3) | (3, 5) | (5, 5) |
| PSNR (dB) | 33.371 | 36.499 | 25.594 | 26.311 |
| SSIM | 0.988 | 0.993 | 0.957 | 0.960 |

Table 1 이미지 변형도에 따른 이미지 품질 수치

위 그림들과 PSNR 수치에서 확인할 수 있듯, 변형도가 바뀜에 따라 이미지의 품질 수치 또한 예상했던 경향성 대로 증가, 감소하고 있음을 확인할 수 있다. 하지만, 보편적으로 100만 화소 이상의 이미지에서 SSIM 수치 0.99~0.97 사이의 이미지는 차이를 구분하기 어렵고, 0.95 이상의 수치를 갖는 이미지를 준수한 품질로 평가하는 만큼 손상 정도가 크지 않고, 사용자가 어색함 없이 이미지를 확인할 수 있을 것이라 판단할 수 있다. [20-23] 이후, 이렇게 변형된 이미지를 학습한 이미지 생성 모델이 정상적으로 비슷한 형태의 이미지를 생성하는지를 확인해 해당 변형법이 효과적인지 확인하고자 하였다.

3.1.3 G-FID 수치를 이용한 생성된 이미지의 품질 평가

본 연구에서는 이미지를 변형시키는 과정에서, AI 모델의 이미지 학습은 방지하되, 동시에 일반적인 사용자들이 이미지를 인식하는데 큰 불편함을 겪지 않는 것을 목표하였다. 이를 위해, 상술했듯 사람이 이미지를 인식하고 인상을 결정하는데 어떤 주요한 과정을 거치는지 확인했다.

이후, 품질 평가를 위해 변형된 이미지가 인식 과정을 방해할 요소가 없는지 정성적으로 확인하였다. 또한, 정량적 평가를 위해 이미지 평가 수치들 도입하였다. 앞서 설명한 PSNR 수치와 SSIM 수치 대신, 생성된 이미지의 성능 평가에 보다 적합한G-FID(Generation Fréchet Inception Distance) 수치를 도입하였다. G-FID 수치는 2017년 개발된 이미지 생성 모델에서의 출력 이미지의 품질 평가를 위해 사용되는 척도이다. 영상 집합 사이의 거리를 나타내는 수치인데, 생성된 영상의 집합과 생성하고자 하는 데이터의 분포상의 거리를 계산하게 된다. FID 수치를 계산하는 식은 다음과 같다. 이때, 는 실제 영상 데이터, 는 생성된 영상, 는 평균, 는 공분산을 나타낸다.[24-25]

이 외에도, 생성 이미지 데이터를 학습한 InceptionV3 모델을 이용해 계산되는 Inception Score 또한 이용해 생성된 이미지의 품질을 보다 객관적으로 평가하고자 하였다.

3.2 이미지 배포 방지를 위한 뷰어와 이미지 확장자 설계

3.2.1 이미지 확장자의 종류와 특징

이미지를 저장 및 배포하기 위해서는 보통 이미지의 용량과 처리 성능을 고려하여 압축을 진행한다. 이미지는 확장자마다 압축 방식과 데이터 저장 방식이 다른데, 예를 들어 웹상에서 흔히 사용되는 .jpeg 혹은 .jpg 이미지 형식은 손실 압축 방식으로 사용자가 이미지를 인식하는데 지장이 적거나 없는 부분에 한해 원본 이미지를 약간 훼손하여 품질은 낮지만 압축 성능이 매우 뛰어나 이미지를 전송하거나 저장할 때 주로 사용된다. 반면 .png 이미지 형식은 무손실 압축 방식으로, 원본 이미지 데이터를 그대로 보존하는 대신 필연적으로 손실 압축 방식 대비 처리 속도의 저하와 용량의 증가가 발생하게 된다. 하지만 세밀한 곡선의 표현 등이 뛰어나므로, 디자인과 로고 등의 배포에서 주로 사용된다.

본 연구에서 설계한 이미지 뷰어는 데이터 크롤링을 철저히 방지하기 위해 이러한 기존의 이미지 파일 확장자들과는 다른, .jhp 라는 새로운 무손실 압축 방식의 이미지 확장자를 개발하였으며, 후술할 압축 알고리즘과 암호화 알고리즘을 통해 오로지 본 연구에서 제시된 뷰어 만을 사용하여 이미지를 렌더링 할 수 있도록 하였다.

3.2.2 이미지 뷰어의 압축 알고리즘 설계

이미지 데이터를 그대로 보존하면서 데이터에 대한 암호화까지 진행하게 되면 이미지 파일의 용량이 raw data에 비해 대단히 증가하게 된다. 이를 보완하고자 본 연구에서는 무손실 데이터 압축 알고리즘을 활용해 .jhp 이미지 형식의 설계에 적용하였다.

이를 위해 압축 과정에서는 허프만 코딩과 LZ77압축 기법으로 구성된 문자열 압축 알고리즘을 구성해, 문자열을 최대한 축소화 시킨 후 평문이 아닌 형태로 압축시키는 과정을 거치도록 하였다. 이중 LZ77 기법은 Search, Lookahead 두 개의 버퍼로 문자열을 순회하며 반복되는 부분을 찾아 문자의 인덱스 번호를 사용하여 압축한다. 이렇게 압축된 데이터를 허프만 부호화 압축 방식을 통해 단어 출몰 빈도수를 기반으로 이진 트리를 형성하여 압축하였다. [26-29]

3.2.3 AES 알고리즘의 구조와 설계

AES 알고리즘은 Advanced Encryption Standard 의 약자로, NIST의 공모를 통해 2001년 발표된 알고리즘이다. 이 과정에서 최종적으로 선정된 Rijndael 알고리즘을 AES라는 이름으로 발표하였다. 암호화 키는 128, 192, 256비트 총 3가지가 있으며, Rijndael 알고리즘은 한 라운드에 크게 4단계를 거치게 된다.[30-31]

가장 먼저, Key Scheduling 과정을 거친다. 이때 하나의 메인 암호화 키로부터 많은 라운드 키들을 만들어 내게 된다. 주 키의 길이 옵션에 따라 총 라운드 수가 달라지므로 생성되는 라운드 키의 개수 또한 다르다. AES-128은 11개, AES-192는 13개, AES-256은 15개의 라운드 키를 생성한다. 이후, SubBytes 과정을 거쳐 Sbox 라는 비선형 대칭표를 이용해 값을 치환하고, ShiftRows 과정에서는 상태행렬의 행 별로 Circular array rotation을 왼쪽 방향으로 진행한다. 그 다음으로 MixColumn 과정을 거친다. 현재 상태행렬의 각 열에 대해 주어진 유한체 Symbol의 요소와 행렬곱셈을 수행한다. 마지막으로는, AddRoundKey 과정을 거친다. 상태행렬의 모든 바이트를 Key Scheduling 과정에서 생성된 라운드 키들과 XOR 연산을 진행한다. 이 과정을 실제로 C 언어를 이용해 구현하였지만, 전체 코드는 생략하고, 이를 쉽게 알아볼 수 있도록 의사코드로 대체해 작성한다. 아래는 AES 알고리즘의 과정을 간단하게 나타낸 의사 코드이다.

|  |
| --- |
| FOR round = 1 TO totalRound STEP 1  subBytes(stateMatrix)  shiftRows(stateMatrix)  mixColumns(stateMatrix)  addRoundKey(stateMatrix, totalRound)  ENDFOR  NEXT round  subBytes(stateMatrix);  shiftRows(stateMatrix);  addRoundKey(stateMatrix, totalRound); |

Code 2 AES 알고리즘의 과정을 나타낸 의사코드

3.2.4 이미지 뷰어 서비스의 설계 및 구현

3.2.2, 3.2.3 에서 설명한 바와 같이 이미지 뷰어를 설계하여, 이를 실제로 사용자들이 활용할 수 있도록 서비스 구조를 두 가지 구조로 구상하였다.

