

# Few-Shot Adversarial Learning of Realistic Neural Talking Head Models

Zakharov, Egor, et al. ICCV 2019

한양대학교 AILAB 박소영

# INTRODUCTION

# 01. Introduction

---

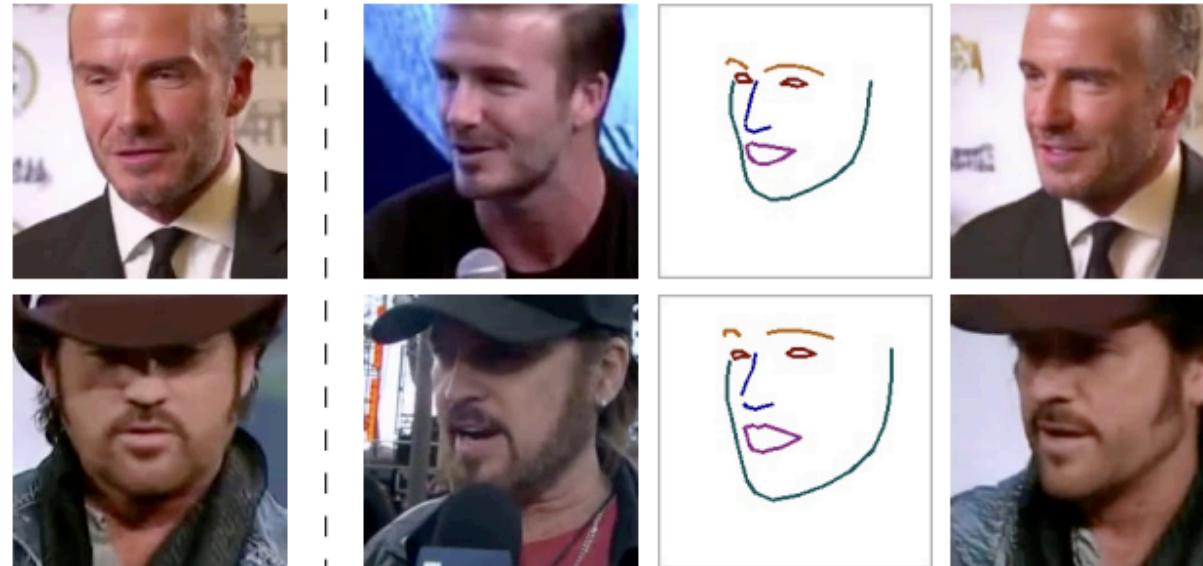
예시를 보자

<https://www.youtube.com/watch?v=p1b5aiTrGzY&t=163s>

00:04 ~ 00:20

# 01. Introduction

## 논문 소개



target 이미지의 landmark 정보로 source 이미지의 talking head를 실시간으로 생성함으로써  
source 이미지의 사람이 말하는 것처럼 보이게 하는 연구

# 01. Introduction

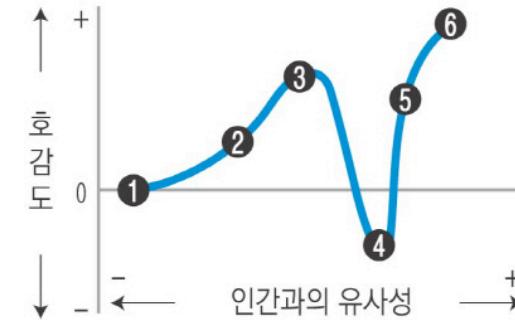
## Modeling Human Face

- 사람의 머리를 생성하는 것은 어려운 작업
  - 얼굴뿐만 아니라 머리카락, 입모양 등을 모두 모델링해야 하기 때문
- 불쾌한 골짜기(uncanny valley)를 피하기 위해 생성될 이미지에 **상당한 리얼리티가 요구됨**
  - 불쾌한 골짜기? : 로봇이 인간을 어설프게 닮을수록 오히려 불쾌함이 증가하는 것



로봇에 대한 사람들의 감정변화

자료: 일본 도쿄공업대 모리 마사히로 명예교수,  
미국 스탠퍼드대와 캘리포니아주립대 연구진



- |          |          |
|----------|----------|
| ① 산업용 로봇 | ④ 좀비     |
| ② 장난감 로봇 | ⑤ 인형     |
| ③ 봉제 인형  | ⑥ 건강한 사람 |

