

Video Super Resolution

CSED451 POSTECH

Team 한국인

- 도승욱 20180331 컴퓨터공학과
- 최은수 20180050 컴퓨터공학과
- 권민재 20190084 컴퓨터공학과

Introduction

Background

최근 4K에 이어 8K 기술까지 고화질, 고해상도의 display기술이 집약적으로 발전하면서 그에 맞추어 고화질 video의 수요가 폭발적으로 증가하고 있다. 새로운 고해상도 video를 새롭게 제작하는 것도 좋지만, 과거에 제작된 저해상도의 영화, 비디오 등의 video 콘텐츠를 고해상도 device에서 세밀하고 촘촘한 pixel로 표현된 video로 시청하고자 하는 수요도 증가하고 있다. 이러한 문제를 해결하기 위하여 video의 해상도를 올리는 video upscaling기술도 발맞추어 발전하고 있다. 기존에는 interpolation algorithm을 사용하여 주변의 픽셀 값으로 모르는 데이터 값을 추정하는 방식을 사용했다. 하지만 이는 이미지의 크기만 조정해 주며 품질 자체는 개선시키지 못하기 때문에 Deep-learning을 사용한 upscaling 방법이 사용되고 있다. 그래서 우리는 이러한 시대적 수요에 맞추어, video upscaling을 프로젝트 주제로 선정하였다.

Goal

이 프로젝트의 목표는 저화질 video를 정수배 개선된 해상도를 가지는 고화질 video로 upscaling하는 프로그램을 제작하는 것이다. 단순한 interpolation 알고리즘에서부터 deep learning까지 upscaling을 하기 위한 다양한 방법이 제시되어 있는데, 이 프로젝트에서는 deep learning을 활용한 upscaling을 구현했다. CVPR에 발표된 논문 중 하나인 "Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit"의 방법을 차용하였으며, 일부 파라미터를 수정하여 학습 속도에 도움이 되도록 하였다. 또한, 단순히 네트워크를 구현하는 것에서 더 나아가서, 해당 네트워크를 이용한 서비스를 사람들이 직접 체험해볼 수 있도록 어플리케이션을 제작하는 것을 목표로 삼았다. 최근 가장 핫한 어플리케이션인 웹 페이지의 형태로 사용자에게 제공하고자 하였기 때문에, 네트워크의 구현 및 학습이 완료된 이후에 Python의 웹 프레임워크들 중 하나인 Flask를 이용해 간단한 web application을 제작하여 유저들이 웹사이트에 upscaling을 원하는 임의의 video를 업로드 하면 upscaling 후 download 받을 수 있도록 구현하는 것을 목표로 삼았다.

Design

Problem Analysis

Super Resolution은 그래픽스 분야의 한 갈래로, 저해상도의 미디어를 고해상도의 미디어로 변환하는 것을 말한다. 이 목적 자체가 도전적이기 때문에 많은 사람들이 도전하고 있는 동시에, 과거 해상도가 낮게 기록된 미디어나 화질 면에서 손상된 미디어를 복구해낼 수 있다는 점에서 흥미로운 주제로 여겨진다. 기존에는 Bicubic Interpolation과 같은 수학적 방법을 이용하여 픽셀 보간을 채우는 방식으로 Super Resolution의 목적을 달성하였으나, 최근에는

Deep Learning이 여러 분야에 도입됨에 따라 Super Resolution에서도 deep learning을 이용하는 추세이다. Super Resolution은 주로 이미지나 비디오에서 사용된다. 비디오는 연속된 이미지의 더미로 구성되며, 1초에 일정한 양의 이미지를 넘기는 방식으로 구현된다. 그렇기 때문에 비디오에 대한 Super Resolution을 진행하는 것은 이미지에 비해 더 어렵지만, 최근 video 기반 미디어 산업이 시시각각으로 발전함에 따라 사용처가 매우 늘어났기 때문에 이미지 해상도를 높이는 것 보다 더 활용성이 뛰어날 것으로 기대된다.

