

1. 서론

1.1 목적

본 문서는 Vision transformer 기술을 활용한 Video Inpainting이 포함된 동영상 편집 프로그램 (VIVE) 개발에 대한 상세 설계서이다. 인하대 컴퓨터공학과 'VIP'의 프로젝트를 위한 것으로, 이를 바탕으로 시스템을 설계 및 구현한다.

1.2 개요

본 프로젝트는 딥러닝 기술과 Vision Transformer를 바탕으로 Inpainting을 구현해 동영상 속 원하는 물체를 제거하고 변형된 이미지를 제공해 동영상에 대한 만족도, 초상권 문제 회피 그리고 동영상 결함 복구 등의 문제를 해결하고자 한다. 이때 우리는 Masking 하는 부분에선 PCVOS의 input, output을 수정해 구현하고 Inpainting 부분에선 직접 train을 시켜 제작하고자 한다.

1.3 개발 환경

- OS: Window 10, Linux
- 개발 도구: VS code, Google Colab
- 개발 언어: Python (Pytorch, GAN, OpenCV, PyQt)
- 개발 기간: 3개월 (2023.08.29-2023.12.05)
- 개발 인원: 3명

1.4 주요 기능

1. 파일 불러오기, 저장
2. 동영상 나누기
3. 자막 추가
4. 오디오 추가
5. Masking 할 물체 그리고 Masking 파일 생성
6. Video Inpainting

2. System 설계

2.1. Modules 개요

- Video Edit Module: 동영상 편집, 오디오, 플레이어 등 관련된 메소드들을 모아 놓은 모듈
- Masking Module: 마스킹과 관련된 메소드를 모아 놓은 모듈
- GAN Module: GAN의 생성기가 생성한 데이터와 원본 이미지 사이의 데이터 오차를 줄이는 방향으로 훈련하기 위한 메소드를 모아둔 모듈
- Train Module: 학습을 시켜 가중치(weight)를 .pth 파일로 저장한 뒤 새로운 동영상 데이터에서 인페인팅할 때 참조할 파일을 만드는 모듈
- VI Module: Masking Module에서 Output으로 나온 Masking된 이미지와 원본 프레임 단위 이미지들을 Input으로 받아 Inpainting을 실행하는데 필요한 메소드를 모아둔 모듈

2.2 Modules 상세 설계

Module Name			
Function Name	Input	Output	Explanation

Video Edit Module			
Make_video_player()	-	Video player	비디오 플레이어를 구성
Open_file_explorer()	Reference video	Reference video	파일탐색기를 실행하여 비디오 파일을 선택하고 선택한 파일 경로를 저장
Toggle_play_pause	Click action	Video state	버튼을 클릭 했을 때 재생 중일 경우 정지, 정지 중일 경우 재생
Close_video_player()	-	-	비디오 재생중인 경우 재생을 중지하고 비디오 자원을 해제
Update_frame()	Reference video	Reference video	프레임을 읽어와 화면에 표시
Manage_volume()	Drag Action	Volume	동영상 볼륨을 조정
Make_timeline()	-	Time line	타임라인을 생성
Move_cursor()	Time	Frame	타임라인 커서를 움직여 재생 위치를 재조정하고 해당 위치의 프레임으로 업데이트
Cut_video()	Start time End time	-	Start time 부터 end time까지 동영상을 자름
Move_cursor()	Time	-	Start frame cursor Endframe cursor의 위치 움직임
Add_subtitle()	Start_time End_time Subtitle	Edited video	자막 내용, 자막을 보여줄 시간을 입력하면 해당 시간대에 자막이

			표시
Open_file_explorer()	Audio file	Audio file	파일 탐색기를 실행하여 오디오 파일을 선택하고 선택한 파일 경로를 저장
Play_audio()	Audio file	Sound	오디오 파일을 실행해 오디오 출력
Move_cursor()	Start_time End_time	Audio	사용하고자 하는 오디오 부분을 지정할 수 있도록 cursor를 움직일 수 있도록 함
Cut_audio()	Audio file	Audio file	Frame이 편집 되는 것에 따라 audio도 변경되니 그 값을 계산해 알맞은 오디오가 출력되게 함

Set Data Module			
Cut_frame()	Video_name.mp4	Video_name[n].png	사용자가 업로드한 동영상 프레임 단위로 이미지 파일로 생성
Save_audio	Video_name.mp4	Video_audio.mp3	사용자가 업로드한 동 영상에서 오디오 파일 을 분리해 생성

Masking Module			
Extract_frame()	Video	Frame	영상에서 프레임 하나 를 추출
Color_frame()	Frame	Colored_frame	지우고자 하는 객체를 색칠할 수 있는 기능
Save_frame()	Colored_frame	Mask_frame	지우고자 하는 객체를 색칠한 후 그 파일을 저장해 PCVOS에 input할 수 있는 형태 로 저장
PCVOS()	Frame Mask image	Masking Frame	지우고자 한 객체를 추격해 프레임 단위로 Masking

GAN Module			
AdversarialLoss()	Loss function type, real_label, fake_label	Loss function type	손실 타입에 따라 손실 함수 나눔
Call()	Output, Model_label	Loss value	클래스 호출할 수 있게 함수 정의, 모델의 출력, 실제 여부 확인
Class Generator()	Super = nn.Module	Image	가짜 사진을 만드는 클래스
Ganlayer()	Data input 차원 수 Data output 차원 수	pipeline	레이어들로 이루어진 리스트 반환하여 모델 구성하는데 사용
Class Discriminator()	Super = nn.Module	진짜인지 가짜인지 확률 출력	이미지가 진짜인지 가짜인지 판별함
Forward()	img	확률 출력	이미지를 평탄화 이미지 판별

Train Module			
Dataset()	Index	Frame tensor Mask tensor	Index에 해당하는 이미지와 마스크를 반환
Get_ref_index()	Len Sample_len	Ref_index	참조 index를 반환
Patch()	Frame	Frame tensor	각 image를 patch 단위로 나누고 tensor로 변환한다
Position()	T, w, h, p, s, k	Patch position	Frame의 t, w, h, p, s, k 값을 입력해 각 patch의 위치 정보를 저장한다.
Weight_sum()	Weight_frame	Weight_frame	겹치는 부분의 가중합을 한다. 가중합을 할지 평균으로 할지

VI Module			
Get_ref_index()	Ne_ids(neighbor), len	Index	전체 비디오에서 샘플 참조 프레임들을 가져온다.
Read_mask	Mask_path	Mask_image	마스크 이미지들을 불러온다.
Read_frame	Args (딕셔너리)	Frame_image	이미지를 RGB 형식 (np.uint8)으로 변환하고 PIL 이미지로 변환
Image.fromarray()	Np 배열 형태 이미지	PIL_image	PIL 이미지로 변환

2.3 Deep learning Model 설계

우리는 Deep Video Inpainting[2]에서 CNN으로 구현된 것과 다르게 우리는 GAN과 Vision Transformer를 활용해 Inpainting을 구현하고자 한다. CNN과 달리 Transformer 같은 경우 input으로 들어오는 image의 크기에 크게 신경 쓰지 않아도 되기 때문에 Dataset과 편집 프로그램으로 제작하는 경우 큰 긍정적 효과를 얻을 수 있다.