# 01. Introduction

## Modeling Human Face



그동안의 노력

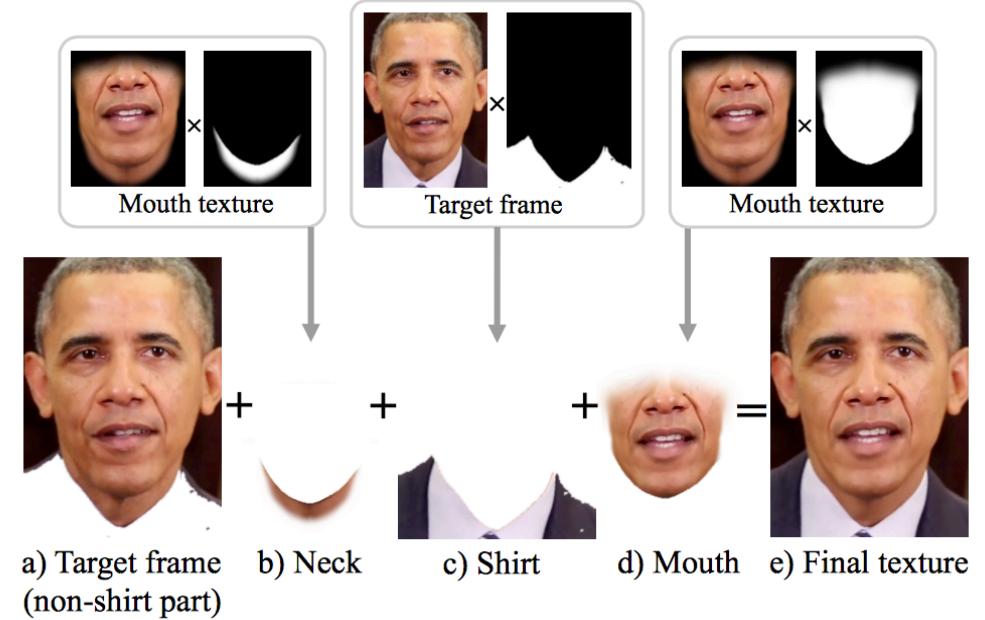
- **warping** : 움직임, 머리의 각도, 입모양 등을 표현하는 데 한계
- **GAN** : 생성하고자 하는 사람에 대한 **다량의 데이터와 많은 학습 시간을 필요**

원하는 사람에 대한 적은 수의 이미지를 빠르게 학습(few-shot learning)하여 해당 사람의 'Talking head'를 생성 가능  
meta learning 의 결과를 이용하면 새로운 사람의 사진 한장만으로도 실시간으로 talking head를 생성해 낼 수 있다.

# RELATED WORKS

## 02. Related works

### O-BAMA



**Synthesizing Obama** : 오리지널 비디오에 있는 말을 입술을 합성해서 입모양 랜드마크를 꺼내서 붙여주는 것

**ObamaNet** : TTS에서 나온걸 랜드마크로 뽑는 것

**Video to video synthesis** : 얼굴 외곽선 피쳐를 가지고 얼굴을 합성

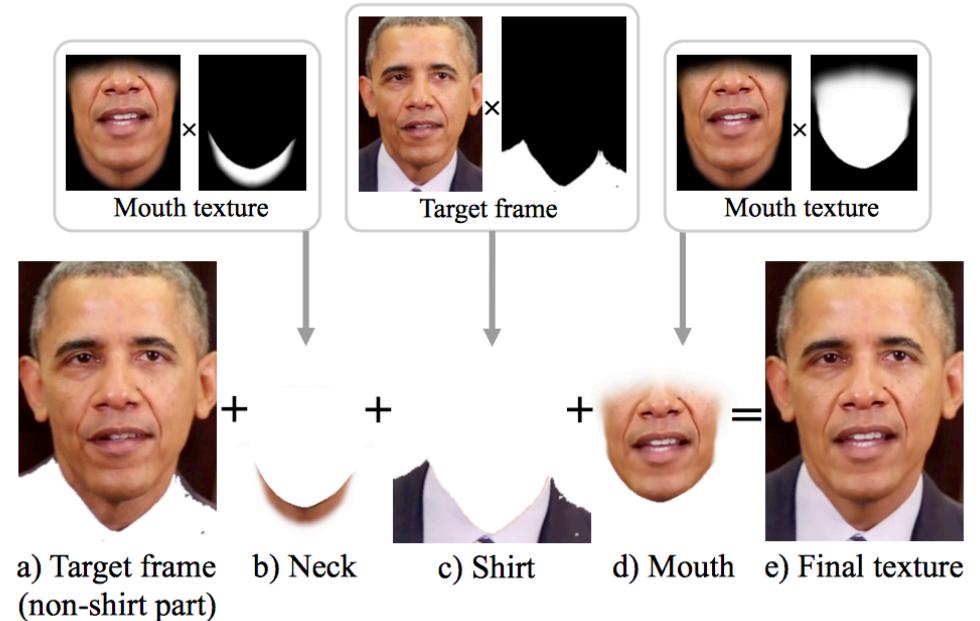
Suwajanakorn, Supasorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. "Synthesizing obama: learning lip sync from audio."

Kumar, Rithesh, et al. "Obamanet: Photo-realistic lip-sync from text."

Wang, Ting-Chun, et al. "Video-to-video synthesis."