Research Directions

Problem Definition

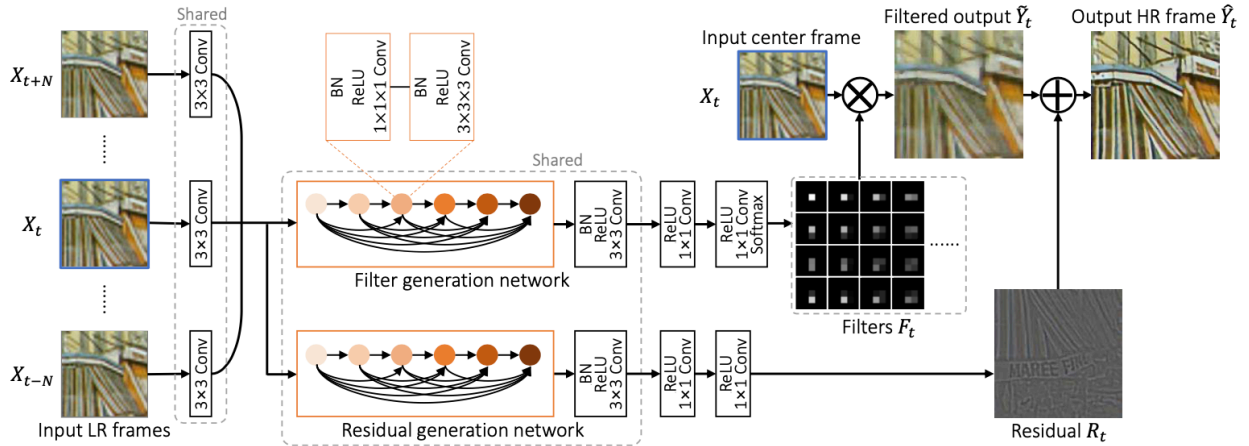
Video super resolution의 목표는 주어진 low resolution (LR) frame $\{X_t\}$ 로부터 high resolution (HR) frame $\{\hat{Y}_t\}$ 을 획득하는 것이다. VU network G 와 network parameter θ 로부터 다음과 같이 VU problem을 정의한다.

$$\hat{Y}_t = G_\theta(X_{t-N:t+N})$$

이 때, N 은 temporal radius를 의미한다. G 의 input tensor shape는 $T * H * W * C$ 이며, $T = 2N + 1$, H 와 W 는 각각 LR frame의 height와 weight를 의미한다. 그리고 C 는 color channel의 개수이다. output tensor shape는 $1 * rH * rW * C$ 이다. r 은 upscaling factor이다.

Network Architecture

\hat{Y}_t 를 생성하기 위해 network는 $\{X_{t-N:t+N}\}$ 로부터 dynamic upscaling filter F_t 와 residual R_t 를 생성한다. 그 후, input center frame X_t 는 먼저 F_t 로 upscaling 된 후, R_t 와 더해져 최종적으로 \hat{Y}_t 가 생성된다.