Vision Transformer는 image frame을 작은 patch로 잘라 transformer에 넣는 방식인데, image를 patch 형태로 자르는 경우 각 patch에 Attention value가 적용되어 상호간의 정보가 단절되어 image를 온전히 구현하기가 어렵다. 이를 해결하기 위해 patch를 자를 때 겹쳐서 만들면 상호간의 정보 교류가 발생하여 결과물의 질이 상승할 거라 기대가 된다.

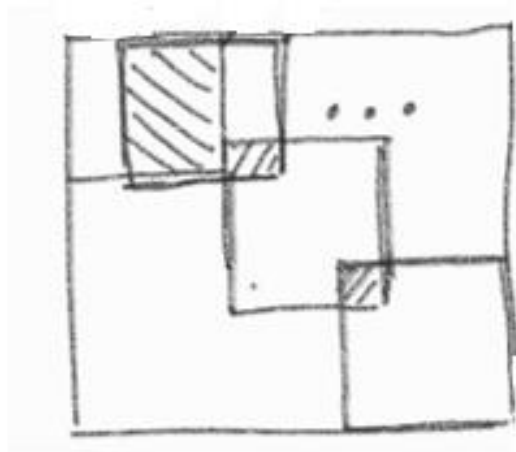


Fig.1 patch를 겹치는 부분이 있게 나누는 경우

이렇게 겹친 부분은 가중합을 하거나 값들의 평균으로 대체해서 다시 최소 patch로 나눈다.

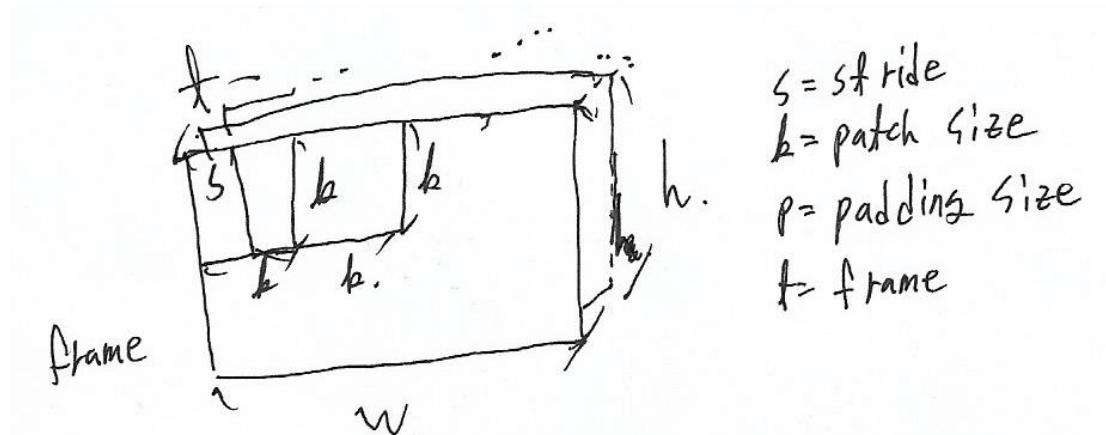


Fig. 2 각 frame 별 patch 모양

$$\text{token } n = \left\lfloor \frac{h+2p-k}{s} + 1 \right\rfloor \times \left\lfloor \frac{w+2p-k}{s} + 1 \right\rfloor$$

Fig. 3 frame 1개 당 token의 개수

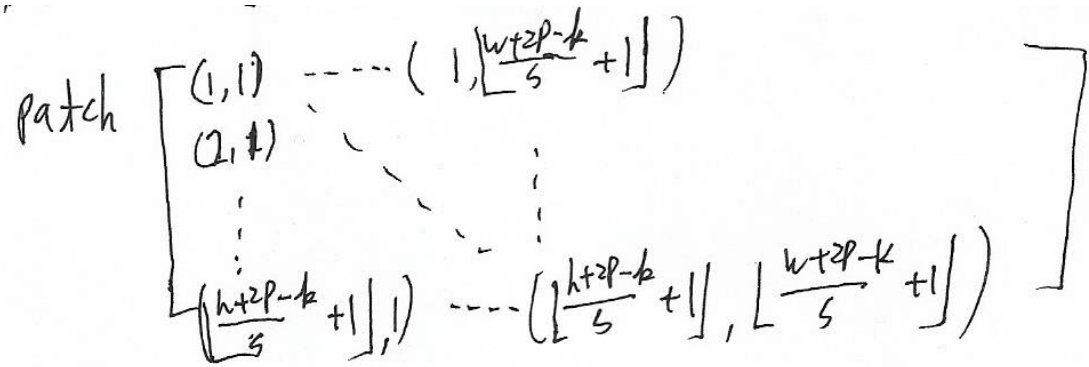


Fig. 4 Position Embedding을 위한 각 patch 별 위치 정보, 3차원에선 t 순서 추가

Image 안에서 연관성을 살렸으니 이제는 Video의 특징인 image간 시간 차원에서의 연관성을 살려야 한다. 이 문제를 해결하기 위해 Deep Video Inpainting[2]이 발표한 방법론 x_t , x_{t-6} , x_{t-3} , x_{t+3} , x_{t+6} frame들을 같이 연관 지었는데 우리는 손상된 t개의 frame을 input으로 넣을 것이다. Train 시간이 늘어나는 경우 t의 값을 줄이는 쪽으로 해결한다.

각각의 patch들을 1차원 이미지로 변환하면서 Position Embedding를 추가한다. Position Embedding은 위치 정보들을 포함시켜 입력시킬 수 있도록 사용하는 위치 정보를 담은 벡터를 생성한다. 이것을 바탕으로 각 frame마다의 위치되는 patch를 역추적하여 본 위치로 갈 수 있게 한다.

DVI 의 경우 ViNet 이란 Network 를 사용하는데 우리는 transformer 에서 자주 쓰이는 Feed Forward Network 를 사용할 것이다. 이는 이미지를 입력으로 받아 특정 객체를 인식하고 분류하는데 우리는 여기서 기본적인 Feed Forward Network 에 Patch 부분의 가중합을 하거나 값들의 평균으로 겹친 부분을 대체하는 연산을 추가해야 한다.

1 개의 frame 에서 나오는 token n 개 \times frame 개수 t 를 Position Embedding 을 하고 Transformer 에 대입한다. 그 후 GAN 을 통해 이미지를 생성하고, 다시 각각의 patch 와 position 정보를 통해 frame 을 복구한다. 여기서 frame 들을 복구할 때 겹치는 곳을 가중합을 한다.