## 02. Related works

### O-BAMA



데이터가 너무 많이 필요하다  
적은 데이터만으로 학습시킬 순 없을까?

Suwajanakorn, Supasorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. "Synthesizing obama: learning lip sync from audio."  
Kumar, Rithesh, et al. "Obamanet: Photo-realistic lip-sync from text."  
Wang, Ting-Chun, et al. "Video-to-video synthesis."

## 02. Related works

### Meta learning

#### 메타

위키백과, 우리 모두의 백과사전.

 다른 뜻에 대해서는 [메타 \(동음이의\)](#) 문서를 참조하십시오.

[[W]] 메타위키에 대해서는 [위키백과:메타](#) 문서를 참조하십시오.

메타(영어: meta-, 그리스어: μετά→ 뒤, 넘어서, 와 함께, 접하여, 스스로)는 영어의 접두사로, 다른 개념으로부터의 [추상화](#)를 가리키며 후자를 완성하거나 추가하는 데에 쓰인다.

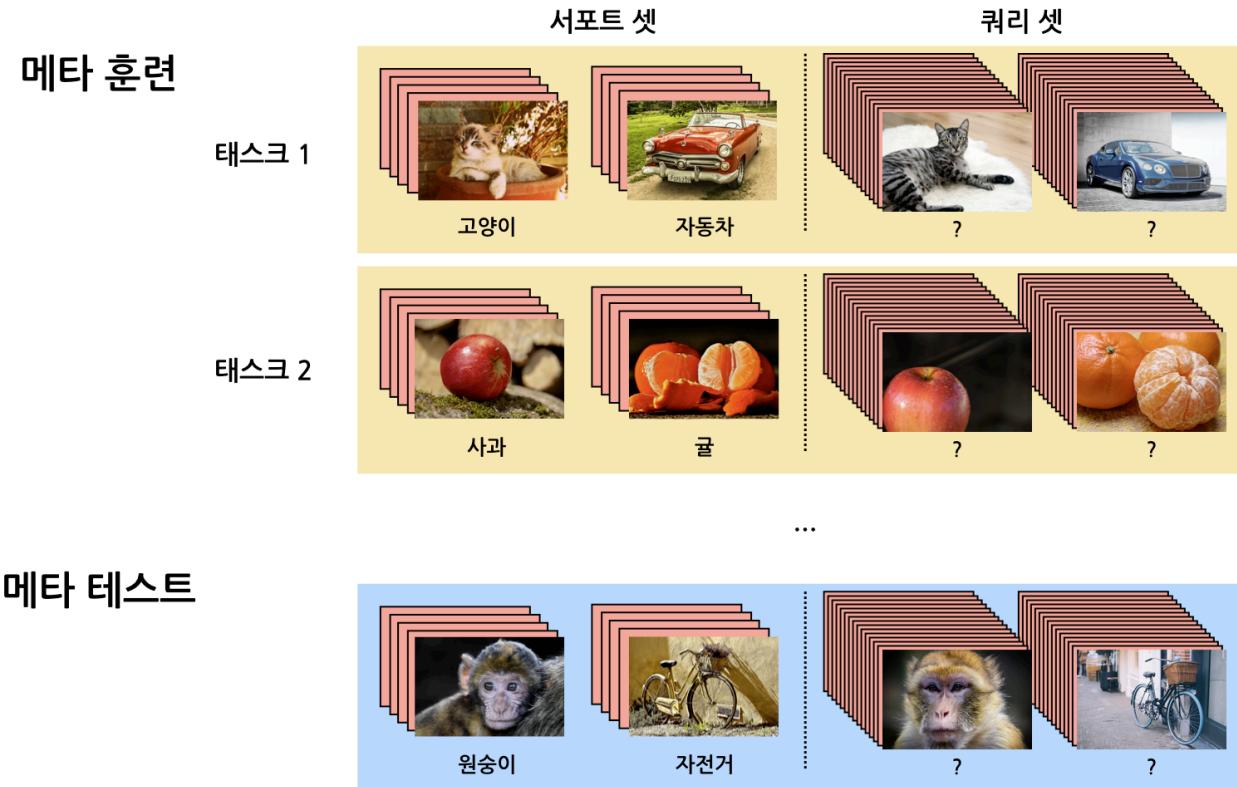
#### Meta learning?