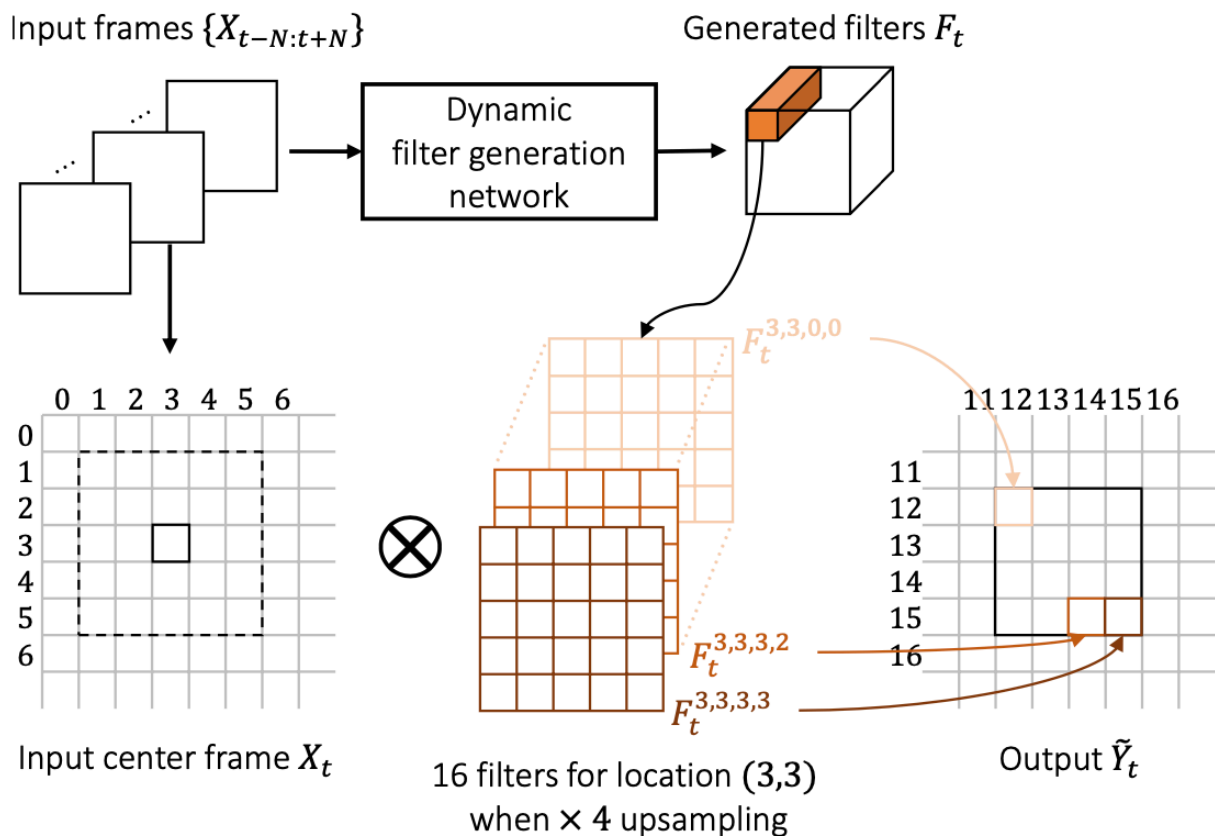


Dynamic Upsampling Filters

훈련된 network는 $\{X_{t-N:t+N}\}$ 을 input으로 받아 특정 사이즈의 filter들의 set F_t 를 output한다. (F_t 는 r^2HW 개의 filter로 구성) 그리고 이것은 filtered HR frame \tilde{Y}_t 를 생성하는데 사용된다. 각각의 HR pixel value는 input frame X_t 의 LR pixel에 local filtering을 filter $F_t^{(y,x,v,u)}$ 로 적용함으로써 얻을 수 있다.

$$\tilde{Y}_t(yr+vr, xr+u) = \sum_{j=-2}^{2} \sum_{i=-2}^{2} F_t^{(y,x,v,u)}(j+2, i+2) X_t(y+j, x+i)$$

여기서, x 와 y 는 LR grid의 좌표이며, v 와 u 는 $r \times r$ output block의 coordinate이다.

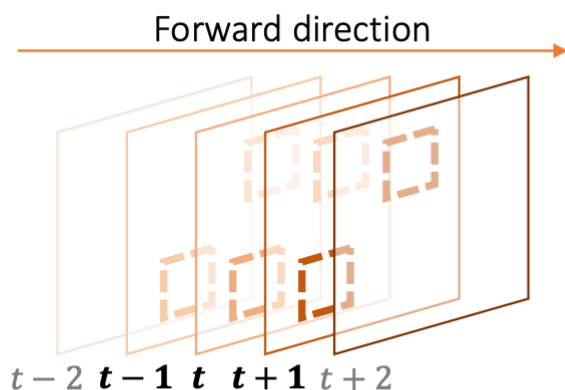


Residual Learning

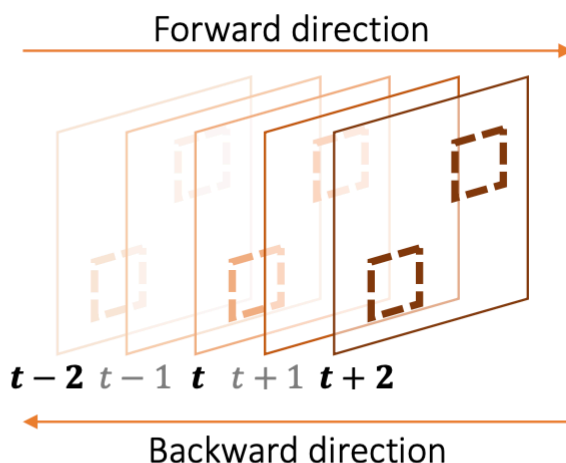
dynamic upsampling filter만을 적용시킨 결과는 sharpness에서 lack을 가진다. 그건 그저 input pixel의 weighted sum이기 때문이다. 이것을 해결하기 위해 residual image를 estimate하여 high frequency detail을 증가시킨다. Dynamic Upsampling process와 residual process를 결합함으로써 HR frame에서 spatial sharpness와 temporal consistency를 성취할 수 있다.