GAN 을 활용해 이미지를 생성할 때 가짜 이미지인지 아닌지 판단하는 Loss 같은 경우

$$Loss = \lambda_R L_R + \lambda_{adv} L_{adv} \quad (1)$$

$$L_R = \text{생성된 이미지와 실제 이미지 사이의 Norm 1 Loss} \quad (2)$$

$$L_{adv} = \text{GAN 의 D 를 속여 생성된 이미지에 대해 판별 값을 1 로 나타내도록 학습} \quad (3)$$

$$V_D = E_Y[\log D(Y)] + E_{\tilde{Y}}[\log(1 - D(\tilde{Y}))] \quad (4)$$

Discriminator 는 실제 이미지는 1 로 생성된 가짜 이미지는 0 으로 판별하도록 학습한다.

$\lambda_R L_R$ 은 모든 픽셀에 대하여 재구성 손실을 의미하는데 이미지를 생성했을 때 복원하는 작업에서 사용된다. 여기서 Loss function 이 다양하지만 기본적인 MSE 를 적용할 것이다.

$\lambda_{adv} L_{adv}$ 은 생성자와 판별자 간의 경쟁을 기반으로 적대적 손실을 가리킨다. 여기서 손실 함수의 타입은 'nsgan' 또는 'lsgan'으로 한다.

$\lambda_R L_R$ 이 크면 재구성의 손실의 중요도가 높아지고 $\lambda_{adv} L_{adv}$ 이 크면 GAN 손실의 중요도가 높아진다.

우리의 모델은 기본적인 Vision Transformer 에서 크게 벗어나지 않고 input 으로 여러 개의 frame 이 들어가는 것을 인지해야 한다. 서로 겹치는 부분을 생성해 하나의 frame 안에서 patch 간의 연관성을 올렸고 여러 개의 frame 을 넣어 시간의 축의 문제를 해결했다.

Patch 의 크기인 k 와 몇 개의 frame 을 넣을 지인 t , train 횟수 T , padding p , patch size k , stride s , loss function 수정 등으로 train 속도와 학습 결과를 조정할 수 있다.

2.4 Dataset

우리는 오픈 되어 있는 Dataset을 활용한다. 두가지의 Dataset을 구할 수 있는데 YouTube-VOS와 DAVIS가 있고 이 Data들은 Train, Test, Valid로 구분된 프레임별 이미지가 저장되어 있다.

이 Dataset을 활용해 Train을 하고 다른 Model들과 성능을 비교해 우리의 목표인 Deep Video Inpainting의 ViNet 모델보다 높은 성능(DAVIS dataset 기준 PSNR: 28.96, SSIM: 0.9411)을 넘는 것을 목표로 한다.


2.5 프로그램의 input data의 변형 순서도

우리가 프로그램을 실행했을 경우,

- 1) mp4 파일로 받아 frame 단위로 자르고 파일을 생성하고 오디오 파일을 따로 추출한다.
- 2) Masking 할 부분을 하나의 첫 frame에 그린다.
- 3) PCVOS를 사용해 나머지 frame들에서 물체를 추적해 masking한다.
- 4) 원본 frame과 masking frame 폴더들을 input으로 Video Inpainting을 작업한다.
- 5) 이때까지의 Video의 길이와 오디오의 길이는 변함없어야 한다.
- 6) 그 외의 편집 기능을 실행해 data를 변형한다.

3.1 User Interface Design

Home

 Inpainting Video Editor

편집

Inpainting Video Editor - 온라인 동영상 편집기

동영상의 길이를 조절하고 원하지 않는 물체를 지울 수 있습니다.

Inpainting Video Editor로 새로운 동영상을 만드는 방법

1

파일 추가

동영상을 편집하려면, 편집 버튼을 클릭하고 편집할 동영상을 선택하십시오.

2

비디오 편집

편집에 사용할 방법을 선택하고 방법에 맞게 동영상을 편집하십시오.

3

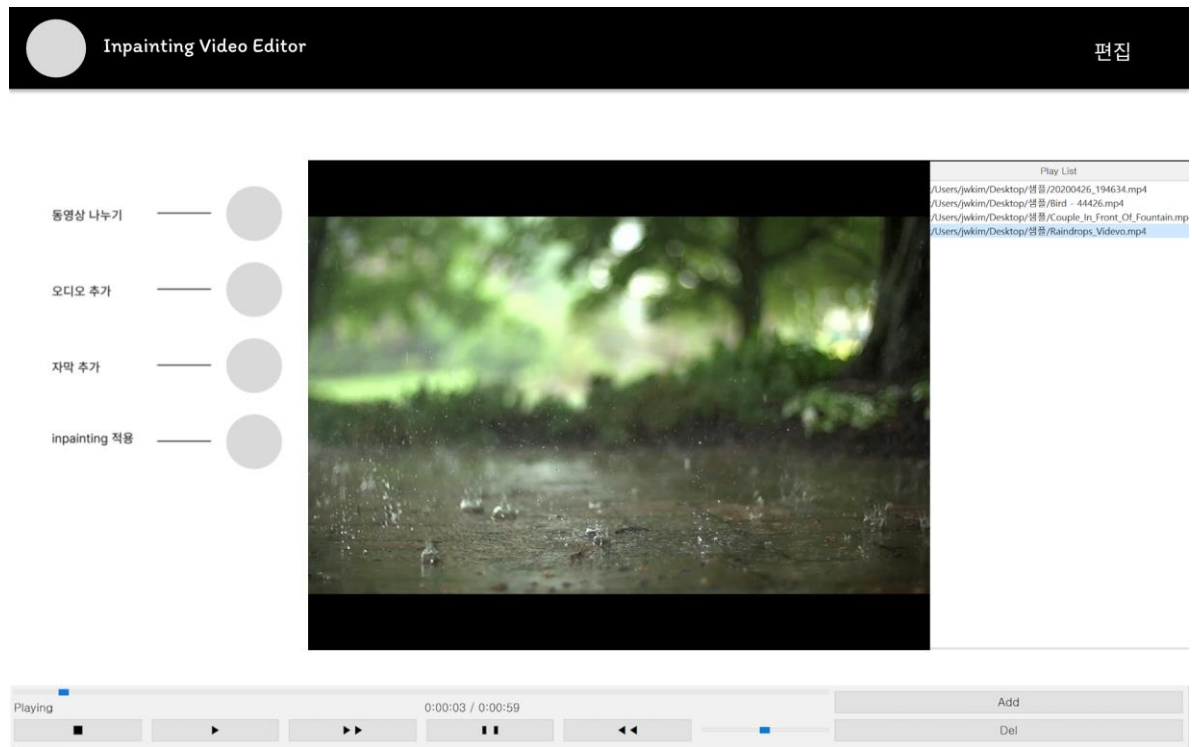
결과 저장

편집이 완료되면 동영상 저장 버튼을 눌러 동영상을 저장하십시오.

