

# 발표 스크립트

[팀원소개](#)

[프로젝트 배경 & 개요](#)

[프로젝트 파이프라인](#)

[T2I](#)

[Lora](#)

[dreambooth](#)

[EasyPhoto](#)

[Experiments](#)

[Adetailer](#)

[Experiments](#)

[Prompt](#)

[I2I](#)

[Upscale](#)

## 팀원소개

강동훈, 박사무엘, 지주영 & 김아진, 남궁맑음, 정양섭

## 프로젝트 배경 & 개요

### ▼ 스튜디오 촬영에 수요 증가

스튜디오 사진 촬영 경험이 있는 사람들은 증명사진과 가족사진 대부분 차지하고

10명중 8명이 프로필을 사진을 많이찍고 MZ 겨냥한 스튜디오 촬영시장이 확대될것으로 전문가들은사료합니다.  
(사진 2개 더 보여주고)

하지만 가격이 너무 비싸게 단점으로 다가옵니다.- 가격사진

### ▼ AI 프로필 뉴스

AI 프로필에 대한 관심이 이 처럼 많은 영상들처럼 관심이 지대합니다.

실제로 이미 하고있는 기업

스노우

뤼튼

컨셉 ai (이름 확인 필요) 등..

이미 많은 기업이 시도를 해보고 있는 시장입니다.

그래서 저희는 우리도 만들어 볼 수 있지 않을까?

5주안에 한번 구현을 해보자

저희는 차별성을 2가지로 두었습니다.

1. .포즈를 선택하여서 사진화 하는게 목표로 두었습니다.
2. warmtone cooltone을 이용한 색채의 프로필사진을 뽑아보고자 하였습니다.  
웜톤은 황금, 노랑, 오렌지빛이 강하며,  
쿨톤은 밝고, 푸른빛이 강하며,  
사진 또한 맞는 컬러를 통해 개인의 피부톤에 맞춰서 특별한 경험 선사하고자 하였습니다.

### ▼ input & output

input - 사용자의 사진 10장 = 사용자의 일상적인 사진들을 받아서  
output - 스튜디오에서 찍은것 과 같은 예술품(프로필 사진)이 나오도록 합니다.

---

## 프로젝트 파이프라인

저희 생성 파이프라인에 대해 소개해드리겠습니다.

### ▼ 파이프라인 구조

먼저 사용자의 input image를 받아 easy photo를 거쳐 t2i 모델을 통해 이미지를 지정한 포즈와함께 생성합니다.

그리고 adetailer로 보정을 진행합니다.

다음으로 t2i의 output은 i2i로 넘어가 adetailer의 보정만을 받아 마지막 해상도를 높이는 upscale로 넘어가게 됩니다.

그럼, 마지막으로 upscale에서는 i2i의 output을 베이스로 그리고 resample의 base로 저장해 그 이미지에 맞게 생성을 하고 그 뒤에 해상도를 높여 최종적으로 생성하여 사용자에게 보여줍니다.

각 모델에 자세히 설명하겠습니다.

## T2I

간단히 stable diffusion t2i 추론 과정에 대해 설명하겠습니다.

### ▼ T2I

먼저, Stable Diffusion은 잠재 공간(latent space)에 무작위 텐서(random tensor)를 생성합니다. 이때, 씨드(seed)번호를 사용해 무작위 숫자 생성기(random number generator)를 통해 무작위 텐서를 제어할 수 있습니다. 무작위 텐서가 바로 잠재 공간에 있는 이미지입니다.

다만, 이때에는 모든 이미지가 노이즈일 뿐이죠.

### 2단계

SD의 모델 아키텍처는U-Net의 구조를 가지고 있습니다. 이는 noise-predictor로 작동합니다. 잡음 예측기(U-Net)가 잠재 잡음 이미지와 텍스트 프롬프트를 입력받아 잡음을 예측합니다. 이 과정도 잠재 공간(4x64x64 텐서)에서 시행됩니다.

### 3단계

latent image space 노이즈를 제거합니다. 그 결과는 새로운 잠재 이미지입니다.

### 4단계

최종적으로 VAE 디코더가 latent space를 픽셀 공간으로 보내어 이미지를 생성합니다. 이것이 스테이블 디퓨전을 돌리면 생성되는 이미지 과정입니다.

다음으로는 SD에서 t2i 과정에서 쓰였던 자세한 과정을 설명하겠습니다

SD에는 여러 파인튜닝 방법이 있지만, 저희는 로라 와 드림부스를 이용했습니다.

## Lora

먼저 LoRA에 대해 설명하겠습니다

## ▼ LoRA

최근 chatGPT 열풍에 이어 LLM에 대한 인기가 뜨겁고 연구가 활발합니다.

하지만 LLM을 파인튜닝을 할때 모델의 파라미터가 너무 많다보니 리소스의 제약이 크다는 점입니다.

LoRA 논문은 기존의 파라미터보다 훨씬 적은 파라미터를 가지고 튜닝하는 방법으로 통해서 리소스 제약에서 벗어나고 성능도 비슷하거나 더 높은 수준으로 훈련시킬 수 있도록 하는 수학적 메커니즘을 소개했습니다.

지금 보시는 이미지는 LoRA의 전체적인 이미지입니다. 이 이미지에서 보시는 것과 같이, 로라는 기존의 weights 대신 새로운 파라미터를 이용해서 동일한 성능과 더 적은 파라미터로 튜닝할 수 있는 방법론을 제시하고 있습니다.