첫번째 방법으로는, 암호화 과정을 거치지 않고 단순히 이미지 데이터만을 포함하는 파일을 생성하는 구조이다. 이러한 방법으로 구현 시, 전용 뷰어에서만 파일을 열 수 있음은 동일하지만, 높은 압축률로 포맷을 구현 가능하고, 따라서 얻어지는 용량의 절약이 가능하다.

이를 통해 추후 웹 프론트엔드 라이브러리 등으로의 개발을 통해 API 키를 이용해 암호화, 복호화 과정을 진행하도록 설계하여 보다 더 높은 범용성과 편리성을 확보함과 동시에, 목표했던 웹 크롤링 방지를 달성할 수 있을 것으로 사료된다. 이미지를 뷰어 라이브러리로만 확인할 수 있게 제한하면, 이미지 데이터의 렌더링을 클라이언트 사이드에서 진행하고, 이는 곧 이미지 검색 스크립트의 SEO (Search Engine Optimization) 성능 저하를 유발한다. 따라서, 자동화된 이미지 크롤링 스크립트에서 이미지가 검색되는 것을 방지해, 학습데이터에 무단으로 사용되는 것을 방지할 수 있다. [32-33]

이를 구현하기 위해, 본 연구에서는 직접 .jhp 파일 형식을 제정하고, 표준화된 규약을 확정하였다. 데이터의 압축 전, 평문 상태의 데이터에서 첫 12 바이트를 채널, 가로 화소 수, 세로 화소 수로 설정해 이미지 파일을 열고 저장하는데 문제가 없도록 했다. 이미지의 형식을 저장한 후에는 각 픽셀별 값을 저장하도록 했는데, 이 과정에서 사용되는 자리수를 최대한 줄이기 위해 허프만 코딩 기법을 활용했다. 이렇게 이미지 데이터가 평문상태에서 완성되면, 해당 데이터를 압축하게 된다.

두번째 방법으로는, 사진 데이터를 암호화해 서버와 클라이언트가 주고 받는 구조이다. 이러한 방법으로 구현 시, 개발한 이미지 확장자를 제 3자가 열어 확인할 수 있는 뷰어 구현에 성공하더라도, 크롤링할 수 없게 된다. 또한, 웹툰, 일러스트 등 민감한 저작물을 전송할 때 본 프로토콜이 활용될 가능성이 존재한다. 3.2.3에서 설명한 AES 알고리즘을 이용해 설계한 암호화 통신 프로토콜의 수행 절차는 다음과 같다.

1. 클라이언트-서버간 연결 시작 후, 클라이언트의 공개 키를 서버로 전송
2. 서버단에서 적절한 만료 기간을 갖는 안전한 AES 세션 키를 생성
3. 서버단에서 생성된 키의 Blob을 클라이언트 공개 키를 이용해 암호화
4. 서버가 암호화된 Blob을 클라이언트에게 전송
5. 서버로부터 받은 Blob을 클라이언트의 비밀 키로 복호화
6. 클라이언트가 적절한 응답을 암호화해 서버로 전송
7. 공유된 세션 키를 이용해 서버와 클라이언트가 통신을 시작

물론 이 외에도 클라이언트 요청의 신뢰도 검사 등의 추가적인

구현이 필요하지만, 위와 같이 간단한 7단계를 거쳐 암호화 된

통신을 이용해 매우 안전하게 이미지 데이터를 주고받을 수 있

는 시스템을 개발할 수 있다.

4. 실험 설정

4.1 이미지 변형을 통한 학습 방지법 실험

4.1.1 Stable Diffusion 실행환경 설정

먼저, 이미지를 생성하는 Stable Diffusion 기법을 대중적으로 쉽게 사용할 수 있는 Stable Diffusion 기법을 쉽게 사용할 수 있는 환경을 설정하고자 하였다. 본 연구에서는 이를 위해 오픈소스로 공개되어 높은 신뢰성을 보이는 Stable Diffusion WebUI를 활용했다. 또한, Stable Diffusion을 보다 효과적인 하드웨어 환경에서 사용하기 위해 Google의 Colaboratory Pro 환경을 이용했다. 이 과정에서 사용된 GPU는 Nvidia A100 모델이다. 아래 사진은 Google Colaboratory 플랫폼을 이용해 Stable Diffusion WebUI를 실행시킨 초기 화면이다.

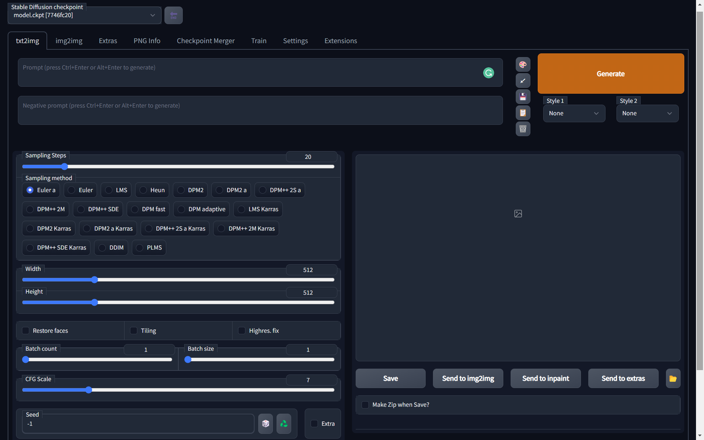


Figure 9 Stable Diffusion WebUI 실행 화면

Stable Diffusion을 이용해 이미지를 생성하기 위해서는 다양한 parameter들의 설정 과정을 거쳐야 한다. 본 연구 과정에서는 이미지 생성에 걸리는 시간과 하드웨어 리소스, 그리고 효율성을 고려해 batch size 1, batch count 1, sampling steps를 20으로 설정하고 실험을 진행하였다.