- 프로그램 실행시 처음 나타나는 화면이다.
- 해당 프로그램의 사용방법을 간단하게 설명한다.

① 편집 버튼
: 영상을 만드는 화면인 *Edit_View 1* 페이지로 이동한다.

Edit_View 1



■ 편집 버튼을 누르면 나타나는 화면으로 영상을 업로드하는 화면이다.

① 파일 첨부

: Add버튼을 클릭하면 영상을 첨부하는 기능이 가능하다.

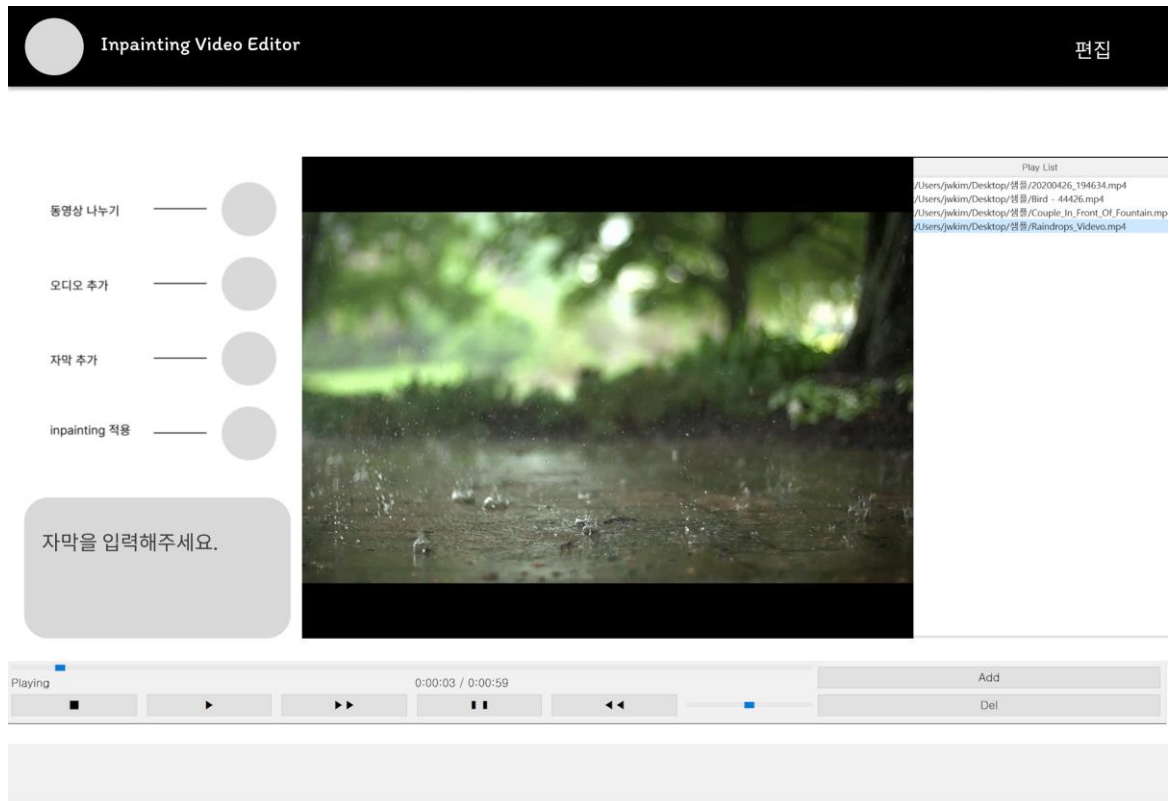
② 메뉴

: 동영상 나누기, 오디오 추가, 자막 추가, inpainting의 기능을 선택할 수 있다.

③ Track

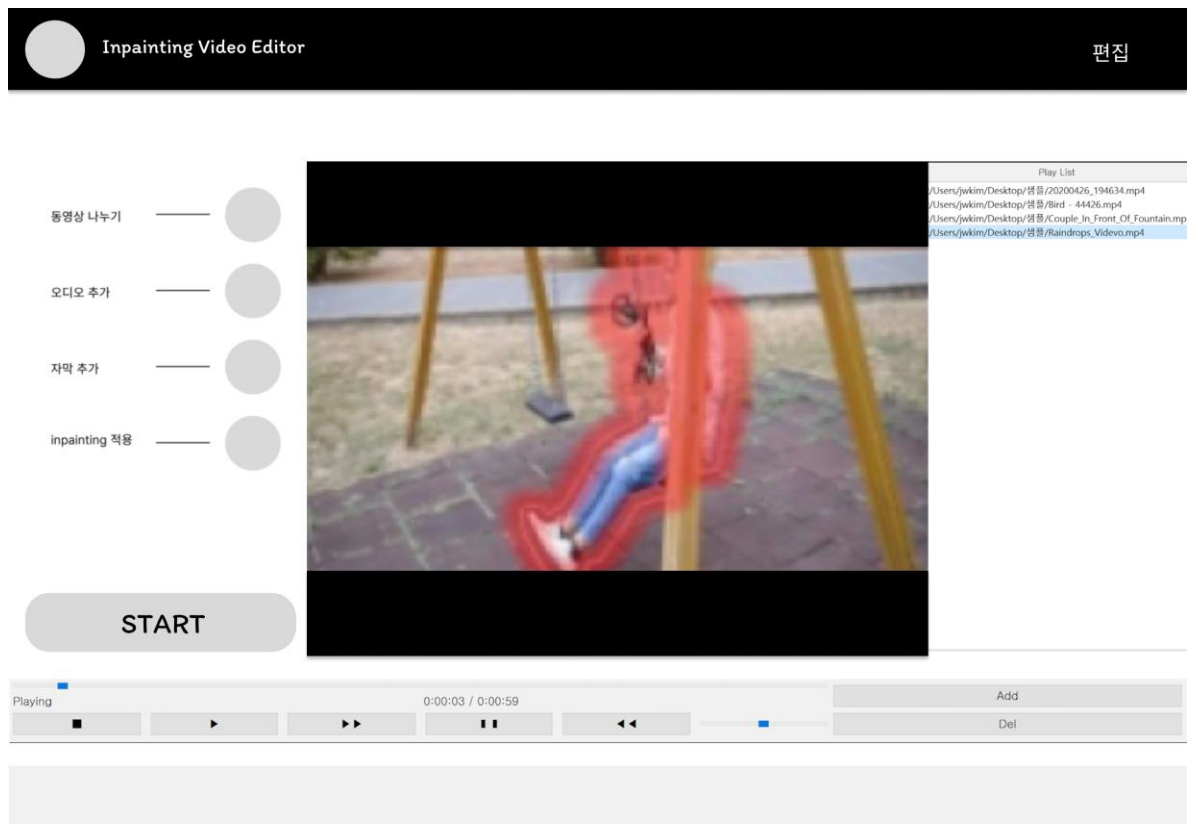
: 영상을 슬라이더를 드래그하여 이동할 수 있고, 정지버튼과 재생버튼 등으로 제어가 가능하다.