1. 논문에서는 기존의 LLM 모델을 하나의 확률 함수로 계산합니다.
2. 그리고 파인튜닝 과정에서 LLM이 튜닝 되는 파이가 최적화 되는 식은 수식 1번처럼 표현 합니다.
  - a. log-likelihood function으로 문제를 해결 할 때 가장 적합한 파라미터 파이의 확률 최대화 하는 것이라고 생각하면 됩니다.
  - b. 직관적으로 backpropagation할 때의 모델을 나타내면,  $\Phi = \Phi_0 + \Delta\Phi$
3. Equation1에 근거하여 만약 accmulated gradient values( $\Delta\Phi$ )를 기존보다 훨씬 적은 파라미터인  $\Theta$ 로 치환하여  $\Delta\Phi(\Theta)$ 로 나타내면 equation2로 바뀌게 됩니다.
  - a. 즉 기존의 log-likelihood 문제에서 모델이 backpropagation 과정에서 이용되는 파라미터 연산문제를 더 적은 파라미터  $\Theta$ 로 치환하여 풀겠다는 의미입니다.

간단하게 설명을 하자면, cross-attention weight는 행렬에 저장됩니다. 행렬은 기본적으로 엑셀 스프레드시트처럼 열과 행으로 나열된 숫자들에 불과합니다. LoRA 모델은 이러한 행렬에 가중치를 추가함으로써 모델을 미세 조정합니다.

LoRA 모델이 동일한 수의 가중치를 저장하면서도 파일 크기가 작은 이유는 뭘까요? LoRA는 행렬을 랭크가 낮은(low-rank) 두 개의 행렬로 분해하기 때문입니다. 보셨던 옆의 그림처럼 이렇게 함으로써 훨씬 더 적은 숫자를 저장할 수 있습니다.

어떤 모델이 1000개의 행과 2000개의 열로 구성된 행렬을 가지고 있다고 가정하겠습니다. 그러면 그 모델 파일에는 2백만 (1000x2000) 개의 숫자가 저장됩니다. LoRA는 이 행렬을 1000x2 행렬과 2x2000 행렬로 쪼갭니다. 이렇게 하면 총 6천개(1000x2 + 2x2000)의 숫자만 필요하고, 따라서 원래의 행렬에 비해 1/333 으로 줄어듭니다.이 때문에 LoRA 파일의 크기가 훨씬 작은 것입니다.

이 예제에서 LoRA에 저장되는 행렬의 랭크(rank)는 2입니다. 원래의 차원의 랭크는 2000 이니 훨씬 작죠. 그래서 저 랭크 행렬(low-rank matrix)라고 합니다.

결과적으로는 - LoRA를 이용하였을 때 해당 fields에서 SOTA를 달성할 뿐만 아니라, GPT-3의 경우 175B의 파라미터 가운데 0.01%의 파라미터 개수만 이용할 정도로 효율성이 좋습니다.

결과 옆의 사진은 간단하게 로라의 학습 폴더 계층 구조를 표현한 것입니다.

## dreambooth

다음으로

### ▼ 드림부스 를 소개하겠습니다.

Dreambooth는 구글 연구진이 발표한 논문에서 사용된 학습 방법의 이름으로 Imagen이라는 text-to-image 생성 모델을 어떤 subject에 대한 몇 장의 사진으로 Fine-tuning 하여 개인화된 text-to-image 생성 모델을 만들고 해당 subject를 새로운 context에서 높은 fidelity로 이미지를 생성할 수 있게 해주는 학습 방법입니다.

Dreambooth는 기존의 Fine-tuning에서 발생하는 다음 두 가지 문제를 해결하는 것을 목표로 제안된 방법입니다.

1. 주제 기반의 이미지 생성. 몇 장의 간단한 사진만으로 주제의 시각적 특징에 대한 높은 충실도를 유지하면서 새로운 맥락의 사진을 합성하는 것.
2. 몇 장의 이미지로 text-to-image diffusion 모델을 fine-tuning 하면서도, 기존 모델의 Semantic knowledge를 보존하는 것.

위 내용을 통해 알 수 있듯이, Dreambooth는 기존의 Fine-tuning과는 달리 적은 수의 이미지만으로 모델의 overfitting, language drift 없이 학습이 가능하기 때문에 개인이 더 손쉽게 fine-tuning이 가능하다는 장점이 있습니다.

보시는 예제 처럼 class name은 [V]의 상위 개념을 써주면 됩니다.

드림부스의 비밀은

Dreambooth는 이런 문제를 class image(Regularization image, 정규화 이미지)를 통하여 해결하는 것입니다.

학습하고 싶은 이미지인 A [V] dog에 대한 input image를 학습함과 동시에 기존 모델을 이용하여 출력한 A dog 라는 class name에 대한 이미지인 class image를 같이 학습하여 기존 모델이 가진 class name에 대한 지식을 잊지 않도록 하는 겁니다.

구체적인 방법으로는, 학습할 때 사용하는 손실함수로 class-specific prior preservation loss 라는 자체적인 손실 함수를 이용했습니다.

저희는 dream-booth를 fine-tuning하고 학습하기에는 학습에 쓰이는 gpu 자원이 부족하고 시간이 부족하다고 판단했고, 이미 잘 만들어진 개인화된 base model들을 찾아 쓰게 좋다고 판단하여 시간을 들여 아래 모델을 찾아 개발을 진행 했습니다.

다음으로

## EasyPhoto

▼ EasyPhoto 를 소개하겠습니다.

Stable Diffusion Web UI(SD-WebUI)는 이 프레임워크의 가장 유명하고 잘 알려진 애플리케이션 중 하나입니다.

SD-WebUI 프레임워크가 제공하는 편리함으로 인해 EasyPhoto 프레임워크 개발자 & 논문 저자는 이를 완전한 애플리케이션이 아닌 웹 플러그인으로 만들기로 결정했습니다.