4.1.2 Fine-Tuning 실행환경 설정

상술했듯, Stable Diffusion 모델에는 직접 찍은 사진이나, 일러스트들을 이용해 신경망을 튜닝할 수 있다. 이는 앞서 언급한 논문에서 제시 되어있는데, 이것이 DreamBooth를 이용한 Fine-Tuning 과정이다. 이 과정에서 실험의 오차를 최소화하고, 학습의 효율성을 확보하기 위해 여러 통제변인을 설정하였다. 가장 먼저, DreamBooth를 통한 Fine-Tuning을 진행할 때 HuggingFace에 업로드 되어 있는 stable-diffusion 모델의 1.5 버전을 사용하였다. 또한, 파인 튜닝 결과물 모델 또한 모든 데이터셋을 하나의 모델로 만드는 것이 아닌, 데이터셋마다 각각 하나의 모델을 생성함으로써 가중치의 변동으로부터 생기는 오차를 방지하고자 하였다. 이에 더해, 이미지 크기를 통일하였는데, 512\*512 크기의 이미지는 처리 속도도 빠름과 동시에 학습 결과물의 특성을 손실없이 나타낼 수 있는 크기라고 판단하였으며, 선행연구 상에서 제시하였기에 이와 같이 사용하였다. 마지막으로, 파인 튜닝 모델 생성과정의 training steps와 learning rate을 통일하였는데, 학습 결과에 이상치가 발현되는 것을 최대한 방지하고자 stable diffusion 모델에 대한 활용 방법과 심층적 연구를 진행하는 기관인 HuggingFace에서 여러 번의 실험을 통해 도출한 최적의 값인의 learning rate 와 400 training step을 차용하였다.

4.1.3 실험 데이터의 설정

상술한 Fine-Tuning 과정을 거쳐, 새로운 이미지를 생성하기 위해 신경망 학습과정, 즉 LAION 데이터셋에 존재하지 않는 새로운 일러스트를 학습시키고자 하였다. 이를 위해 두 명의 다른 일러스트레이터의 그림들을 연구 동의를 받아 확보하였고, 이를 이용해 Fine-Tuning을 진행하였는데, 각 일러스트레이터의 그림별로 10장 내외의 사진을 각각의 일러스트레이터 별 키워드를 생성해 학습시켰다. Fine-Tuning 논문을 따라 구현된 학습이 완료된 후, 반환되는 .ckpt 파일을 이용해 Stable Diffusion 환경에 대입한 후, 생성 이미지가 적절히 반환되는지 확인하고자 하였다. 데이터 수집 과정에서 더욱 정확한 Fine-Tuning 모델 생성을 위해 몇가지 과정을 거쳤다. 첫번째로, 일러스트레이터가 그린 그림이 같은 캐릭터에 대한 것인지는 고려하지 않는다. 같은 캐릭터에 대해 그린 그림이 아니어도 학습 데이터에 포함토록 하였다. 두 번째, 과도한 그림 장식등으로 인해 캐릭터의 본 모습을 모델이 인식하는데 방해가 생길 우려가 있는 경우 모델 학습에서 제외하였다. 세번째로, 일러스트레이터가 그린 그림이 본인의 독창적인 그림이 아닌 타 캐릭터를 모사한 그림인 경우, 학습 데이터에서 제외하였다. 위 사유로 학습에서 제외된 그림들의 예시는 다음과 같다.

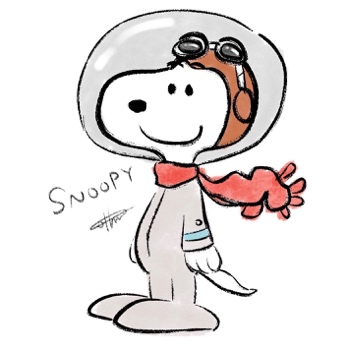


Figure 10 학습 데이터에서 제외된 사진의 예시

[Figure 10]의 왼쪽과 같은 사진의 경우, 일러스트 상 눈 아래에 그려진 꾸밈 효과로 인해 Fine-Tuning 과정에서 모델에 악영향이 가해졌으며, 오른쪽의 사진의 경우 캐릭터의 그림을 모사한 경우로, 일러스트레이터의 그림체를 학습하는데 악영향이 가해질 것을 우려해 제외하였다.

이 외의 경우들은 연구 진행자의 과도한 주관이 가미될 우려가 있어 그림들을 배제시키지 않고 학습하였다. 각 일러스트레이터 별로 학습 과정에서 사용된 두 일러스트레이터의 그림 예시는 다음과 같다.

|  |  |
| --- | --- |
|  |  |

Figure 11 첫번째 일러스트레이터의 그림 예시

|  |  |
| --- | --- |
|  |  |

Figure 12 두번째 일러스트레이터의 그림 예시

4.1.4 실험 과정

4.1.3 에서 설명한 Fine-Tuning 과정을 이용해 새로운 이미지가 포함된 이미지 생성 모델을 확보할 수 있게 되었다. 이에, 본 연구에서는 다음과 같은 과정으로 실험을 진행하고, 연구 가설을 검증하고자 하였다. 가장 먼저, 원본 이미지 모음을 이용해 모델 학습 과정을 거친다. 이 과정에서 [A] 라는 이미지 키워드를 부여하고, 그 키워드를 이용해 이미지를 생성할 수 있도록 한다. 모델의 Fine-Tuning이 완료되면 키워드를 이용해 학습된 이미지의 그림체나 캐릭터의 형태를 생성모델이 잘 익혔는지 확인한다. 그 후, 앞서 세운 가설들과 구현한 프로그램을 이용해 이미지를 다양한 parameter를 부여하며 변형시키고, 해당 이미지들도 동일하게 Fine-Tuning을 진행한다. 이 때 생성된 모델을 이용해 Stable Diffusion 모델의 이미지 생성을 요구하고, 학습 데이터와의 유사도를 판단한다. 위와 같은 과정을 두번째 일러스트레이터에 대해서 반복한다.

이 과정을 통해 보다 객관적으로 이미지 변형에 따른 생성모델의 학습률 저하를 검증할 수 있을 것이라 판단하였다.

4.1.5 출력값 검증

4.1.4에서 설명한 실험 과정을 거쳐 생성된 이미지의 품질을 평가하기 위해 상술한 G-FID 수치를 도입하였다. 앞서 언급했듯 G-FID 수치는 생성된 이미지에 대해 그 품질을 평가하는 수치로서 활용된다. 본 연구에서 또한 G-FID 수치를 이용해 정량적 품질 평가를 진행하고, 동시에 정성적 평가를 통해 이미지 생성 모델의 성능을 평가하고자 하였다.

또한, G-FID 수치 외에도 Inception Score를 도입하였다. Inception Score는 엔트로피, 즉 무작위성을 이용해 생성된 이미지의 품질을 측정한다. 예를 들어, 두가지 분포가 주어졌을 때, 한 분포가 다른 분포에 비해 균일한 값을 가지고 있다면 예측 가능성이 감소, 즉 더 높은 엔트로피를 가지고 있다고 할 수 있다. 따라서, Inception Score는 임의의 이미지가 주어졌을 때, 분류가 잘 되는지에 따라 평가를 하는 척도로, 생성된 이미지가 다양할 수록 균일한 분포가 반환된다. [34-35]

Inception Score는 위의 식으로 계산 가능하며, 본 연구에서는

생성된 이미지의 품질 평가를 위한 척도로써 활용하였다.