Edit_View 2



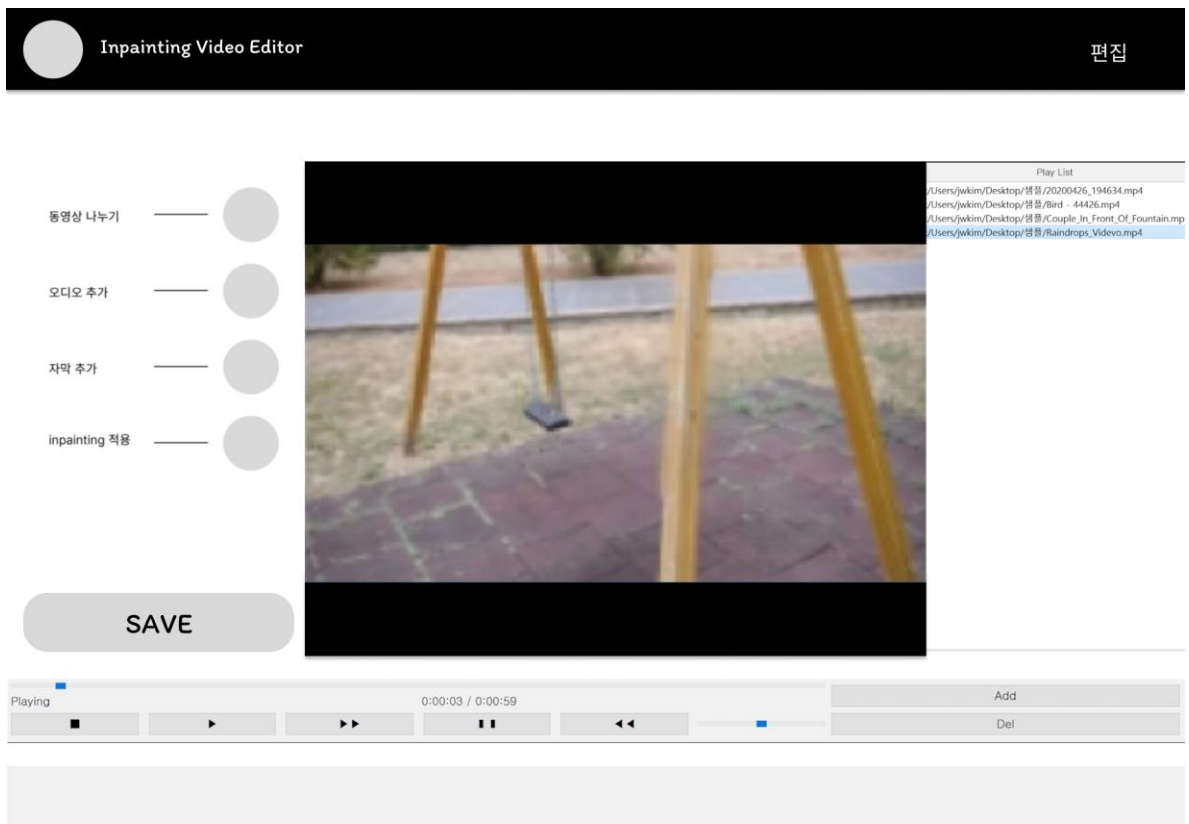
- 메뉴 중 자막 추가를 클릭 시 트랙이 추가로 생성되며 영상의 좌측에 자막을 입력할 수 있는 텍스트 박스가 생성된다.
- 트랙에서 자막을 추가하고 싶은 위치를 클릭 후 자막 입력 텍스트 박스에서 자막을 입력하면 자막이 추가된다.

Edit_View 3



- 메뉴 중 Inpainting 적용 클릭 시 보여지는 화면이다.
- 아래의 슬라이더를 조절하여 원하는 시간대로 이동하고 위의 화면처럼 지우고 싶은 객체를 색칠한다.
- START 버튼을 클릭하면 Inpainting 이 적용되어 해당 객체가 지워진다.

Edit_View 4



- Edit_View 3 에서 START 버튼을 클릭하면 Inpainting 이 적용된 후의 영상을 보여준다.
- 영상을 아래의 버튼들과 트랙을 이용하여 재생할 수 있다.
- SAVE 버튼을 클릭하면 영상을 저장할 수 있다.

4. WBS

	7	8	9	10	11	12	13	14
개발 환경 구축								
UI 구성								
편집 기능 개발								
masking 기능 개발								
PCVOS 수정								
생성 모델 구축								
.pth 모델 구축								
Traing								
성능 테스트								
Inpainting 기능 개발								
PCVOS와 Inpainting 모듈 합성								
전체 모듈 합성								
테스트 및 수정								
최종 기능 구현 테스트								

5. 참고 자료

5.1 참고 자료 및 논문

- [1] Per-Clip Video Object Segmentation, <https://arxiv.org/abs/2208.01924>
- [2] Deep Video Inpainting, <https://arxiv.org/abs/1905.01639>
- [3] GAN, https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- [4] Gan Introduce, <https://pseudo-lab.github.io/Tutorial-Book/chapters/GAN/Ch1-Introduction.html>
- [5] 데이터 전처리, <https://pseudo-lab.github.io/Tutorial-Book/chapters/object-detection/Ch3-preprocessing.html>
- [6] ViT, <https://browse.arxiv.org/pdf/2010.11929.pdf>
- [7] Attention Is All You Need, <https://browse.arxiv.org/pdf/1706.03762.pdf>
- [8] Rich feature hierarchies for accurate object detection and semantic segmentation, <https://arxiv.org/abs/1311.2524>
- [9] GAN – loss 총 정리, <https://velog.io/@simba-pumba/GAN-loss-%EC%B4%9D%EC%A0%95%EB%A6%AC>

5.2 PCVOS에서 쓸 모듈-함수 정리

TM Module			
All_to_onehot	Masks Labels	Onehot 인코딩된 마스크	Masks 배열에서 각 라벨에 해당하는 마스크 추출 및 형태 반환
Class TestDataset()	Torch-Dataset	Data 전처리	Data를 로드하고 전처리함
__init__()	Data_root split	-	데이터를 분할함
__getitem__()	idx	딕셔너리 data	Idx를 기반으로 데이터 로드 후 전처리
__len__()	-	Dataset의 길이	Dataset의 전체 길이 반환
Aggregate()	Prob(확률)	확률을 집계한 Tensor	주어진 확률 값들을 집계하고 반환
Class BasicConv()	nn.Module	-	기본적인 합성곱 레이어 정의
Forward()	Input data	합성곱	합성곱을 수행하고 결과 반환
Class Flatten()	nn.Module Input data	Input data 평탄화된 형태	입력 데이터 차원 평탄화
Class ChannelGate()	nn.Module	채널간의 관계 계산	채널 간의 관계 강조
Class ResBlock()	nn.Module input data	합성곱	Reidual block 정의 합성곱 연산
Class FeatureFusionBlock()	nn.Module indim f16	Outdim	입력 특성과 다른 크기의 특성을 결합해 특성 향상
ResBlock()	Indim	Outdim	정보 손실을 막고 더 깊은 네트워크를 학습
Torch.cat()	Tensor1 Tensor2	Tensor	tensor들을 합친다
Intra-Clip Refinement Module			
Class Mlp()	nn.Module	-	다층 퍼셉트론 (MLP) 정의
Class ICP()	nn.Module	-	내부 클립을 미세 조정하는 메인 모듈
_freeze_stages()	Self	-	네트워크에서 지정된 수의 스테이지를 동결