종종 ID 상실로 어려움을 겪거나 이미지에 비현실적인 특징을 도입하는 기존 방법과 달리 EasyPhoto 모델은 Stable Diffusion 모델의 이미지 간 기능을 활용하여 정확하고 사실적인 이미지를 생성합니다. face swap 을 하는 프레임워크 입니다.

EasyPhoto 모델은 얼굴 LoRA 또는 낮은 순위 적응 모델을 온라인으로 훈련하기 위해 몇 개의 이미지를 업로드하여 디지털 도플갱어를 생성하도록 사용자에게 요청합니다.

LoRA 프레임워크는 하위 적응 기술을 사용하여 확산 모델을 신속하게 미세 조정합니다.

이 과정을 통해 기반 모델은 특정 사용자의 ID 정보를 이해할 수 있습니다.

그런 다음 훈련된 모델은 간섭을 위해 기존 안정 확산 모델에 병합 및 통합됩니다.

또한 간섭 과정에서 모델은 간섭 템플릿의 얼굴 영역을 다시 칠하기 위해 안정적인 확산 모델을 사용하고 다양한 ControlNet 장치를 사용하여 입력 이미지와 출력 이미지 간의 유사성을 검증합니다.

요약하자면, EasyPhoto 모델은

1. 생성된 이미지의 얼굴 충실도를 유지하기 위해 여러 LoRA 모델을 통합하여 LoRA 모델을 훈련하는 새로운 접근 방식을 제안합니다.
2. 다양한 강화 학습 방법을 사용하여 얼굴 신원 보상을 위한 LoRA 모델을 최적화합니다. 이는 학습 이미지와 생성된 결과 간의 신원 유사성을 향상시키는 데 더욱 도움이 됩니다.
3. 높은 심미성과 유사성을 갖춘 AI 사진 생성을 목표로 하는 이중 단계 인페인팅 기반 확산 프로세스를 제안합니다.

보시다시피 모델은 먼저 사용자에게 훈련 이미지를 입력하도록 요청한 다음 얼굴 감지를 수행하여 얼굴 위치를 감지 & 초점을 맞춘 사전 정의된 특정 비율을 사용하여 입력 이미지를 자릅니다.

그런 다음 모델은 피부 미화 및 돌출 감지 모델을 배포하여 깨끗하고 깨끗한 얼굴 훈련 이미지를 얻습니다.

또한 훈련 단계 동안 프레임워크에는 사용자 입력 이미지와 훈련된 LoRA 모델에 의해 생성된 확인 이미지 사이의 얼굴 ID 차이를 계산하는 중요한 검증 단계가 포함됩니다.

검증 단계에서는 LoRA 모델의 융합을 달성하고 궁극적으로 훈련된 LoRA 도플갱어, 즉 사용자의 정확한 디지털 표현으로 변환됩니다.

최적의 Face\_id 점수를 갖는 검증 이미지를 Face\_id 이미지로 선택하고, 이 Face\_id 이미지를 사용하여 간접 생성의 신원 유사성을 향상시킵니다.

양상불 프로세스를 기반으로 모델은 우도 추정이 주요 목표인 LoRA 모델을 훈련하는 반면, 얼굴 신원 유사성을 유지하는 것은 다운스트림 목표입니다.

(maximum likelihood estimation as the objective)

논문에서는 결과적으로 LoRA 모델이 학습하는 얼굴 특징은 템플릿 생성 결과 간의 유사성을 향상시키는 개선을 보여주고 템플릿 전반에 걸친 일반화도 보여줍니다.

## Experiments

### ▼ 로라 실험 결과

아래의 사진은 뉴진스의 다니엘라와 해린의 이미지를 통해 lora를 학습을 진행했지만, 웬지만큼 학습은 되어도 추론에서 좋은 결과를 얻을 수 가 없었습니다.

SSD-1B의 경우 추론속도 빠르나 품질이 만족스럽지 않았고, 결국 SDXL과 SD1.5 기반으로 결정하게 되었습니다.

얼굴사진 학습시 512\*512인 SD1.5가 학습 상태가 좋았기에 SD 1.5로 결정

셋 다 LoRA 를 제작시 얼굴 부분을 크롭 및 text matching 시켜줘야 하는데 EasyPhoto가 만들어주는 LoRA 성능이 높았고, 심지어 FaceSwap시에도 적절하였습니다. 그 역할을 더 잘 수행하였기 때문에, EasyPhoto로 결정하여 LoRA만을 따로 학습만 진행하여 사용자 개인의 LoRA를 생성했습니다.

easyphoto 학습 수 또한 800대에서는 20분 초중반,

1000대에서는 30분초

1600대는 50분대

2000은 길어도 70분 정도 걸립니다.

그라디오에서 학습이 너무 크면 아웃풋이 생성됨에도 출력이 안되는 현상이 있었습니다.

성능은 2000대에서는 가끔 좋은 결과가 나오나 대부분 overfitting하게 로라가 학습되는게 보였고, 800~1000대는 너무 적게나 어중간한 느낌이 나오는 경우가 많았습니다.

1000에서 2000대 사이로 타협을 보는게 좋았고, 저희는 1600으로 스텝을 정했습니다.

## Adetailer

T2I 모델안에서는 저희는 Adetailer라는 기능을 이용했습니다.

### ▼ Adetailer란?

스테이블 디퓨전 사용시 얼굴 뭉개짐을 방지할 수 있는 확장 기능 입니다.

사용한 모델은 제가 직접 학습한 ultralytics의 yolov8과, mediapipe의 얼굴인식 모델입니다.

- hugginc face 지표 소개

작동방식은

1. 이미지 detection을 통해 부분을 인식
2. 인식한 부분을 새로 묘사한 프롬프트대로 이미지 생성하는 과정을 밟는다.