Temporal Augmentation

dynamic upsampling filter만을 적용시킨 결과는 sharpness에서 lack을 가진다. 그건 그저 input pixel의 weighted sum이기 때문이다. 이것을 해결하기 위해 residual image를 estimate하여 high frequency detail을 증가시킨다. Dynamic Upsampling process와 residual process를 결합함으로써 HR frame에서 spatial sharpness와 temporal consistency를 성취할 수 있다.



(a) $TA = 1$.



(b) $TA = \{-2, 2\}$.

Implementation

Dataset

Crawling

저작권 문제가 없는 video 수집을 위해 "Pixabay"에서 구할 수 있는 상업적 용도로 사용 가능한 동시에 출처를 밝히지 않아도 되는 비디오들 중 움직임이 dynamic한 video들로 수집하였다.

Python을 이용하여 해당 웹사이트의 video들을 crawling 했으며, "Walking", "Animal", "Exercise", "Transport" 등의 keyword로 dynamic한 motion이 포함되어 있고, 길이가 8~15초 이내인 video 들로만 crawling 하였다.

Convert to Frame

OpenCV를 사용하여 총 670개의 video에서 대략 320,000 개의 frame image를 추출하였다. Ground truth는 frame image 중 모션이 가장 잘 나타난다고 추정할 수 있는 중앙 부분의 128x128 크롭된 image를 사용하였고, LR input은 32x32 frame image를 사용하였다.

Network Architecture

Network

Tensorflow 2.0의 high level framework인 keras를 이용해 Network를 구성했다. Network를 이루는 주요 요소인 Convolutional와 Batch Normalization layer는 keras 제공 layer를 사용하였으며 local filtering layer는 사용자 정의 층으로 구현해 사용하였다.

Training

제작한 dataset과 network를 사용하여 training 을 진행하였다. 컴퓨터공학과 공용 클러스터 서버를 이용하여, 100 Epoch, Batch size 16, GPU 2개로 학습하였다. 100 epoch 이상에서 유의미한 학습이 진행되지 않았기 때문에, 학습은 100 epoch로 제한한 상태에서 huber loss의 delta나 cosine decay의 warmup 파라미터를 조금씩 변경하며 validation loss를 줄이는 방향으로 학습을 진행하였다.

Web Application

제작된 network를 실제 유저들이 프로덕션 레벨에서 사용해볼 수 있도록 웹 어플리케이션을 제작하였다. 기존 코드와 매끄럽게 통합되는 동시에 간단한 페이지를 제작하는 것을 목표로 삼았기 때문에, Python의 가벼운 웹 프레임워크인 Flask를 이용하여 어플리케이션을 제작하였고, docker를 이용한 컨테이너화를 통해 누구나 쉽게 배포할 수 있게 설정하였다.


Upload


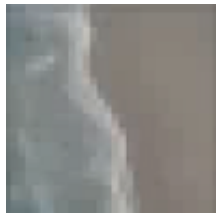


유저가 웹 페이지에 접속하면, 우선 파일을 업로드할 수 있는 창을 볼 수 있다. 유저가 파일을 업로드하면, 서버는 해당 파일을 각 유저의 세션 공간에 저장한다. 이것은 서로 다른 유저가 다른 유저의 파일을 열람할 수 없게 하기 위함이다. 해당 저장이 완료된 경우, 유저는 웹페이지에서 파일 목록에 자신이 업로드한 파일이 있는 것을 볼 수 있으며, 해당 파일 옆의 아이콘을 눌러서 변환을 요청할 수 있다.

Convert

유저가 변환을 요청하면, 해당 요청을 asynchronous 하게 처리하기 위해 우선 uWSGI thread를 이용하여 video 전처리와 upscaling을 background에서 진행하는 동시에, 유저에게는 우선 처리 중 이라는 사실을 알려준다. 한편, background에서는 유저가 업로드한 video를 우선 전처리한다. 이 프로젝트에서 구현한 network는 128x128의 크기를 가지는 이미지를 입력으로 받기 때문에, 유저가 업로드한 video를 프레임 별로 나눈 이후, 각 프레임을 다시 128x128 크기로 나눈다. 이후 각 부분을 network에 넣어서 그 결과를 얻은 이후에 각 결과를 merge하고, 또 그것을 다시 video로 merge하여 결과물을 만든다. 결과물이 만들어지면, 유저는 결과물이 만들어졌다는 사실을 웹 페이지의 아이콘을 통해 알 수 있으며, 해당 아이콘을 눌러서 다운로드 받을 수 있다.