4.1.6 이미지 변형 시간 측정

또한, 이미지 변형 처리를 하는 과정에서 걸리는 시간에 대해서 검증이 필요할 것이라 판단하였다. 만약, 변형에 걸리는 시간이 과도하게 오래 걸리는 경우, 일반적으로 이미지를 배포하는 과정에서 사용자가 사용하기 어려울 수 있다고 판단, 적절한 시간 내에 이미지의 변형이 완료 되는지 확인하고자 하였다.

4.2 새로운 이미지 확장자를 이용한 이미지의 배포

4.2.1 이미지 형식 변환 시간

상술한 방식으로 이미지의 이미지 확장자를 설계하였다. 이는 일반적인 경우에서 Google Image Search 등의 이미지 검색 엔진에서 검색이 불가능할 뿐 아니라, 대량의 크롤링 과정에서도 렌더링 되지 않는 클라이언트 사이드 렌더링 이미지 파일로서 웹 기술에도 활용될 수 있다. 이 때문에, 이미지를 .jhp 파일로 변환하고 .jhp 파일을 본 연구진이 개발한 뷰어를 통해 확인하는 과정에서 소요시간이 과도하게 길 경우 불편함을 겪을 수 있다고 판단하였기에 이미지를 .jhp 파일로 변환하는데 소요되는 시간을 확인하고자 하였다. 다양한 크기를 가지는 이미지 파일을 이용해, 이를 .jhp 파일로 변환하는 시간과 .jhp 파일을 뷰어에서 렌더링해 열람할 수 있게하는 과정에서 소요되는 시간을 계산하였다.

4.2.2 설계한 이미지 확장자의 압축 용량 측정

상술했듯, 새로운 이미지 파일이 무손실 압축법이라는 점을 고려하여도 과도하게 이미지 용량이 커질 경우 이미지를 배포하고 저장하는데 어려움이 발생할 수 있다. 이러한 위험성에 대해 평가하기 위해, 동일한 비손실 압축 파일 포맷인 png, bmp와 직접 설계한 이미지 형식이 동일한 이미지에 대해 용량 차이가 어느정도 발생하는지 확인하고자 하였다. 다양한 png, bmp 형식의 이미지를 이용해 이를 .jhp 파일로 변환하고, 기존 상태의 이미지와 .jhp 파일의 용량을 비교하였다.



Figure 15 ‘[A] reading a book' 프롬프트로 생성한 이미지

5. 실험 결과

5.1 이미지 변형을 통한 학습 방지법 실험

5.1.1 원본 이미지 데이터를 통한 Fine-Tuning

4.1.2, 4.1.3의 과정을 통해 생성한 Fine-Tuning된 Stable Diffusion 이미지 생성 모델을 이용해 다양한 프롬프트를 제시하고, 이에 따른 출력 이미지 데이터를 확인하였다. 그 결과는 다음과 같다. 실험 결과는 [Figure 11], [Figure 12]의 첫번째, 두번째 일러스트레이터의 순 으로 제시한다.



Figure 16 '[A] holding a pen' 프롬프트로 생성한 이미지

먼저 다음 3장의 일러스트는 별도의 프롬프트 없이, 키워드를 설정하고, 생성을 거친 결과물이다. Fine-Tuning 과정에서는 상술했듯, 이미지 데이터셋에 키워드를 연결해, 이를 추후 Stable Diffusion에서 입력해 사용할 수 있도록 한다. 본 문단에서는 편의를 위해 첫번째 일러스트레이터의 이미지를 키워드 [A], 두번째 일러스트레이터의 이미지의 키워드를 [B] 라 칭한다. 생성된 이미지는 다음과 같다.



Figure 14 [A] 프롬프트를 입력했을 때 생성된 이미지



Figure 13 [A] 프롬프트를 입력했을 때 생성된 이미지

[A] 키워드에 대해 별도 프롬프트 없이 객체만을 매개변수로 생성된 그림들을 정성적으로 분석했을 경우 확인할 수 있듯, 이미지의 형태나 색감면에서 어색함이 없고, 원 일러스트와 눈의 형태, 얼굴형의 형태, 사용한 디지털 페인트 도구의 질감, 색감, 그리고 일러스트레이터의 고유 사인 까지 모두 모사하는데 성공한 모습을 확인할 수 있다. 다음으로는, [A] 키워드에 더해 추가적인 매개변수를 포함해 프롬프트를 제공하고, 이미지를 생성한 모습이다.

위의 두 이미지는 [A] 프롬프트에 더해 상황을 특정할 수 있는 키워드를 제시하고, 이에 따라 생성된 이미지들이다. 위 이미지들 역시 동일하게 일러스트레이터의 특성을 모사하는데 성공하였고, 이미지에서 어색하게 생성된 부분이 존재하지 않고, 학습 데이터의 그림체와 매우 유사하게 이미지를 생성하였다. 이에 더해, 학습 데이터에는 존재하지 않는 책과 펜 과 같은 물체를 일러스트레이터의 그림체에 맞춰 생성하는데에 성공하였다.

두번째 일러스트레이터의 그림으로 Fine-Tuning된 모델의 이미지 생성 결과는 다음과 같다.



Figure 17 [B] 프롬프트를 입력했을 때 생성된 이미지

[B] 키워드에 대해서도 동일하게 프롬프트를 제시하고 이미지를 생성한 결과이다. 볼 수 있듯, 학습 데이터 상에 존재하지 않는 물체에 대해서도 표현이 어색하지 않고, 원본 일러스트의 그림체를 잘 모사하여 어색함이 없으며, 색감과 특징적인 배경 또한 비슷한 그림체로서 나타난다. 해당 실험들을 통해 Stable Diffusion 모델이 일러스트를 단순히 Fine-Tuning 하는 과정만 거치고도 일러스트레이터의 그림체를와 동일한 새로운 그림을 생성할 수 있음을 보였다.



Figure 18 [B]에 대해 reading a book, holding a pen 프롬프트를 입력하고 생성된 이미지

5.1.2 변형된 이미지를 이용한 Fine-Tuning

5.1.1을 통해 본 연구에서 초기에 제시한 조건이었던 일러스트의 그림체 학습 가능성을 입증하였다. 별도의 정보 없이 단순히 10장 내외의 일러스트들을 학습한 결과, 일러스트레이터의 그림체는 물론 그림의 질감과 사인 까지도 학습하였음을 보였다. 이제, 상술하였던 가설인 이미지 변형을 통한 학습률 저하를 실험하고자 하였다.

실험에 사용된 코드는 Code 1에서 제시한 그것이다. 이 과정에서, parameter는 kernelSize와 nearbyBlurSize의 값으로 이루어져 있는데, 3.1.2에서 제시한 표기법을 기준으로, 각각 (5, 3)의 변형도, (3, 3)의 변형도로 설정한 후 이미지를 변형하였다.