Init_weights()	Self	-	네트워크의 백본 부분에서 가중치를 초기화
Forward()	Key Value	Value	Feature를 미세하게 조정하기 위해 레이어를 순차적으로 적용
Build_Intra_Clip_Refinement()	Args	ICR	클래스의 인스턴스를 생성해 반환
ValueEncoder Module			
Class ValueEncoder()	Img Feature Masks	Incoding value	인코딩 값 반환
KeyEncoder Module			
KeyEncoder()	Img	Feature	이미지를 받아 중요한 특성 추출
Decoder Module			
Decoder()	Feature	Img	고수준 특징을 입력으로 받아 낮은 해상도의 이미지 생성
Memory_bank Module (+Memory Matching Module)			
_global_matching()	두개의 메모리 쿼리	Affinity	유사도 계산
_readout()	affinity	Torch.bmm(mv,affinity)	Tensor 행렬 곱
Match_memory()	Qk	Qk information	메모리 은행에서 해당 정보 검색
Add_memory()	Key Value	-	메모리 key, value값을 추가로 입력함

5.3 주요 기술들의 기본 정의

5.3.1 Tracking and Masking (Per-Clip PCVOS Model)

* Per-Clip Video Object Segmentation (PCVOS) Model 개요

Object Detection는 CNN을 통해 Data를 labeling해 분류하고 그 물체가 어디 있는지 박스 (Bounding Box) 위치 정보를 나타내는 Localization 문제를 둘 다 해결해내는 분야이다.

이 분야의 발전 흐름으로는 Deep Learning을 이용한 Object Detection은 크게 1-stage Detector와 2-stage Detector로 나눌 수 있다. 우리가 흔히 알고 있는 모델 YOLO는 대표적인 1-stage Detector이고 아래 R-CNN이 대표적인 2-stage Detector이다.

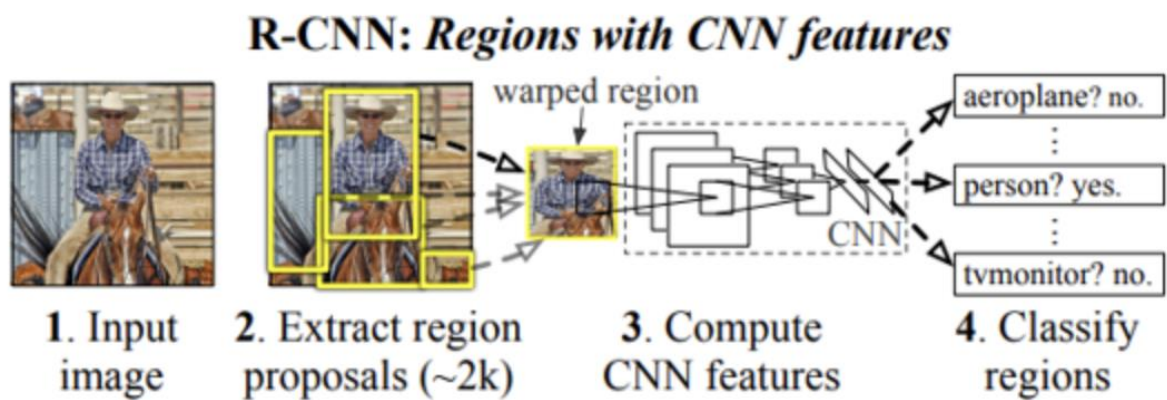


Fig.1 R-CNN : Regions with CNN features [8]

그리고 우리가 사용할 Per-Clip Video Object Segmentation은 2-stage Detector의 최신 기술 중 하나이다. 2-Stage Detector의 특징은 Localization문제와 Classification문제를 순차적으로 실행해 비교적 느리지만 정확도가 높다.

최근 memory-based approaches는 이전 준지도 비디오 물체 분할에 대한 유망한 결과를 보여준다. 이 방식은 이전 마스크의 빈번하게 업데이트 되는 메모리를 이용하여 객체 마스크를 프레임 단위로 예측한다. 이 프레임별 추론과 달리, PCVOS는 비디오 객체 분할을 클립 단위 마스크 전파로 취급하였다.

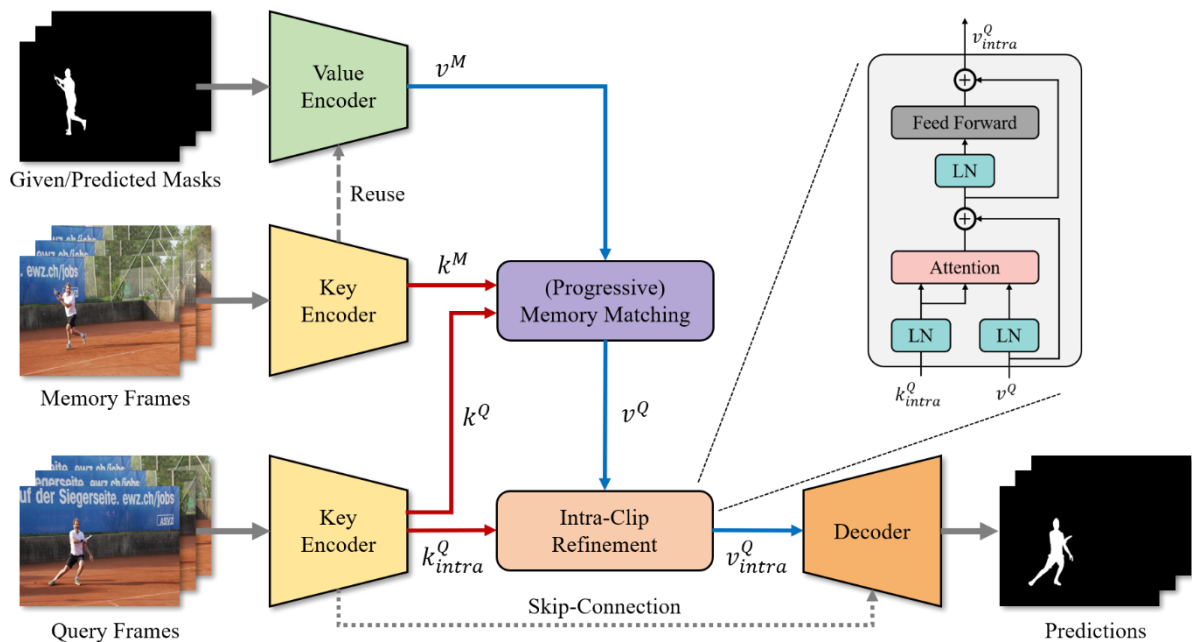


Fig. 2 The Overview of the Proposed Framework [1]

PCVOS는 5개의 모듈이 있다.