=> inpaint의 방식을 취해 작동합니다.

After Detailer는 일그러진 얼굴을 복원하는데 사용할 수 있습니다. Face restoration은 CodeFormer 혹은 GFGAN과 같은 별도의 AI 모델을 사용해 얼굴을 복원합니다.

반면 After Detailer의 경우, 이미지에 포함된 얼굴/손 등을 감지한 후, 확대하여 인페인트를 실시하고 다시 원래의 이미지 축척에 맞게 줄이는 방식으로 작동됩니다.

Adetailer를 통해 얼굴과 손을 포착하여 inpaint을 하여 보정하는 역할을 하도록 하였습니다.

## ControlNet

### ▼ ControlNet

{논문 이름: Adding Conditional Control to Text-to-Image Diffusion Models (ControlNet)}

컨트롤넷은 스테이블 디퓨전 모델을 제어하기 위한 신경망 모델로서, 단독으로는 사용할 수 없고, 다른 스테이블 디퓨전 모델과 함께 사용해야 합니다.

스테이블 디퓨전에서 가장 기본적인 형태는 text-to-image, 즉 텍스트 프롬프트(prompt)를 조건부여(conditioning)로서 입력하면, 이를 바탕으로 이미지를 생성하는 것입니다. ControlNet은 조건부여를 하나 더 추가합니다.

ControlNet에도 매우 많은 종류 존재하지만, 여기에서는 인체 자세 감지를 예를 들어 보이겠습니다.

컨트롤넷에서 인체의 자세를 감지하는 모델로는 Openpose 가 있습니다. 이 모델은 손, 발, 머리 등의 위치와 같은 인간의 자세를 추출하는 빠른 키포인트(keypoint) 감지 모델입니다.

컨트롤넷 작동흐름을 설명하겠습니다.

Openpose를 이용한 ControlNet입니다.

Openpose는 입력된 이미지로 부터 키포인트를 추출하고, 이 키포인트의 위치를 포함하는 제어 맵(control map)으로 저장합니다.

이 제어 맵이 Stable Diffusion에 전달되어 텍스트 프롬프트와 함께 추가적인 조건부여로 사용됩니다. 이미지는 이처럼 두가지\_텍스트 프롬프트와 컨트롤넷 조건부여에 기초하여 생성됩니다.


ControlNet은 학습 가능한 네트워크 모듈을 Stable Diffusion 모델의 핵심인 U-Net(잡음 예측기)의 여기 저기에 부착하는 방식으로 작동합니다.

Stabel Diffusion 모델의 가중치는 훈련중 변경되지 않도록 고정됩니다. 학습 중에는 오직 부착된 모듈만 수정됩니다.

먼저, 부착된 네트워크 모듈의 가중치는 모두 0이므로, 새 모델은 학습되어 잠긴 모델을 활용할 수 있습니다.

학습중에는 학습용 이미지와 함께 두개의 조건부여(conditioning)이 제공됩니다. 이때, 각각의 전처리별 방법은 독립적으로 학습받게 됩니다.

잠긴 블록  과 학습가능한 블록  을 만든다.

1. 학습 가능한 블록으로만 학습하고, 잠긴 블록을 통해 기존 모델을 보존한다.
2. 이를 통해 원래 모델에서 파괴나 왜곡이 일어나지 않아 안전하다.
3.  때문에 더 좋은 결과를 낼 수 있다.

ControlNet을 간단하게 묘사하면 다음과 같다.

diffusion model의 parameter를 복사하여 새로운 학습 프레임워크를 원래 parameter와 병렬로 구성한다. 이를 각각 “trainable(학습 가능한) copy”와 “locked(학습 불가능한) copy”라고 부른다.

Locked copy는 기존 network의 성능인 이미지 생성에 필요한 representation을 유지하고 있다고 생각할 수 있다.

Trainable copy는 conditional control을 위해 여러 task-specific dataset에 대해 학습되는 프레임워크다.

Locked copy와 Trainable copy는 zero convolution을 통해 서로 연결된다. Zero convolution 또한 학습 가능한 레이어에 속한다.

이 부분은 그림을 보면 이해가 쉽습니다.

x가 들어가서 y가 나오는 구조는 diffusion process에 접목시키게 되면 특정 시점의 noised latent vector

$z_t$  가 input으로 들어가서 다음 시점의 noised latent vector  $z_{t-1}$  를 예측하는 것과 같다. 회색으로 된 neural network는 원래의

diffusion model로 파라미터가 고정된 채 변하지 않게끔 하면 사전 학습된 디퓨전 모델의 이미지를 만드는 성능을 해치지 않고 가만히 놔둘 수 있다.

좌측의 얼어있는 친구는 가만 놔두고 우측의 불타는 친구만 condition에 대해 학습한다고 생각하면 된다. Trainable copy이므로 fine-tuning 과정인데 원래의 parameter를 최대한 손상시키지 않겠다는 의도가 보이는 학습 구조가 된다.

## Method

그렇다면 구체적으로 어떻게 해당 학습이 효과적으로 conditioning을 할 수 있는지 수식적으로 살펴보도록 하자.

conditioning을 하는 neural network block은 흔히 우리가 알고있는 resnet에서의 bottleneck block이나 transformer의 multi-head attention block을 생각하면 편하다.

2D(이미지와 같은 형태)의 feature를 예시로 들어보자. 만약 feature map  $x \in R^{h \times w \times c}$ 가 정의되어 있다면, neural network block  $F_{\Theta}(\cdot)$ 는 블록에 포함되는 parameter  $\Theta$ 를 통해 input feature map  $x$ 를 transform하게 된다.  $y = F_{\Theta}(x)$

바로 이 과정이 앞서 그림에서 봤던 (a)에 해당된다.