학습하는 방법을 학습하는 것

사람이 통제하던 것을 자동화함으로써 스스로 학습 규칙을 익힐 수 있게 하는 것

## 02. Related works

### Meta learning



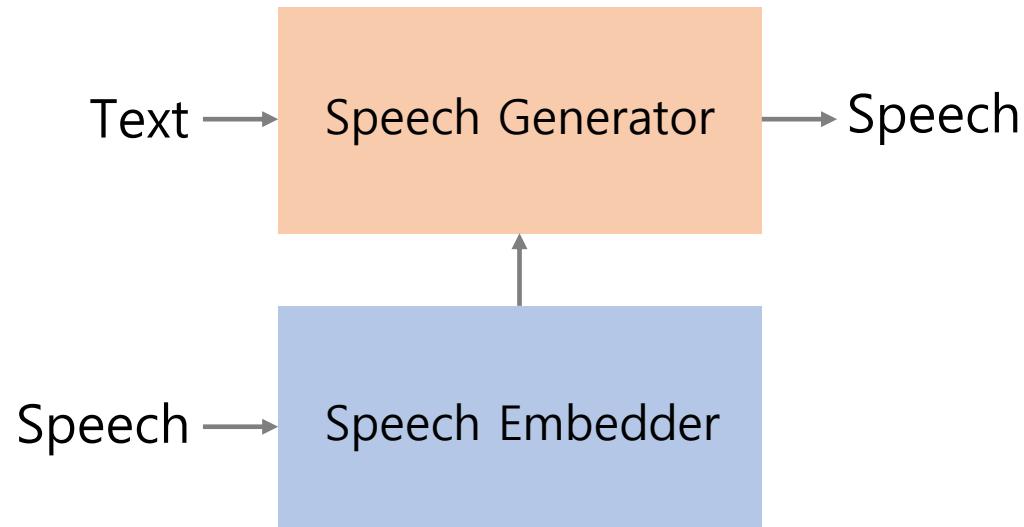
# METHODS

# 03. METHODS

## 다른 분야, 흡사한 Approach

### Speech 분야

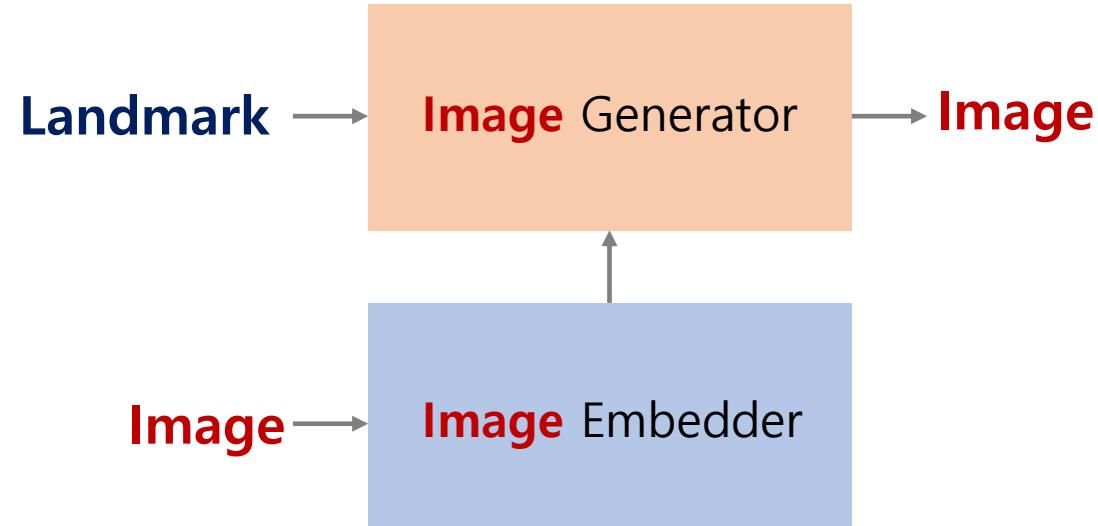
- 원래 텍스트를 가지고 스피치를 뽑음
- 기존엔 데이터가 굉장히 많이 들기 때문에 Speech를 인풋으로 주고 이걸 특정 화자에 대해 임베딩
- 특정 Speech feature을 갖는 Speech Embedding vector를 주면 그 feature와 비슷하게 Speech를 얻을 수 있다



# 03. METHODS

## 다른 분야, 흡사한 Approach

- Landmark feature를 가지고 Face Image를 생성하는 network
- 비슷한 Image를 input으로 주고 Embedding을 해서 학습
  - 비슷한 image의 feature만 가지고 input image feature와 비슷한 Image를 만들 수 있다