먼저, 4.1.3에서 학습 이미지의 예시로 제시한 이미지에 변형을 가했을 때, 다음과 같다.

|  |  |  |
| --- | --- | --- |
| Picture (a) | Picture (b) | Picture (c) |

Figure 19 변형도에 따른 이미지의 변화

위의 세 그림들 에서, Picture (a)는 원본 이미지, Picture (b)는 (5, 3)의 parameter로 변형한 이미지, Picture (c)는 (3, 3)의 parameter로 변형한 이미지이다. 5.1.1의 학습 과정에서 사용된 이미지들을 각각 (5, 3), (3, 3)의 변형도로 모두 변형하고, 이를 새로운 Fine-Tuning 과정에서 각각 분리해 사용하였다. 이를 통해 총 4가지 Fine-Tuning 모델이 생성되었고, 이를 이용해 5.1.1에서 진행한 방식으로 이미지를 생성하였다. 먼저, A에 대해 변형하고 생성한 Fine-Tuning 모델의 이미지 생성 결과는 다음과 같다.

|  |  |
| --- | --- |
| Picture a | Picture b |
| Picture c | Picture d |
| Picture e | Picture f |

Figure 20 변형된 A로 Fine-Tuning해 생성된 이미지

위 6장의 이미지중, Picture a~c는 (5, 3)의 변형도, d~f는 (3, 3)의 변형도를 가해 변형한 사진으로 학습된 모델의 결과물이다. Picture a와 d는 별도의 프롬프트 없이 [A], b와 e는 ‘[A] holding a book’, c와 f는 ‘[A] holding a pen’ 을 프롬프트로 생성한 이미지이다. 위 6개의 이미지를 정성적으로 분석했을 때, 이미지의 변형을 통한 학습률의 저하가 성공적으로 이루어졌다고 입증 가능하다. 먼저, Picture a, b, e의 경우, 블러 처리했던 이미지내의 요소들을 학습하였기 때문에 생성된 이미지 역시 눈, 턱, 머리카락 등이 매우 어색하고 일그러진 상태로 생성되었다. 더욱이, Picture e의 경우 머리카락 요소를 book 요소로 모델 상에서 오판이 이루어졌다고 추측되며, 그 결과 book 요소 대신 머리카락이 들려 있는 모습을 확인할 수 있다. Picture c의 경우는 생성과정에 더욱 오류가 컸던것으로 확인할 수 있다. [A]라고 볼 수 없는 형태의 얼굴이 보이지만, 그 마저도 정상적인 배치가 아니며, 얼굴 내에 눈, 코, 입이 명확히 확인되지 않는다. 프롬프트에서 주어졌던 pen이 확인되긴 하지만, 정상적인 몸체와 얼굴이라고 판단할 수 없다. Picture d와 f에서는 (3, 3)이라는 더욱 큰 변형도로 인해 원본 이미지의 그림체를 전혀 확인할 수 없고, 극사실적인 사진들이 생성되어, 정상적인 Fine-Tuning 과정이 진행되었다고 판단하기 어려웠다. 변형된 이미지를 통해 Fine-Tuning한 모델에서 생성된 이미지들은 일부만 저조한 품질을 보인것이 아닌, 모두 정상적인 이미지 생성을 하지 못하였다. 이러한 기전은 [B] 일러스트들에 대해서도 동일하게 보여졌는데, [B]에 대해 동일한 과정으로 Fine-Tuning을 거치고, 생성한 이미지들은 다음과 같다. 단, 분량의 제약상 이미지 생성의 결과가 [A]의 경우와 매우 유사하기에 예시 이미지 세 장만을 논문에 탑재하였다.



Figure 21 변형된 B로 Fine-Tuning해 생성된 이미지

위 세 장의 이미지에서 확인할 수 있듯, 4.1.3에서 제시한 B 일러스트의 그림체는 소실되었으며, 오른쪽의 이미지의 경우, 생성된 이미지들 중 가장 원본 이미지와 비슷하게 생성되었으나, 안면부의 작화가 매우 소실되었고, 눈, 코, 입 등의 연결이 매끄럽지 않고 뭉개진 부분이 다수 발견되었다.

이러한 실험 과정들을 통해서 논문 초반부에서 제시한 이미지 변형을 통한 학습효율 저하 가설을 정성적으로 검증하였다.

이에, 이미지를 변형시켜 온라인상에 배포하였을 경우에 사용자들이 해당 이미지를 인식하는데 어려움이 없는지 검증하고자 하였다.

5.1.3 생성된 이미지의 정량적 품질

5.1.2에서는 변형된 이미지 데이터셋으로 Fine-Tuning된 모델이 생성한 이미지의 품질에 대해 정성적으로 분석하였다. 이 과정에서 충분히 이미지의 생성 품질이 저하된다는 것을 입증 하였지만 보다 객관적인 수치로서 이를 증명하고자 한다. 이를 위해 상술했던 G-FID 수치와 Inception Score를 활용하였다. 실험은 먼저 A와 B를 (5, 3) 변형도로 변형한 이미지 데이터 셋을 이용해 Fine-Tuning을 거치고, 생성된 모델을 이용해 Stable Diffusion 모델 상에서 프롬프트를 제공하지 않고 50장의 이미지를 생성하였다. 이후, 해당 50장의 이미지의 Inception Score와 G-FID 수치를 계산하여 평균치와 표준편차를 계산하며 이상치의 여부를 확인하였다. 단, 상기했듯 이미지의 품질 자체는 우수하지만, 학습된 그림체에서 완전히 벗어나는, 즉 5.1.2 Table 2의 Picture d와 같은 이미지는 가중치로 간주하고 제외한 후 계산하였다. Inception Score와 G-FID 수치를 계산하기 위한 메소드의 코드는 다음과 같다.

Code 4 G-FID를 계산하기 위한 함수

|  |
| --- |
| (…)  def calculate\_inception\_score(  sample\_dataloader,  test\_dataloader,  device="cpu",  num\_images=50000,  splits=10,  ):  inception\_model = InceptionScore(device=device)  inception\_model.eval()  (…)  for k in range(splits):  part = preds[  k \* (num\_images // splits) : (k + 1) \* (num\_images // splits), :  ]  py = np.mean(part, axis=0)  scores = []  for i in range(part.shape[0]):  pyx = part[i, :]  scores.append(entropy(pyx, py))  split\_scores.append(np.exp(np.mean(scores)))  return {"Inception Score": np.mean(split\_scores)} |

|  |
| --- |
| (…)  def calculate\_gfid(mu1, sigma1, mu2, sigma2, eps=2e-6):  (…)  covmean, \_ = linalg.sqrtm(sigma1.dot(sigma2), disp=False)  if not np.isfinite(covmean).all():  msg = ('fid calculation produces singular product; '  'adding %s to diagonal of cov estimates') % eps  print(msg)  offset = np.eye(sigma1.shape[0]) \* eps  covmean = linalg.sqrtm((sigma1 + offset).dot(sigma2 + offset))  if np.iscomplexobj(covmean):  if not np.allclose(np.diagonal(covmean).imag, 0, atol=1e-3):  m = np.max(np.abs(covmean.imag))  (…)  covmean = covmean.real  tr\_covmean = np.trace(covmean)  return (diff.dot(diff) + np.trace(sigma1)  + np.trace(sigma2) - 2 \* tr\_covmean) |