이제부터 해당 parameter  $\Theta$ 는 잠궜을 것이다(학습하지 않을 것). 그리고 이를 똑같이 복사한 trainable parameter  $\Theta_c$ 는 잠궜은 친구와는 다르게 input condition  $c$ 를 input으로 받아 학습에 사용될 것이다.

참고로 더해지는 부분에 대해서는 네트워크가 activation을 저장해놓을 필요가 없기 때문에 학습 시에 메모리를 2배로 가질 필요성도 없어진다. Backpropagation을 통해 계산된 gradient는 학습 가능한 모델에 대해서만 optimization을 진행할 것이기 때문이다.

- zero convolution

이때 더해질 때 바로바로 이 논문에서 가장 중요한 녀석인 zero convolution이라는 개념이 사용되는데, 각 neural block의 앞/뒤로 하나씩 붙는다고 생각하면 된다.

앞/뒤에 붙는 녀석들을 각각  $Z_{\Theta_1}(\cdot)$ ,  $Z_{\Theta_2}(\cdot)$

라고 해보자. 물론 zero-convolution은 feature map의 크기를 변화시키면 안되기 때문에 1x1크기를 가지는 convolution이며 weight와 bias 모두 zero로 초기화된 상태로 학습이 시작된다.

위의 그림대로 원래의 output  $y$ 에 conditioning 함수를 거친 output을 더하면 다음과 같다.

그림 1

여기에서 대체 왜 weight 및 bias가 0으로 초기화된 'Zero convolution'이 사용되었는지 이유가 등장한다. Zero-convolution은 weight 및 bias가 모두 0이므로, input에 상관없이 처음엔 모두 0을 output으로 내뱉는다.

그림 2

즉 처음에는  $y_c = y$ 로 시작하게 된다. 해당 내용이 암시하는 것은 training이 시작되는 당시에는 ControlNet 구조에 의한 input/output 관계가 사전 학습된 diffusion의 input/output과 전혀 차이가 없다는 것이고, 이로 인해 optimization이 진행되기 전까지는 neural network 깊이가 증가함에 따라 영향을 끼치지 않는다는 것을 알 수 있다.

(b)ControlNet 에서 Zero convolution은 Convolution filter의 가중치 값이 0이 되도록 하는 기술입니다. 일반적으로 Convolution filter의 값이 0이면 결과값(Destination pixel)은 0이 됩니다. 그러나 ControlNet에서는 이를 역전파를 이용해 학습시키는데, 이를 통해 이미지의 실루엣처럼 남아서 가중치 부분이 조건부 생성을 지원합니다.

이는 ControlNet에서 이미 학습된 모델의 의미론을 보존하면서 새로운 조건을 학습시키는 핵심 기술 중 하나입니다. 이러한 ControlNet 구조는 다양한 이미지 생성 작업에서 사용될 수 있으며, 높은 수준의 이미지 생성 결과를 보장합니다.

- Gradient flow in zero conv

Gradient flow in zero convolution1×1convolution 구조를 가지는 zero convolution에 대한 연산 과정에 local gradient를 유도할 수 있다. 예컨대 input feature map  $I \in \mathbb{R}^{h \times w \times c}$  가 있을때 forward pass는

그림 1

이처럼 표현되고, zero convolution은 최적화 전까지는  $W=0$ ,  $B=0$ 이기 때문에  $I_{p,i}$  가 0이 아닌 모든 point에 대해서 그림 2 같이 정리된다.

Input에 대한 gradient는 0으로 만들지만 weight나 bias에 대한 gradient는 0이 아니기 때문에 학습이 가능하다. 왜냐하면 first step 만 지나게 되면 Hadamard product 기호인  $\odot$ 에 대해 그림 3

0이 아닌 weight를 만들기 때문에 바로 다음 step에서는 그림 4  
고로 학습이 잘된다.

정리하자면 아래처럼 된다.

그림 5

- loss function

위에는 loss function 수식

위에서 설명한 구조를 기존 stable diffusion에 구현한 구조는 위와 같다. Loss는 기존 diffusion algorithm에 task specific condition cf만 추가된 형태가 된다.

저희는 오픈포즈에서도 손가락등 자세하게 포착을 못하는 것이 아쉬웠고, 전처리 과정을 좀 더 좋은 결과를 위해 DW pose

#### ▼ Dwpose

- introduction

RTMPose를 뛰어넘는 논문 : Effective Whole-body Pose Estimation with Two-stages Distillation

그림 1 & 2 & 3

dwpose는 두단계의 knowledge distillation(KD)를 통한 효율적이고 정확한 pose estimation데이터 한계를 극복하여, hand gesture와 facial expression과 다양성 있는 데이터 사용을 통한 실생활을 타겟팅  
RTM Pose를 기반에 KD 방법과 데이터활용을 통한 성능 향상했습니다.(RTMPose-1: 64.8% -> 66.5%)

- method

two-stage pose distillation(TPD)를 제안

그림에서도 볼 수 있듯, 2가지 단계로 구성됩니다. 1단계에서는 feature 및 logit 수준에서 처음부터 student를 guiding 하는 pre-trained teacher가 있습니다.

그 다음 2단계에서는 self-KD 방식을 사용합니다.

자체 logit을 이용하여 레이블이 지정된 데이터 없이 head를 학습함으로써 간단하게 학습을 하되 상당한 성능 향상을 갖고옵니다.

- The First-stage distillation

$F_t, F_s$  : teacher와 student의 backbone에서 나온 feature

$T_i, S_i$  : teacher와 student의 output logits



1단계 distillation에서는 student가 teacher's feature  $F_t$  와 logit  $T_i$  을 강제로 학습하게 합니다.