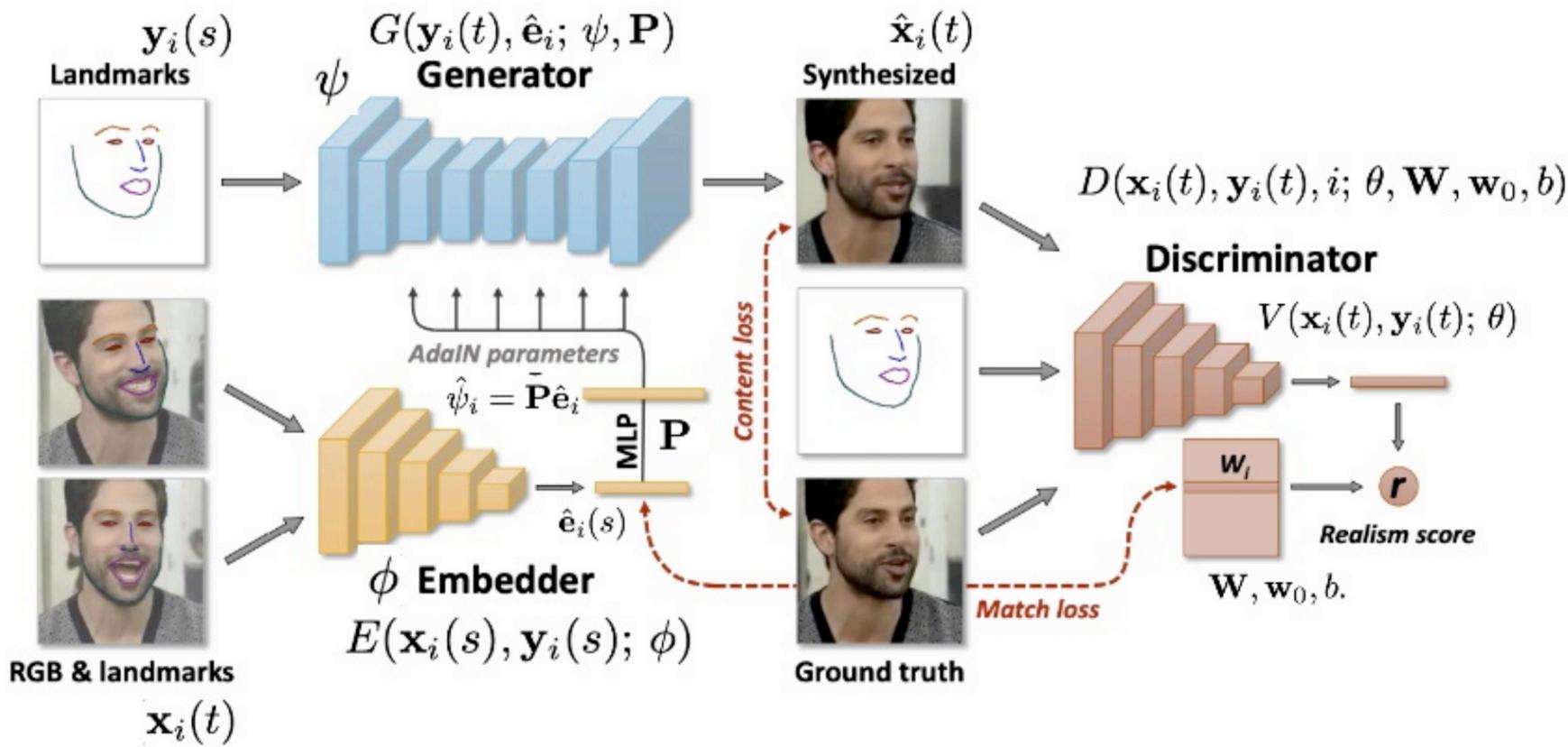
# 03. METHODS

## 3.1. 전체적인 Architecture

$x_i$ :  $i$  번째 비디오

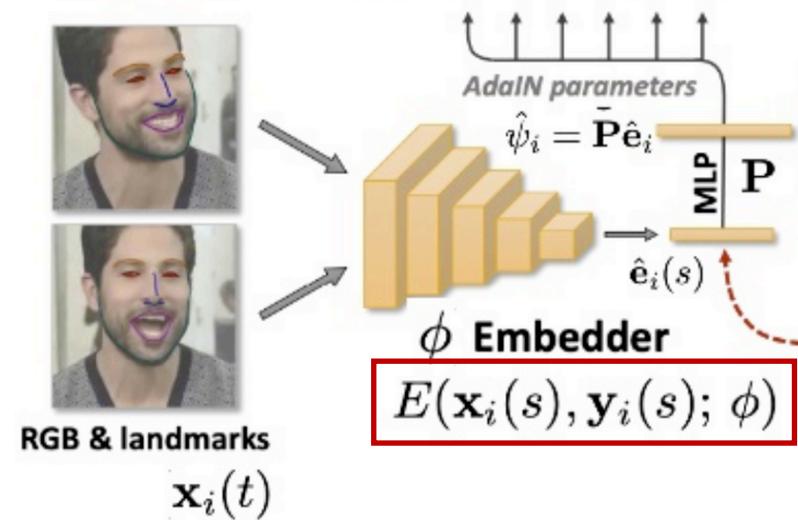
$x_i(t)$ :  $t$  번째 비디오의  $t$  번째 프레임

$y(t)$ :  $x_i(t)$ 로부터 추출한 이미지화된 얼굴 랜드마크



# 03. METHODS

## 3.1. Architecture - Embedder



비디오 프레임  $x_i(s)$ 와 얼굴 랜드마크 이미지  $y_i(s)$  를 N차원 벡터  $\hat{e}_i(s)$  로 임베딩

$\phi$  : Meta learning 단계에서 embedder의 파라미터

-  $\phi$  는  $\hat{e}_i(s)$  가 포즈에 강건하고 프레임  $s$  의 비디오 정보(person's identity)를 학습