- 1) 메모리와 쿼리 프레임 사이에 공간 temporal 대응관계 구축하는데 사용하는 주요 특징을 추출한 Key Encoder
- 2) Value Encoder: 네트워크가 값 특징에 이전 마스크 정보를 포함한다.
- 3) 초기에 메모리로부터 value 정보를 검색하는 Memory Matching
- 4) 변환기가 Clip 내 상관 관계를 활용하여 검색된 값 특징을 정제하는 clip내 정제 모듈
- 5) 정제된 정보를 받아 마스크 결과를 예측하는 Decoder

또한 클립당 추론을 위한 모델을 전문화하기 위해 새로운 훈련 체계와 메모리 매칭 모듈의 변형인 점진적 메모리 매칭 매커니즘을 적용되었다.

5.3.2 GAN

* Generative Adversarial Networks Model 설명

GAN은 단순히 특정한 이미지를 이해하고 분류하는 것을 넘어서 실제 데이터와 흡사한 가짜 데이터를 생성하는 알고리즘이다. GAN은 생성자(Generator) 모델과 구분자(Discriminator) 모델로 두 개의 모델이 학습하는 방식으로 동작한다.

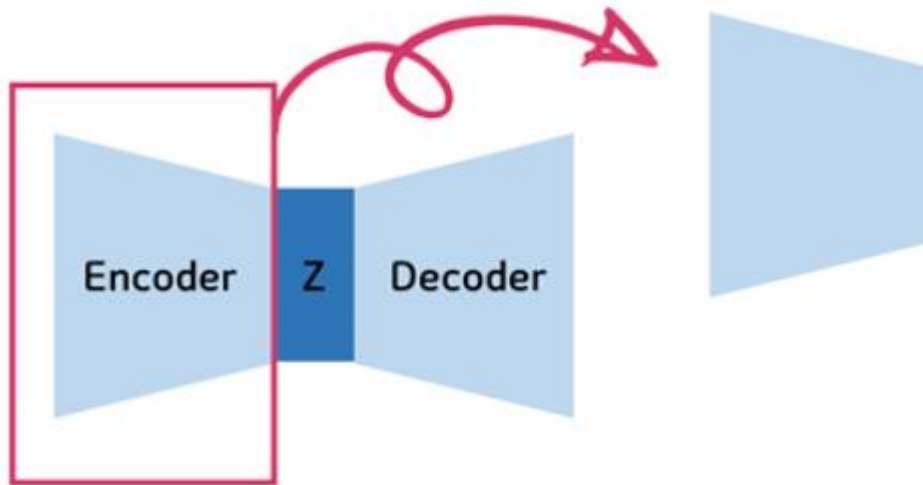


Fig.3 Encoder & Decoder GAN 학습 과정[4]

구분자 모델은 먼저 진짜 데이터를 진짜로 분류하도록 학습시키고 생성자 모델이 생성한 데이터를 가짜로 분류하도록 학습시킨다. 그 후 학습된 구분자 모델을 속이는 방향으로 생성자 모델을 학습시킨다.

1. Discriminator 학습

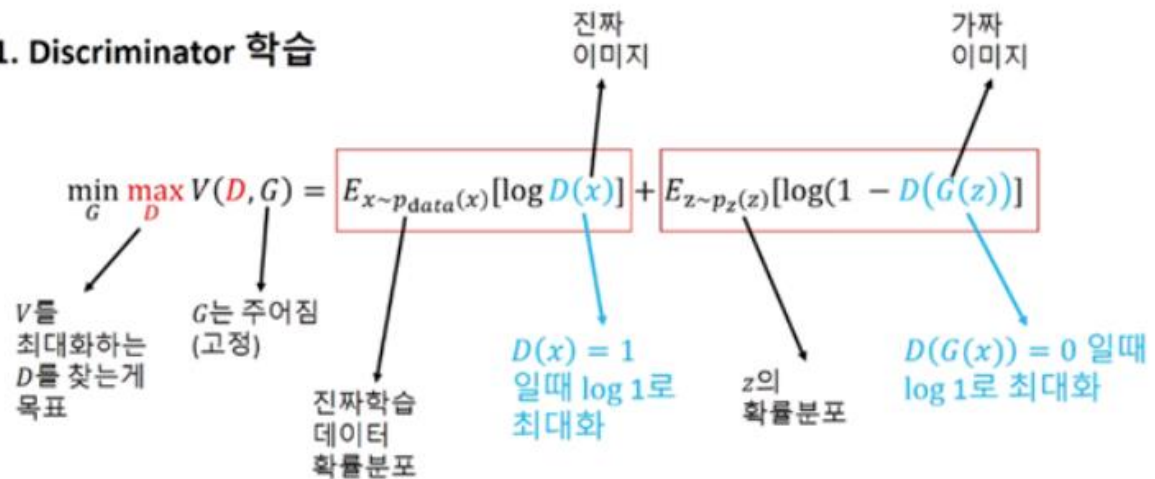


Fig.4 Discriminator 학습[3]

2. Generator 학습

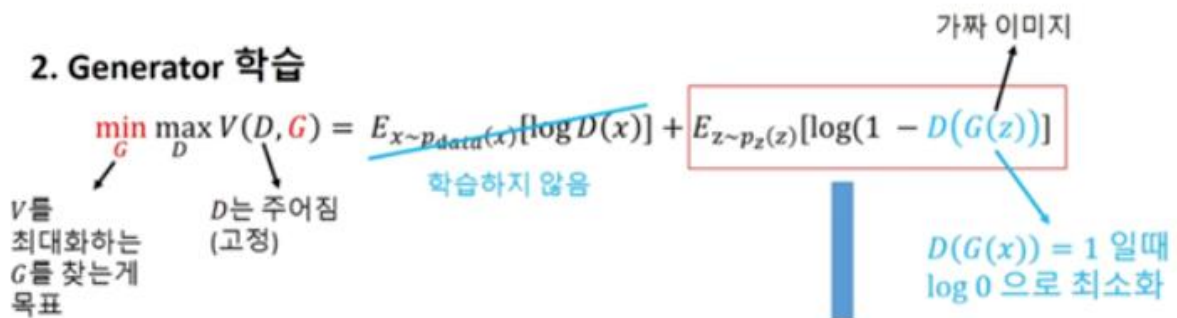


Fig.5 Generator 학습[3]

이 방정식을 D 의 입장, G 의 입장에서 바라보면 D 의 입장에선 V 를 최대화하는 D 를 찾는 것이 목표이고 $D(x) = 1$ 일 때 $\log 1$ 로 최대가 된다. 또한 $D(G(z)) = 0$ 일 때 $\log 1$ 로 최대화가 된다.