Code 3 Inception Score를 계산하기 위한 함수

위의 Python 코드 두 종류를 이용해 계산한 Inception Score와 G-FID의 A, B 각각에 대한 통계 값은 다음과 같다. 유효숫자는 Inception Score의 경우 소수점 아래 5자리, G-FID의 경우 소수점 아래 첫째자리로 설정하였다.

|  |  |  |
| --- | --- | --- |
|  | Inception Score | G-FID |
| 평균 | 0.00926 | 275.8 |
| 표준편차 | 0.00063 | 44.0 |

Table 2 A 이미지셋에 대한 품질평가척도

|  |  |  |
| --- | --- | --- |
|  | Inception Score | G-FID |
| 평균 | 0.01526 | 249.2 |
| 표준편차 | 0.00053 | 28.4 |

Table 3 B 이미지셋에 대한 품질평가척도

실험 결과, 앞서 정성적으로 평가한 결과와 동일하게 매우 저조한 수치를 기록했다. FID와 Inception Score를 제시하고 설계한 연구를 참조하였을 때, 주어진 본 수치는 생성모델에서 매우 높은 disturbance를 기록한 것으로, 이미지 생성 모델로서의 가치를 상실하였다고 판단할 수 있었다. 이러한 과정을 통해 이미지 변형을 통한 학습률 저하의 효과에 대해 정량적으로 입증할 수 있었다.

5.1.4 변형된 이미지의 사용자 인식성

상술한 실험을 통해 이미지를 변형한 경우에 Stable Diffusion 모델과 Fine-Tuning 과정에서 부정확도가 증가하고, 사용자가 의도한 이미지 생성이 불가능해짐을 확인하였다. 이에, 일러스트레이터, 만화가 등이 본인의 저작물을 배포하고자 할 때, Text To Image AI에 무단 학습되는것을 방지하고자 본 연구에서 제시한 변형법을 사용할 수 있게 되었다. 하지만, 이를 위해서는 변형된 그림이 생성 모델에는 최대한의 부정확함을 제공하되, 일반 사용자가 그림을 인식하는 데에는 큰 지장이 없도록 하여야 한다. 따라서 본 5.1.3 장에서는 변형된 이미지가 원본 이미지에 비해 어느정도의 품질 하락을 겪는지, 또한 사용자가 이미지를 인식하는데 어려움이 없는지 확인하고자 하였다.

먼저, 실험 설정을 위해 변형된 이미지의 품질을 평가할 수 있는 척도로서 상술한 PSNR 수치와 SSIM 수치를 도입하였다.

첫번째로, A에서 (5, 3)의 변형도로 변형된 11장의 이미지 데이터셋을 이용해 원본사진과의 PSNR, SSIM 수치를 계산했다. 계산한 결과는 다음과 같다. 유효숫자는 소수점 아래 셋째 자리까지로 설정하였고, 실험 결과 또는 , 즉 외부의 이상치는 발견되지 않았다.

|  |  |  |
| --- | --- | --- |
|  | 평균 | 표준편차 |
| PSNR (dB) | 27.867 | 8.27 |
| SSIM | 0.923 | 0.051 |

Table 4 (5, 3)의 변형도로 변형된 A 이미지셋의 품질

위와 같은 수치는 평균 25 이상의 PSNR 수치, 0.9 이상의 SSIM 수치임을 고려했을 때, 이미지 변형이 극심하지 않으며, 사용자가 이미지를 인식하는 데에도 큰 문제가 발생하지 않을 것이라 유추할 수 있다. B에 대해서 또한 (5, 3)의 변형도로 변형한 13장의 이미지에 대해서도 품질 수치를 계산하여 통계적으로 정리하였다. B에서 또한 이상치는 발견되지 않았다.

|  |  |  |
| --- | --- | --- |
|  | 평균 | 표준편차 |
| PSNR (dB) | 26.981 | 7.44 |
| SSIM | 0.927 | 0.054 |

Table 5 (5, 3)의 변형도로 변형된 B 이미지셋의 품질

결과적으로, A와 B에 대해 진행한 변형된 이미지와 원본 이미지 간의 PSNR, SSIM 수치 계산을 통해 이미지 변형이 이미지 생성 모델의 성능 저하에는 크게 유효하지만, 선행 연구들 에서 제시했던 준수한 이미지 품질 범위 내에 있기에, 사용자가 인식하는데 중대한 지장을 끼치지 않을 것이라 판단했다. 이로써 이미지 변형을 통한 이미지 생성 모델의 학습효율 저하가 유효하며, 일반적인 사용자들의 이미지 열람에도 큰 지장을 주지 않음을 확인할 수 있었다. 또한, 앞서 설명한 선행연구들에 비추어 보았을 때, 인간이 이미지와 사물을 인식하는 메커니즘을 참고하여 인간이 이미지를 인식하는데 그 방해가 최소화되도록 하였기에, 이미지의 무단학습은 막되, 일반적인 사용자가 이미지를 확인할 때에는 불편함이 최소화되도록 하였다.

5.1.5 이미지 변형 과정의 소요시간

또한, 이미지 변형에 소요되는 시간을 측정하고자 하였다. 이미지 변형에 과도하게 시간이 소요될 경우, 사용자가 본 연구에서 제시하는 변형법을 활용해 이미지 무단학습을 차단하는 과정에서 어려움을 느낄 것이라 판단, 본 실험을 진행하였다.

앞서 A와 B 이미지 데이터셋에 포함된 총 24장의 이미지에 변형 효과를 적용하는데 걸리는 시간을 측정하였다. 계산 결과, 평균 약 1.8초의 시간이 걸리는 것을 확인하였다. 하지만, 실험 결과에서 2.7 시그마 범위를 +방향으로 벗어나는 값이 발견 되었다. 해당 값은 변환 과정에서 Canny Edge Detection을 진행할 때, 다른 이미지들보다 많은 테두리가 검출되었고, 이로 인해 blur 효과를 적용해야하는 픽셀 수가 늘어나며 발생한 값으로 판단하였다. 결과적으로, 1.8초의 변환 시간은 사용자가 불편함을 느낄 시간이 아니라고 판단하였으며, 변환을 진행하고자 하는 사용자가 편하게 변환과정을 진행할 수 있을 것이라 사료된다.