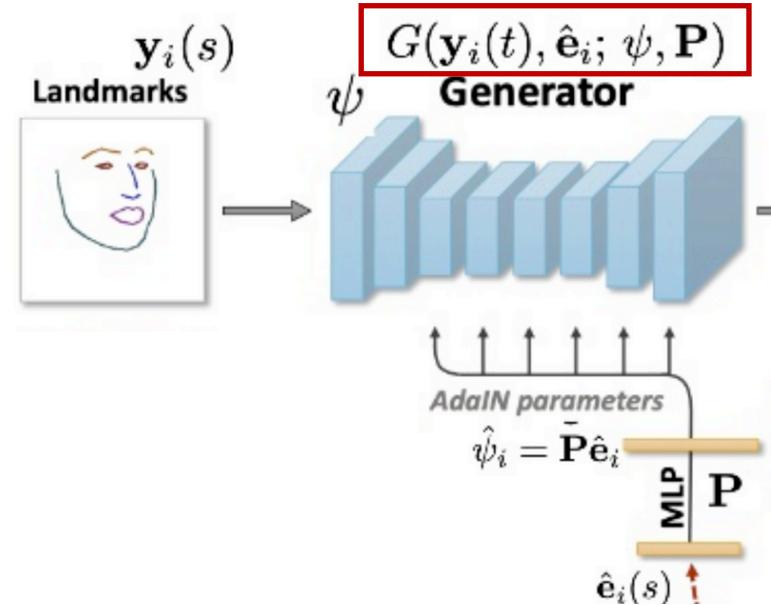
    -  $\hat{e}_i(s)$  : embedder에 의해 계산된 벡터

- frame들의 평균을 내므로 샷들을 대표하는 vector representation을 학습

- MLP : Multiple FCNs

# 03. METHODS

## 3.1. Architecture – Generator

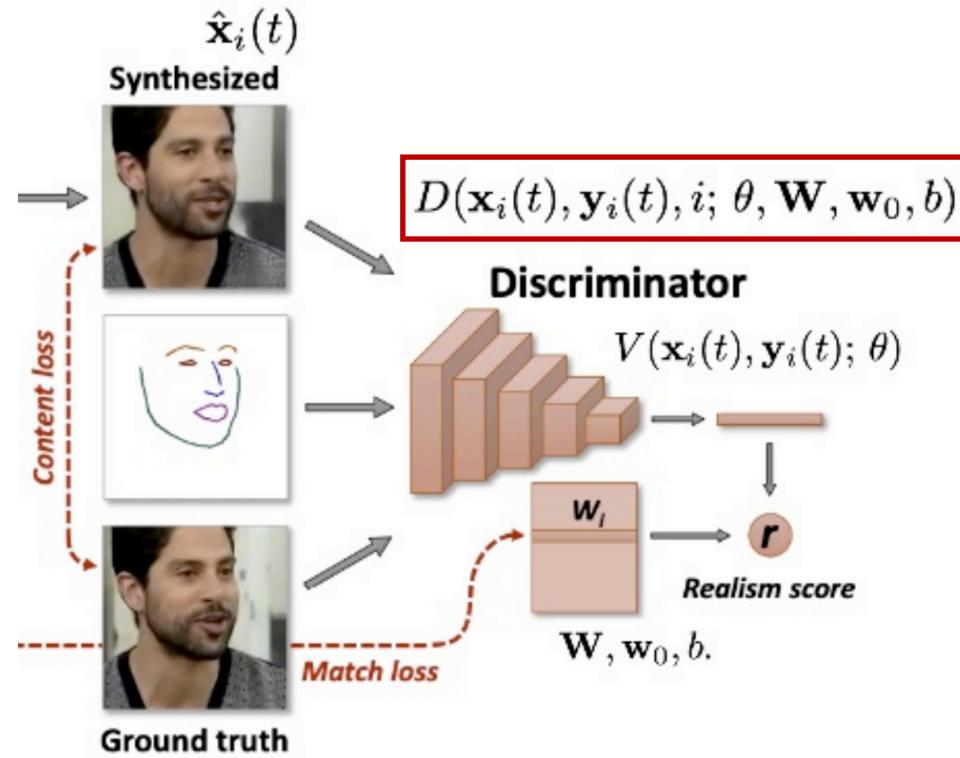


embedder가 보지 못한 랜드마크  $y_i(t)$ 와 임베딩 된 벡터  $\hat{e}_i(s)$ 를 입력으로 받아 비디오 프레임  $x_i(t)$ 를 생성