Result

Input (256 x 256)	Output(1024 x 1024)
	

Input	Output
	
	

첫번째 표를 통해 해상도가 256x256인 video의 frame이 1024x1024 x16 upscaling되는 것을 확인할 수 있다. Input video와 output video의 frame 일부를 crop하여 비교해 보면 output에서 화질이 개선된 것을 가시적으로 확인할 수 있다. 학습이 완료되었을 때 최종 validation set에 대한 loss는 0.00435 이었으며, hyperparameter는 다음과 같다.

- Cosine Decay
 - Initial Learning Rate: 0.005

- First Decay Steps: 70
- Huber loss
- Δ : 200.0

Discussion

그 Motion Area identification

논문에서는 dataset을 제작할 때, video frame을 그대로 사용하지 않고 "motion이 충분히 보이는 area"를 crop하여 일부분만 사용하였다. 비슷하게 motion이 충분한 area를 crop하여 dataset을 제작하고자 하였으나 300개가 넘는 video를 일일이 확인하여 crop 위치를 결정하는 것은 무리가 있었다. 그래서 대체적으로 frame의 가운데 부분에서 object의 motion이 활발하게 보인다고 가정하고 중심부분을 crop하여 제작하였다. 그렇게 하다 보니 video에 따라서는 motion이 충분하지 않아 training에 도움이 되지 못하는 data들도 있었다. 논문에서는 300개의 video로 충분한 결과를 도출하였지만, 본 project에서는 dataset이 더 필요하게 되었고 추가적으로 300개의 video를 더 crawling해서 최종 dataset을 제작하였다. 이러한 motion area를 identify할 수 있는 방법에 대해 좀 더 고민해 본다면 적은 dataset으로도 더 좋은 성능을 낼 수 있을 것이다.

Speed

10초의 240p video를 4배로 upscaling하는데 약 6분의 시간이 소요된다. Video를 시청하는 대중소비자들에게 있어 직접 본 작업을 수행하도록 것은 매력적이지 않다. 따라서 현재의 기술로는 콘텐츠 공급자가 본인의 video를 upscaling 작업한 후 이를 대중에게 제공하도록 하는 것이 합리적인 business plan으로 보인다. video super resolution의 속도를 높이기 위해 다음의 방법을 논의하였다.

본 논문에서 video super resolution을 수행하기 위해 "Data augmentation"이 활용된다. 7개의 frame을 input하여 temporal consistency를 이용하는 방법인데, 이 경우 100 frame의 길이를 가진 video를 upscaling하기 위해 약 700개의 frame에 대한 computation이 필요하다. 계산되어야 할 총 프레임 수를 줄이기 위해, 100개의 frame 중 keyframe을 정하여 해당 프레임은 7개의 frame을 data augmentation을 활용한다. 나머지 frame들은 더 적은 수의 frame을 data augmentation을 활용하되, keyframe들의 interpolation으로부터 residual을 계산해 더해줌으로써 부족할 수 있는 data augmentation을 보충해준다. 이를 통해 계산량을 줄이고 이로 인해 발생할 품질 저하를 줄일 수 있기를 기대한다.

Category Specific Training

애니메이션, 자연 풍경, 건축물 등 여러 종류의 video가 존재하고 각 종류별 비디오가 나타내는 텍스처와 움직임은 상이하다. Category별로 비디오를 분류하여 dataset을 준비하여 category별로 training을 하였을 때, 성능이 향상하는지를 확인하고 싶다. 만약 유의미한 성능향상이 이루어진다면 사용자가 향상시키고자 하는 비디오의 종류에 맞추어 더 좋은 품질의 video super resolution을 제공할 수 있음을 기대한다.

참고문헌

- Jo, Y., Oh, S. W., Kang, J., & Kim, S. J. (2018). Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3224-3232).
- Jo, Y., Oh, S. W., Kang, J., & Kim, S. J. (2019). VSR-DUF. <https://github.com/yhjo09/VSR-DUF>

- Some code from the source is included in this project.