5.2 이미지 확장자 설계를 통한 무분별 크롤링의 방지

5.2.1 이미지 포맷 변환 소요 시간 측정

상술하였듯, 본 연구에서 새롭게 .jhp 이미지 형식을 제안하였다. .jhp 파일은 그 설계 자체로 우수성을 갖지 않지만, 압축 과정이나 데이터의 설계가 공개되어 뷰어를 제작하는 것이 가능한 다른 이미지 포맷들과 달리, 본 연구진이 제공하는 뷰어로만 이미지의 열람이 가능하도록 설계되었다. 하지만, 변환하는 과정에서 시간 소요가 너무 크거나, 변환 후의 용량이 대중적인 비손실 압축 이미지 포맷인 png, bmp등과 비교했을 때 과도하게 클 경우, 실질적인 사용이 어렵다고 판단하였다. 이에 이미지 포맷을 .jhp로 변환하는데 소요되는 시간과 변환후의 용량을 계산, 실사용 가능할지에 대해 평가하고자 하였다. 변환 과정에서, 이미지 간의 차이로 인해 발생할 수 있는 소요 시간의 차이를 배제하기 위해, 변환 과정상 이미지 데이터를 평문으로 변환하는데 걸리는 시간을 측정하였다. 이후, 가장 극단적인 상황을 예시로 4K 사진에 대해 변환 과정에서 데이터 평문화부터 데이터 압축까지 걸리는 시간을 동일한 컴퓨팅 환경에서 측정하였다.

먼저, 이미지 포맷을 변환하는데 걸리는 시간을 측정하였다. 실험은 화소를 가지는 이미지에서 을 100부터 1000까지일 때에 대해 이미지를 평문 데이터로 변환하는데 걸리는 시간을 측정하였다. 이 과정은 python 언어 상에서 time.perf\_counter() 함수를 이용해 계산하였으며 그 결과는 다음 그래프로 나타낼 수 있었다.

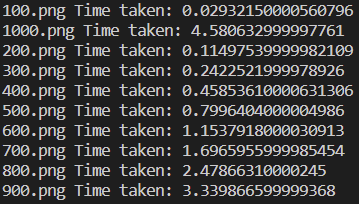


Figure 22 실험 결과

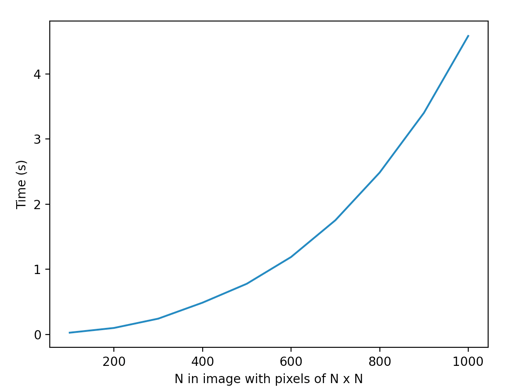


Figure 23 실험 결과를 나타낸 그래프

예상과 동일하게 한 변의 픽셀 개수에 대해 2차원 그래프 형태로서 증가하는 모습을 띄고 있으며, 이미지의 경우 이미지 데이터를 평문으로 저장하는데 4초가 소요되었다.

이후, 4K 사진을 압축된 형태로서 저장하는데 걸리는 시간을 계산하였다. 그 결과 총 소요시간은 36초로, 다소 느린 실행 시간을 보였지만, 실험에 사용된 이미지가 초 고화질 이미지였다는 점, 그리고 이미지 포맷 변환 과정을 위해 짜여진 코드가 실행속도가 다소 느린 Python 기반이었던 점을 고려했을 때, 효율성 면에서 성능 개선의 여지가 충분하다고 판단하였다.

5.2.2 변환된 이미지 포맷의 용량 측정

본 연구에서 제시한 이미지 포맷이 실질적으로 사용되기 위해서 고려해야할 요소중에는 변환 시간 외에도 변환 후의 용량이 있었다. 이를 고려하기 위해, 10개의 서로 다른 png 이미지와 10개의 서로 다른 bmp 이미지를 이용해 변환시의 용량을 비교하였다. 이 과정에서 투명한 픽셀의 값은 (256, 256, 256)으로 설계하였다. 변환된 이미지와 원본 이미지 간의 용량 차는 다음과 같다.

|  |  |  |
| --- | --- | --- |
|  | PNG | BMP |
| 용량 비율 (%) | 353 | 210 |

Figure 6 jhp 이미지 포맷과 png, bmp 형식의 용량비교

본 실험 결과에서 유추 할 수 있듯, .jhp 이미지 포맷의 압축 성능이 일반적인 이미지 포맷들과 비교하였을때 다소 떨어지는 점을 확인할 수 있다. 하지만, 본 연구에서 제시한 이미지 포맷의 설계 목적이 웹 크롤링 방지이며, 연구 기간과 연구 인력이 타 이미지 포맷들에 비해 매우 적음을 고려하였을 때, 추후 발전 가능성이 크다고 판단하였다.

6. 결론

본 연구에서는 Stable Diffusion으로 대표되는 Text To Image AI의 학습 과정에서 무분별한 데이터 수집이 발생함을 확인하였고, 이 과정에서 동의되지 않은 저작물들이 학습에 이용되는 문제를 인식하였다. 이는 법적으로도 모호한 소지가 있어, 저작권자들이 보호를 받기가 힘들기에, 저작권자들이 본인의 저작물이 제 3자의 인공지능에 의해 학습되는것을 방지하기 위해 사용할 수 있는 해결책을 제시하고자 하였다. 이를 위해 일반적인 사용자가 이미지를 열람하는데 방해받지 않는 선에서 이미지를 다소 변형해, 인공지능이 해당 이미지를 학습하는 과정에서 학습이 정상적으로 이뤄지지 않도록 하는 설계를 가설로서 제시하였다. 또한, 이 외에도 온전한 이미지 배포를 위해, 전용 뷰어로만 이미지를 열람할 수 있도록하는 .jhp 파일의 설계를 제시하였다. 이에 더해, 추후 웹 프론트엔드 라이브러리의 발전 가능성을 제시하고, 그 과정에서 AES 알고리즘을 활용해 보안성이 강화된 통신을 해, 그 사이에서 이미지 파일을 주고 받는 프로토콜에 대해서도 제시하였다. 전자의 가설들을 검증하기 위해 이미지를 변형하는 프로그램을 구현하였고, 이를 직접 DreamBooth Fine-Tuning을 활용해 변형된 이미지를 모델에 학습시켜 이미지를 생성해보았다. 그 결과, 원본 일러스트를 학습하였던 생성 모델은 일러스트의 화풍, 질감, 그림체를 모두 학습해 유사한 이미지를 생성했지만, 일러스트를 다소 변형해 학습시킨 경우, 학습률이 급감해 매우 낮은 품질의 이미지를 생성함을 확인했다. 이러한 과정을 정성적 평가 뿐 아니라 G-FID 수치와 Inception Score를 도입해 확인하였다. 또한, 일반적인 사용자가 변형된 이미지를 확인하는데 있어 불편함이 있는지 SSIM과 PSNR 수치를 이용해 확인하였다. 이 외에도 .jhp 파일의 구현을 완성하였고, 압축 알고리즘을 이용해 상용되는 이미지 포맷에 견줄 수 있을 정도의 압축률을 보이는 이미지 포맷을 구현하였고, 이미지 포맷 변환 소요시간, 타 이미지 포맷과의 용량 비교를 통해 본 연구에서 제시하는 이미지 포맷이 실제로 활용될 여지가 충분함을 보였다. 추후에 웹 프론트엔드 라이브러리로의 발전을 꾀해, 이미지 검색 엔진이나 크롤링 엔진에서 감지되지 않도록 .jhp 파일이 클라이언트 사이드에서 렌더링 되도록 설계한다면 검색 엔진의 이미지 SEO를 저하시켜 보다 좋은 효과를 볼 수 있을 것이라 판단하였다. 따라서, 본 연구에서 제시한 방법론들이 실제로 저작권자들이 자신들의 저작물이 무단 학습되는 것을 막고, 권리를 보장받을 수 있게 기여할 것이라 기대한다.