- 생성된 이미지와 ground truth의 유사도가 최대로 되도록 학습
- generator의 모든 파라미터는 두 부분으로 구분된다.
  - $\psi$  : person-generic parameter (global)
  - $\hat{\psi}_i$  : person-specific parameter
- 메타러닝 때 /  $\psi$ 는 직접 학습되고,  $\hat{\psi}_i$ 는 projection 행렬  $P$ 와 embedding vector  $\hat{e}_i(s)$ 로부터 계산된다 ( $\hat{\psi}_i = P\hat{e}_i(s)$ )

# 03. METHODS

## 3.1. Architecture – Discriminator



비디오 프레임  $x_i(t)$  와 이에 대한 랜드마크 이미지  $y_i(t)$  , 그리고 비디오 인덱스  $i$ 를 입력으로 받는다.

- Input frame과 landmark image를 N차원 벡터로 매핑하는 ConvNet  $V(x_i(t), y_i(t); \theta)$ 를 포함
- $(\theta, W, w_0, b)$ 는 모두 discriminator의 학습 파라미터)

최종적으로 D는 (a)입력 프레임이 실제 이미지인지, (b)랜드마크에 부합하는지를 나타내는 스코어  $r$ 을 출력

# 03. METHODS

## 3.2 Meta-learning stage

- 메타러닝 단계의 각 에피소드에서, 하나의 비디오  $i$ 와 해당 비디오의 프레임  $t$ 를 임의로 선택한다.
- $t$  외에 추가로  $K$ 개의 프레임을 추출한다( $s_1, s_2, \dots, s_k$ )
  - K-shot learning: 만약  $k$ 가 8이라면 8개 프레임을 가지고 학습.
  - 랜덤하게 샘플링한다
- $i$  번째 비디오에 대한  $\hat{e}_i$  는 단순히  $k$ 개의 embedding vector  $\hat{e}_i(s_k)$  의 평균을 구해 얻는다.

$$\hat{\mathbf{e}}_i = \frac{1}{K} \sum_{k=1}^K E(\mathbf{x}_i(s_k), \mathbf{y}_i(s_k); \phi) . \quad (1)$$

- 생성되는 이미지  $x_i(t)$  는 다음과 같이 generator에 의해 생성된다

$$\hat{\mathbf{x}}_i(t) = G(\mathbf{y}_i(t), \hat{\mathbf{e}}_i; \psi, \mathbf{P}) . \quad (2)$$

- embedder와 generator의 파라미터는 아래의 로스 함수를 최소화하도록 학습된다. (파라미터 최적화)

$$\begin{aligned} \mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = & \boxed{\mathcal{L}_{\text{CNT}}(\phi, \psi, \mathbf{P})} + \\ & \boxed{\mathcal{L}_{\text{ADV}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b)} + \boxed{\mathcal{L}_{\text{MCH}}(\phi, \mathbf{W})} . \end{aligned} \quad (3)$$

# 03. METHODS

## 3.2 Meta-learning stage – Loss functions

$$\begin{aligned}\mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = & \boxed{\mathcal{L}_{\text{CNT}}(\phi, \psi, \mathbf{P})} + \\ & \boxed{\mathcal{L}_{\text{ADV}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b)} + \boxed{\mathcal{L}_{\text{MCH}}(\phi, \mathbf{W})}.\end{aligned}\quad (3)$$

Generator에서의 Loss function

$$\begin{aligned}\mathcal{L}_{\text{DSC}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = & \\ & \max(0, 1 + D(\hat{\mathbf{x}}_i(t), \mathbf{y}_i(t), i; \phi, \psi, \theta, \mathbf{W}, \mathbf{w}_0, b)) + \\ & \max(0, 1 - D(\mathbf{x}_i(t), \mathbf{y}_i(t), i; \theta, \mathbf{W}, \mathbf{w}_0, b)).\end{aligned}\quad (6)$$

Discriminator에서의 Loss function

## 03. METHODS

### 3.2 Meta-learning stage – Generator Loss function $L_{CNT}$

$$\begin{aligned}\mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) &= \boxed{\mathcal{L}_{CNT}(\phi, \psi, \mathbf{P})} + \\ &\quad \mathcal{L}_{ADV}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) + \mathcal{L}_{MCH}(\phi, \mathbf{W}).\end{aligned}\tag{3}$$

- content loss term.
- GT 이미지  $x_i(t)$  와 생성된(재구성된) 이미지  $\hat{x}_i(t)$ 사이의 거리를 측정한다.
  - VGG19, VGGFace에서 나온 perceptual loss를 사용

# 03. METHODS

## 3.2 Meta-learning stage – Generator Loss function $L_{ADV}$

$$\begin{aligned}\mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) &= \mathcal{L}_{CNT}(\phi, \psi, \mathbf{P}) + \\ &\quad \boxed{\mathcal{L}_{ADV}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b)} + \mathcal{L}_{MCH}(\phi, \mathbf{W}).\end{aligned}\tag{3}$$