- Feature-based distillation

Feature-based distillation는 student가 teacher의 backbone Layer를 따라할 수 있도록 하는 방법입니다. student의 feature  $F_s$ 와 teacher의 feature  $F_t$  의 차이를 계산하기 위해 MSELoss를 사용했습니다.

수식 1

여기서,  $F_s$  를  $F_t$  의 dimension과 맞추기 위한,  $f$ 는  $1 \times 1$  conv 입니다.

- Logit-based distillation

RTMPose는 keypoint localization을 수평 및 수직 좌표에 대한 분류 작업으로 처리하는 SimCC 기반 알고리즘으로 포인트를 예측하게 됩니다.

수식 2

그 형태에 맞춰서 logit-based KD를 수행합니다.

$N$ 은 batch에서 person sample을 뜻하고,  $K$ 는 keypoint num 입니다.

$W_{n,k}$  는 distinguish invisible keypoint를 위한 target weight mask 입니다.  $V_i$  는 label value 입니다.

그러나, label value와 다르게 invisible 키포인트는 teacher에 의한 합당한 값이 될 수 있기 때문에, 아래와 같은 수식으로 만들었다고 합니다. == 마치 confidence와 같은 것

logit의 distillation loss는 다음과 같이 정의 될 수 있습니다.

수식 3

- Weight-decay strategy for distillation

student의 loss는 다음과 같은 total loss로 정리할 수 있습니다.

여기서  $\alpha$ 와  $\beta$ 는 하이퍼파라미터입니다.

수식 4

detection distillation 방법인 TADF를 따라서, 점차 생기는 distillation의 패널티를 줄이기 위해서, weight decay 전략을 사용합니다. 이런 방법은 student가 라벨에 집중하고, 더 좋은 성능을 보일 수 있도록 하는데요.

그래서 이를 위해 시간 함수  $r(t)$ 를 사용하게 됩니다. 여기서  $t$ 는 current epoch을 뜻하고  $t_{\max}$ 는 total epoch을 뜻합니다.

수식 5

그리하여 First-stage distillation의 total loss는 다음과 같이 정의됩니다.

수식 6

- The Second-stage distillation

두번째 단계에서는 student model이 더 좋은 성능을 갖게하는 방법입니다. self로 학습하게 합니다.

이러한 방식으로 distillation을 통해 처음부터 학습이 되었는지 여부에 관계없이 student model을 개선할 수 있습니다.

pose estimator는 encoder, decoder 구조로 구성됩니다.

우선, train된 backbone과 그렇지 않은 head로 student를 만들고, 같은 학습된 backbone과 head로 teacher를 준비합니다.

그리고, student의 backbone은 freeze하고 head만 학습시에 사용합니다.

teacher와 student는 같은 구조이기 때문에 backbone에서 한번만 feature를 뽑으면 되고, 그 feature를 student와 teacher의 head에 입력하여 logit  $S_i$ 와  $T_i$ 를 얻어 냅니다.

식 3에 따라 2단계 distillation을 위해 logit  $L_{\text{logit}}$ 으로 student를 학습합니다. == logit-distillation에서 사용했던 같은 방식의 loss를 적용하는 것  
이렇게 되면 label로 계산된 값을  $L_{\text{ori}}$ 를 떨어뜨릴 수 있습니다.

loss scale에 대한 하이퍼파라미터를 나타내기 위해  $\gamma$  를 사용하면 second-stage distillation의 최종 손실은 다음과 같이 공식화 할 수 있습니다.

수식 7

이는 기존의 self-KD 방법과 달리 head-aware distillation 방법은 단 20%의 학습 시간으로 head에서 지식을 효율적으로 추출하고 localization capability를 더욱 향상시킬 수 있습니다.

- experiments

## Experiments

저희 컨트롤넷 실험으로는



pose 실험을 통해 10개 이상이 있었으나 최종적으로 8개에서 4개로 줄었다.

왜?

타율이 별로 좋지 않았음 하지만 최종적인 포즈들도 100센트의 좋은 타율을 어려웠던점이 아쉬웠습니다.

여러번 실험을 통해 잘 통하는 포즈를 찾는게 중요해 보입니다.

여기까지가 t2i 모델에서 쓰이는 모델들의 원리와 작동방식을 소개했습니다.

다음으로는 SD에서 중요한 프롬프트에 대해서 말하겠습니다.

## Prompt

### ▼ 프롬프트

- 해상도

해상도 또한 파라미터가 될 수 있습니다. 위의 해상도들에서 색이 칠해진 부분들의 해상도 크기가, sd1.5을 학습할 때 쓰였던 이미지들의 사이즈였기에, 위의 해상도일 때 사진들에서 프롬프트가 잘됩니다.

저희는 생성 속도와 퀄리티를 생각해서 2:3으로 비율로 정하여 출력합니다.

너무 해상도가 크면 생성의 속도부터 차이가 크게 납니다.

- 토큰

프롬프트에는 토큰수가 정해져있습니다.

Positive prompt 와 Negative prompt에서의 토큰이 둘다 75 이상으로 입력은 가능합니다. 하지만 그 경우에는 Prompt가 둘다 정확하게 유도가 되지않는 현상이나 잊혀지는 현상이 나타나게 됩니다.

75 토큰의 수는 SD1.5가 Clip을 통해 한번에 처리하는 토큰의 개수입니다. 그 이상은 75개가 들어가고 OpenClip을 통해 처리가 되게 됩니다. 그래서 프롬프트가 잊혀지거나 묻히는 경향이 나오는 거임.

그래서 저희는 오히려 적은 토큰으로 잘 뽑는 방법을 찾기 위해 노력했습니다.