References

[1] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[2] Stability AI. Official documentation. Cited Nov 14, 2022. Available from: <https://stability.ai/blog/stable-diffusion-public-release>. (accessed Nov 14, 2022).

[3] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks." Communications of the ACM 63, no. 11 (2020): 139-144.

[4] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

[5] Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).

[6] Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis." *arXiv preprint arXiv:1809.11096* (2018).

[7] Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

[8] Ramesh, Aditya, et al. "Zero-shot text-to-image generation." *International Conference on Machine Learning*. PMLR, 2021.

[9] Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684-10695. 2022.

[10] Luisier, Florian, Thierry Blu, and Michael Unser. "Image denoising in mixed Poisson–Gaussian noise." IEEE Transactions on image processing 20.3 (2010): 696-708.

[11] https://laion.ai/blog/laion-5b/ LAION AI. Official blog post. Cited Nov 28, 2022. Available from: <https://laion.ai/blog/laion-5b> (accessed Nov 28, 2022).

[12] Ruiz, Nataniel, et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation." arXiv preprint arXiv:2208.12242 (2022).

[13] Lawrence W. Stark, Claudio M. Privitera, Huiyang Yang, Michela Azzariti, Yeuk Fai Ho, Theodore T. Blackmon, and Dimitri A. Chernyak "Representation of human vision in the brain: how does human perception recognize images?," Journal of Electronic Imaging 10(1), (1 January 2001). <https://doi.org/10.1117/1.1329895>

[14] Kanwisher, Nancy. "Functional imaging of human visual recognition Nancy Kanwisher"", Marvin M. Chun", Josh McDermott", Patrick J. Ledden." *Cognitive Brain Research* 5 (1996): 55-67.

[15] Haxby, James V., Leslie G. Ungerleider, Barry Horwitz, Jose Ma Maisog, Stanley I. Rapoport, and Cheryl L. Grady. "Face encoding and recognition in the human brain." *Proceedings of the National Academy of Sciences* 93, no. 2 (1996): 922-927.

[16] Wardle, Susan G., and Chris I. Baker. "Recent advances in understanding object recognition in the human brain: deep neural networks, temporal dynamics, and context." *F1000Research* 9 (2020).

[17] Raichle, Marcus E. "A brief history of human brain mapping." *Trends in neurosciences* 32, no. 2 (2009): 118-126.

[18] Ding, Lijun, and Ardeshir Goshtasby. "On the Canny edge detector." *Pattern recognition* 34, no. 3 (2001): 721-725.

[19] Xu, Zhao, Xu Baojie, and Wu Guoxin. "Canny edge detection based on Open CV." In *2017 13th IEEE international conference on electronic measurement & instruments (ICEMI)*, pp. 53-56. IEEE, 2017.

[20] Huynh-Thu, Quan, and Mohammed Ghanbari. "Scope of validity of PSNR in image/video quality assessment." *Electronics letters* 44, no. 13 (2008): 800-801.

[21] Hore, Alain, and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." In *2010 20th international conference on pattern recognition*, pp. 2366-2369. IEEE, 2010.

[22] Sara, Umme, Morium Akter, and Mohammad Shorif Uddin. "Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study." *Journal of Computer and Communications* 7, no. 3 (2019): 8-18.

[23] Channappayya, Sumohana S., Alan Conrad Bovik, and Robert W. Heath. "Rate bounds on SSIM index of quantized images." *IEEE Transactions on Image Processing* 17, no. 9 (2008): 1624-1639.

[24] Han, Yu, Yunze Cai, Yin Cao, and Xiaoming Xu. "A new image fusion performance metric based on visual information fidelity." *Information fusion* 14, no. 2 (2013): 127-135.

[25] Naeem, Muhammad Ferjad, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. "Reliable fidelity and diversity metrics for generative models." In *International Conference on Machine Learning*, pp. 7176-7185. PMLR, 2020.

[26] Moffat, Alistair. "Huffman coding." *ACM Computing Surveys (CSUR)* 52, no. 4 (2019): 1-35.

[27] Knuth, Donald E. "Dynamic huffman coding." Journal of algorithms 6, no. 2 (1985): 163-180.

[28] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).

[29] Kreft, Sebastian, and Gonzalo Navarro. "LZ77-like compression with fast random access." In 2010 Data Compression Conference, pp. 239-248. IEEE, 2010.

[30] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." (1999).

[31] Bogdanov, Andrey, Dmitry Khovratovich, and Christian Rechberger. "Biclique cryptanalysis of the full AES." In *International conference on the theory and application of cryptology and information security*, pp. 344-371. Springer, Berlin, Heidelberg, 2011.

[32] Sharma, Dushyant, Rishabh Shukla, Anil Kumar Giri, and Sumit Kumar. "A brief review on search engine optimization." In *2019 9th international conference on cloud computing, data science & engineering (confluence)*, pp. 687-692. IEEE, 2019.

[33] Gudivada, Venkat N., Dhana Rao, and Jordan Paris. "Understanding search-engine optimization." Computer 48, no. 10 (2015): 43-52.

[34] Barratt, Shane, and Rishi Sharma. "A note on the inception score." *arXiv preprint arXiv:1801.01973* (2018).

[35] Obukhov, Artem, and Mikhail Krasnyanskiy. "Quality assessment method for GAN based on modified metrics inception score and Fréchet inception distance." In *Proceedings of the Computational Methods in Systems and Software*, pp. 102-114. Springer, Cham, 2020.

|  |  |
| --- | --- |
| 김 진 호(Jinho Kim) | |
| 사진  25×30 | • 2021년 2월 : 언주중학교 졸업  • 2021년 3월~현재 : 민족사관고등학교 재학 |
| <관심분야>  Computer Vision, Deep Learning, System Software | |
| 한 준 희(Jooney Han) | |
| 사진  25×30 | • 2022년 2월 : 가원중학교 졸업  • 2022년 3월~현재 : 민족사관고등학교 재학 |
| <관심분야>  Computer Graphics, Artificial Intelligence, Encryption  Algorithms | |