G 의 입장에선 V 를 최대화하는 G 를 찾는 것이 목표이고 D 값은 고정이기 때문에 $\log D(x)$ 는 학습하지 않는다. 또한 $D(G(z)) = 1$ 일 때 최소화가 되기에 이상적인 결과, 최솟값은 마이너스 무한대임을 확인할 수 있다.

5.3.3 Vision Transformer

* Vision Transformer 를 활용한 Inpainting train

Vision Transformer 는 이미지를 패치 단위로 쪼개 토큰화를 시켜 각각의 패치에 linear embedding 을 적용 후 시퀀스로 만들어 둔 형태로 Transformer 입력으로 작동하게 된다.

우리는 이미지에 적용되고 있는 Vision Transformer 가 Deep Video Inpainting 논문에서 사용된 기술인 CNN 보다 능가하는 성능을 보임을 알 수 있었고 Image 가 아닌 Video 에서도 적용되는지 확인을 하고자 한다.

우리의 모델은 Deep Video Inpainting 에서 문제를 제기한 이미지와 비디오의 차이인 시간 차원의 확장 및 연계 부분을 시계열 데이터인 동영상 프레임 사이의 정보를 파악하기 위해 여러 개의 프레임을 패치 단위로 나누어 Vision Transformer 모델에 입력하고 그 값을 이미지 생성 모델인 GAN 을 활용하여 각 프레임별로 이미지를 복원해 다시 동영상을 생성하고자 한다.

• Architecture

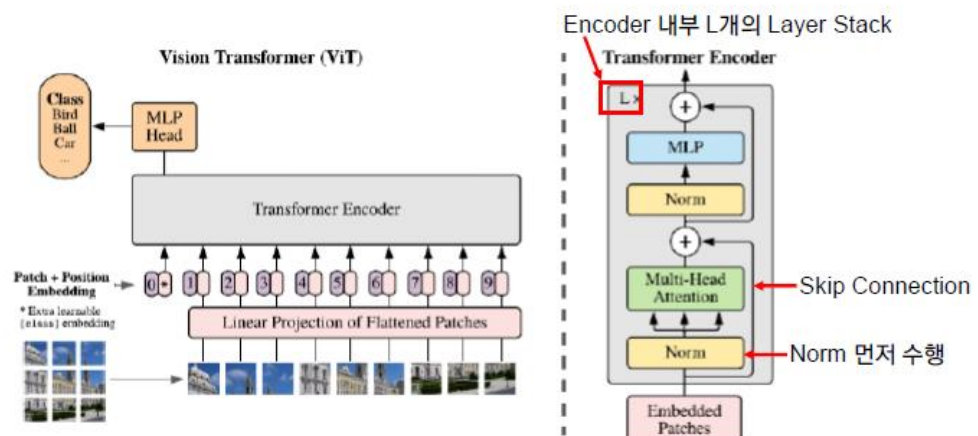


Fig.6 Vision Transformer Architecture [6]

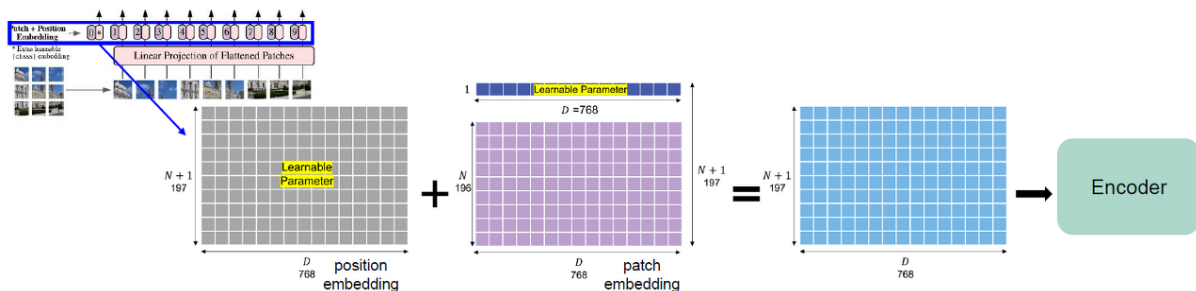


Fig.7 Embedding (Vision Transformer) [6]

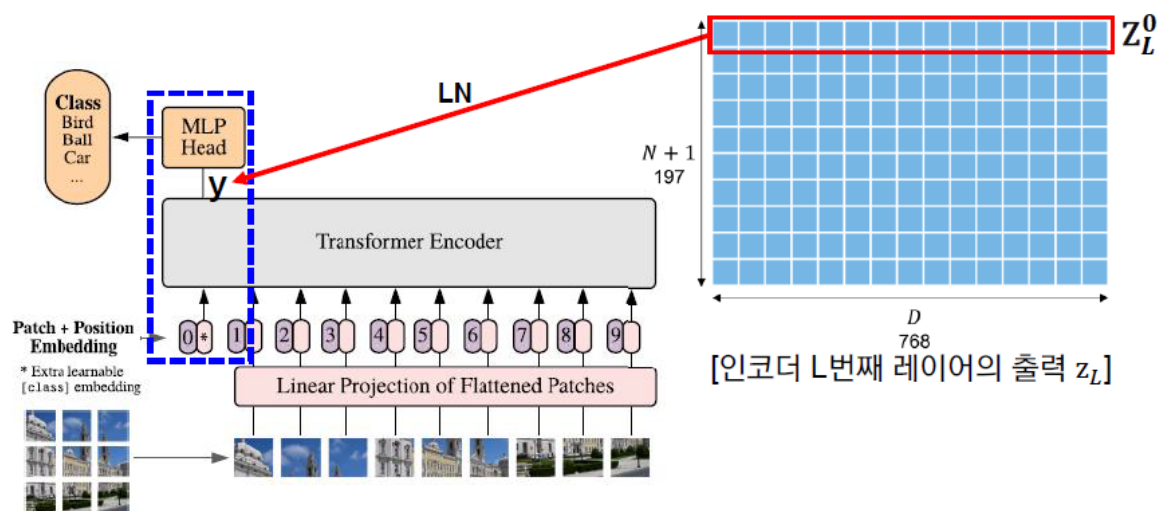


Fig.8 Classification Head[6]

그림[6-8]은 Vision Transformer 에 대한 자료들로 이해를 하는데 도움이 된다. 우리는 현재의 모델에서 시간의 차원을 늘려 이미지들 간의 연관성을 파악하는 모듈을 추가하고자 한다.