- noise predictor

아래는 프롬프트의 토큰이 처리되는 과정입니다.

먼저 프롬프트에 포함된 단어들을 토큰생성기(Tokenize)가 토큰(token)이라고 하는 숫자로 변환합니다.

각각의 토큰은 임베딩(embedding)이라고하는 769 개 값의 벡터로 변환됩니다.(이게 AUTOMATIC1111에서 사용하는 임베딩과 동일합니다.)

다음으로 text transformer가 임베딩을 처리하여 noise predictor가 사용할 수 있는 데이터를 생성하게 됩니다.

- 토큰나이저

먼저 텍스트 프롬프트는 CLIP tokenizer를 통해 토큰으로 바뀌게 됩니다.

CLIP(Contrastive Language-Image Pre-Training)은 OPEN AI에서 개발한 딥러닝 모델로서, 이미지에 대한 텍스트 설명을 설명하기 위해 사용됩니다.

사람들은 단어를 읽을 수 있지만, 컴퓨터는 숫자를 읽을 뿐이고, 따라서 사람이 입력한 단어들은 토큰화를 통해 숫자로 변환됩니다.

- embedding

임베딩은 768개의 값을 갖는 벡터입니다. 각각의 토큰은 각각의 고유한 임베딩 벡터를 갖고 있습니다. 임베딩은 CLIP 모델에서 생성되어 고정되어 있습니다. 다시말해서 학습과정에서 임베딩이 결정되는 것  
적절한 임베딩을 찾아내면, 임의의 물체와 스타일을 발동시킬 수 있다는 것을 발견했습니다. 이것이 텍스트 인버전(textual inversion)이라고 하는 세밀 조정 기법입니다.

임베딩은 텍스트 트랜스포머(text transformer)에서 처리된 후 잡음 예측기에 공급됩니다.

트랜스포머(transformer)는 조건부여(conditioning)를 위한 범용 어댑터와 같습니다.

트랜스포머는 데이터를 추가로 처리할 뿐 만 아니라, 다양한 조건부여 방식을 수용할 수 있는 메커니즘도 제공합니다. 결국, 일정하게 학습한 데이터를 사용해도 사용자가 조건부여를 달리하면 다른 이미지가 만들어지게 됩니다.

- 교차 인지(cross-attention)

U-Net 전체에서 노이즈 예측기는 텍스트 트랜스포머의 출력을 여러 번 사용합니다. U-Net은 교차 인지(cross-attention) 메커니즘을 통해 이를 사용합니다. 여기가 바로 프롬프트가 이미지와 만나는 지점입니다.

"붉은 머리를 가진 여자"라는 프롬프트를 예로 들어 보겠습니다. 스테이블 디퓨전은 "붉은"과 "머리"이라는 두 단어를 함께 짝을 지어(= 이는 프롬프트 내의 교차 인지) 붉은 머리를 가진 여자를 생성합니다.

그런 다음 이 정보를 사용하여 Reverse diffusion과정에서 붉은 머리가 포함된 이미지로 유도됩니다. (= 프롬프트와 이미지 간 교차 인지)

위에서 설명한 LoRA 모델은 교차 인지(cross-attention) 모듈의 가중치를 수정하여 특정한 스타일을 변경합니다. 이 모듈을 통해 원하는 이미지를 유도할 수 있다는게 얼마나 중요한지 알 수 있습니다.

- Clip

CLIP 모델은 레이어로 구성된 구조를 가지고 있습니다. 각 레이어는 이전 레이어보다 더 구체적입니다.

예를 들어 레이어 1이 "사람"이라면 레이어 2는 "남성", "여성"이 될 수 있습니다: "남성" 및 "여성"이 될 수 있고, "남성"의 경로를 따라 가면 레이어 3이 될 수 있습니다: 남자, 소년, 청년, 아버지, 할아버지... 등이 될 수 있습니다.

이 방식이 클립 모델의 정확한 구조는 아니지만, 예를 들어 설명하기 위한 것입니다.

예를 들어 1.5 모델의 깊이는 12랭크입니다. 여기서 12번째 레이어는 텍스트 임베딩의 마지막 레이어입니다. 각 레이어에는 일정한 크기의 매트릭스가 있고, 각 레이어에는 추가 매트릭스가 있습니다.

클립 건너뛰기는 기본적으로 "텍스트 모델을 얼마나 정확하게 만들 것인지"에 대한 설정이라고 생각하면 됩니다.

예를 들어 들판에 서 있는 젊은 남성에게 대한 자세한 설명이 있는 경우 클립 단계가 낮을수록 "서 있는 남자", 더 깊게는 "서 있는 젊은 남자", "숲에 서 있는 젊은 남자" 등의 이미지가 표시될 수 있습니다.

클립 건너뛰기는 특별한 방식으로 구조화된 모델을 사용할 때 정말 유용합니다.

클립 건너뛰기는 클립을 사용하거나 클립을 사용하는 모델을 기반으로 하는 모델에서만 작동한다는 점을 기억하세요.

1.x 모델과 그 파생 모델에서와 같이. 2.0 모델 및 파생 모델은 OpenCLIP을 사용하기 때문에 CLIP과 상호 작용하지 않습니다.

12개의 CLIP 레이어 중 마지막 레이어에서 각 토큰 텍스트들의 의미가 하나의 feature 임베딩에 맞춰지도록 압축되어, 토큰의 본래 의미가 사라지기 때문이라고 한다.

- Clip 결론

그래서 프롬프트 너무 말을 안듣는다 하면 clip skip 1 도 괜찮다.

표현자유도를 더 주고 인체나 구도 안정성을 원한다면 3~5까지 올려봐도 괜찮았습니다.

그리고 실사에서는 2정도가 유명했습니다. 왜냐하면 대부분 모델이 2에서 개발 되었기 때문에, 우리도 2를 채택하였고, 베이스 모델의 제작자도 clip 2를 가지고 제작하였기에 가장 잘 나왔습니다.

프롬프트 연구에서 결론은 한것들을 이야기해서 하게끔 한다.

최종적으로 저희는 t2i단에서 사용자의 이미지를 받아 easy photo를 거쳐 유저의 특징을 가진 lora를 제작 프롬프트에 추가하여 유저의 얼굴을 가진 이미지를 생성하도록 하며 동시에 포즈를 컨디셔닝을 하여서 포즈 또한 같이 생성하게 합니다. 이 아웃풋을 i2i단으로 보냅니다.

## I2I

이제 다음 단 i2i 모델에 대해 설명하겠습니다.

i2i 와 t2i의 차이를 간단하게 설명하겠습니다.

### ▼ i2i

Image-to-Image에서는 이미지와 텍스트 프롬프트가 입력으로 사용됩니다. 생성된 이미지는 입력 이미지와 텍스트 프롬프트로 조건부여 (conditioning)됩니다.

먼저

1 단계 : 입력 이미지가 잠재 공간으로 인코딩 됩니다.

2 단계 : 잠재 이미지에 잡음 추가됩니다. 이때 잡음 제거 강도(Denoising strength) 값에 따라 추가되는 잡음의 양이 결정됩니다. 0으로 설정할 경우 잡음이 추가되지 않으며, 1일 경우, 잡음이 최대치로 추가되어, 잠재 이미지가 완전한 무작위 텐서(random tensor)가 됩니다.

3 단계 : 잡음 예측기(noise predictor) U-Net이 잠재 잡음 이미지와 텍스트 프롬프트를 입력으로 받아, 잠재 공간(4x64x64 텐서)에서 잡음을 예측합니다.

4 단계 : 잠재 이미지에서 잠재 잡음을 제거합니다.. 그 결과 새로운 잠재 이미지가 생성됩니다. 3단계와 4단계는 지정된 샘플링 수(예 20 회)만큼 반복됩니다.

5 단계 : 마지막으로 VAE(가변 자동 인코더)의 디코더가 잠재 이미지를 픽셀 공간으로 내보냅니다. 이렇게 나온 결과가 image-to-image의 결과가 됩니다.

요약하자면, image-to-image가 text-to-image와 다른 점은, 입력된 이미지에 약간의 잡음을 추가한 초기 잠재 이미지가 입력으로 사용되는 것 뿐입니다.

저희는 t2i의 아웃풋을 base image로 두고 t2i의 prompt를 조금 더 삭제하여 filter 느낌의 토큰들만 남겨 더 실사에 가게끔 지금의 이미지를 더럽히지않게끔 생성하고, 다시 나온 이미지들에서 adetailer를 통해 얼굴과 손등 부족한 부분들을 후보정을 하였습니다.

## Upscale

마지막으로 upscale에 대해서 설명하겠습니다.

### ▼ Upscale

- upscale 모델 비교  
upscale에는 3 methods to upscale images in Stable Diffusion (ControlNet tile upscale, SD upscale, AI upscale) 가 있습니다.

method 1 : ai sacler

webui 내부 기능으로 구현이 되어있는 것입니다.

method 2 : sd scaler

img2img에서 script 기능을 통해  
prompt를 적용이 되는 scaler  
i2i 처럼 denoise를 정도를 걸어 줄 수 있는것이 차이입니다.

method3

img2img에서 ultimate upscale을 이용 설정을 하고  
이번에는 ControlNet을 통해, tile\_resample preprocessor를 통해 이미지를 고정하고 샘플링하게끔 하는 방식이다.  
프롬프트에 tile의 이미지 정보를 참고해서 고정하도록 컨디셔닝을 주며 디테일을 올리는 방법이였습니다.

저희는 생성된 이미지의 오염이 없으면서, 디테일한 부분들을 올리며 해상도를 올리는 방식을 채택하게 되었습니다.

#### ▼ 모델 선택 실험 방식

쿨톤 과 웜톤으로 샘플러는 나눴습니다. 실험을 해보면서 , Euler a 와 DPM++ 2M Karras가 cool tone으로 이미지를 잘 뽑는것이 보였고,

Eular 와 DPM++ SDE Karras가 warm tone으로 이미지를 develop을 하는것이 보여 나눠 샘플러를 바꿔게 설정하였습니다.

- CFG 는 이미지 생성과정중 프롬프트를 얼마나 정확하게 따를지를 조절하는 설정으로 비교하며 모델을 찾아 선정하였습니다.
- 프로젝트에서 6000장 이상을 뽑으며 실험과 개발을 하였습니다.

서비스 구현 방식에 대해 간단히 말씀드리겠습니다.

저희는 DE단 web을 Gradio를 이용해 구현해놓았습니다. 그리고 Flask APi를 통해 gradio 와 webui Api를 연결 학습 과 추론을 진행하게하여 구현하였습니다.

#### • 느낀점

- DE 랑 DS 간에 같이 협업을 하면서 소통을 많이 진행함에도, 기본적인 배경지식이 조금씩 달라 거기서에서의 오해가 생길 수 있었고, 커뮤니케이션이 중요함을 다시한번 깨닫게 되었습니다.
- 사용하는 스택과 툴들도 올바른 목적과 근거에 맞춰 사용해야한다는 것을 많이 깨달았습니다.
- 프롬프트가 생각만큼 유도하고 조절하는것이 어려움을 프로젝트 내내 느꼈습니다.

마지막으로 시현영상으로 마무리하겠습니다.