- adversarial term
- discriminator에서 나온게 얼마나 realistic한지 계산한다

$$\begin{aligned}\mathcal{L}_{ADV}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) &= \\ &- D(\hat{\mathbf{x}}_i(t), \mathbf{y}_i(t), i; \theta, \mathbf{W}, \mathbf{w}_0, b) + \mathcal{L}_{FM}.\end{aligned}\tag{4}$$

- 첫번째 항: discriminator에 의해 계산
  - 실제와 얼마나 같은지를 나타내는 스코어, 높을수록 실제와 가까운 이미지라는 뜻.
  - projection discriminator에 따라, W의 i번째 칼럼  $W_i$ 은 번째 동영상의 embedding을 나타냄.
  - discriminator는 입력을 먼저 N차원 벡터  $V(x_i(t), y_i(t); \theta)$ 로 임베딩하고 realism score를 계산
  - $w_0$  와  $b$  는  $x_i(t)$ 의 일반적인 realism과, 랜드마크 이미지  $y_i(t)$  와의 일치 정도를 파악하는 데 사용
- 두번째 항: 각 Discriminator output에 대해 perceptual similarity를 측정

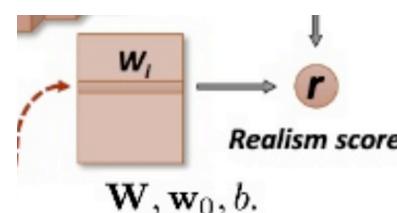
# 03. METHODS

## 3.2 Meta-learning stage – Generator Loss function $L_{MCH}$

$$\begin{aligned}\mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) &= \mathcal{L}_{\text{CNT}}(\phi, \psi, \mathbf{P}) + \\ &\quad \mathcal{L}_{\text{ADV}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) + \boxed{\mathcal{L}_{\text{MCH}}(\phi, \mathbf{W})}.\end{aligned}\quad (3)$$

- embedding matching term
- 지금까지의 내용에 의해, 본 논문의 시스템은 다음과 같이 두 종류의 embedding을 갖는다.
  - embedder에 의해 계산된 것
  - discriminator의 파라미터 중 하나인  $\mathbf{W}$ 의 각 칼럼
- 이 두가지가 얼마나 비슷한지 ( $L1$  dif)
- $\mathcal{L}_{\text{MCH}}(\phi, \mathbf{W})$ 은  $\hat{e}_i$ 와  $W_i$  사이의  $L1$ 거리에 패널티를 줌으로써 두 embedding이 비슷해지도록 한다.
  - $\mathbf{W}$ 는 비디오마다  $r$ 개가 있는데 각각 비디오의 대표 임베딩을 학습
  - 임베딩이 들어왔을 때 inner product를 하면 가장 비슷한 값이 아웃풋으로 나올 것

$W_i$  : 각 비디오들의 대표 임베딩을 갖고 있는 row



# 03. METHODS

## 3.2 Meta-learning stage – Loss functions

$$\begin{aligned}\mathcal{L}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = & \boxed{\mathcal{L}_{\text{CNT}}(\phi, \psi, \mathbf{P})} + \\ & \boxed{\mathcal{L}_{\text{ADV}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b)} + \boxed{\mathcal{L}_{\text{MCH}}(\phi, \mathbf{W})}.\end{aligned}\quad (3)$$

Generator에서의 Loss function

$$\begin{aligned}\mathcal{L}_{\text{DSC}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = & \\ & \max(0, 1 + D(\hat{\mathbf{x}}_i(t), \mathbf{y}_i(t), i; \phi, \psi, \theta, \mathbf{W}, \mathbf{w}_0, b)) + \\ & \max(0, 1 - D(\mathbf{x}_i(t), \mathbf{y}_i(t), i; \theta, \mathbf{W}, \mathbf{w}_0, b)).\end{aligned}\quad (6)$$

Discriminator에서의 Loss function

# 03. METHODS

## 3.2 Meta-learning stage – Discriminator Loss functions

$$\mathcal{L}_{\text{DSC}}(\phi, \psi, \mathbf{P}, \theta, \mathbf{W}, \mathbf{w}_0, b) = \quad (6)$$

$$\begin{aligned} & \text{(a)} \max(0, 1 + D(\hat{\mathbf{x}}_i(t), \mathbf{y}_i(t), i; \phi, \psi, \theta, \mathbf{W}, \mathbf{w}_0, b)) + \\ & \text{(b)} \max(0, 1 - D(\mathbf{x}_i(t), \mathbf{y}_i(t), i; \theta, \mathbf{W}, \mathbf{w}_0, b)). \end{aligned}$$

- 위 목적 함수는 가짜 이미지와 진짜 이미지의 realism를 비교
- discriminator의 파라미터가 각각의 경우에 realism score가 -1 밑으로, +1 위가 되도록 하는 방향으로 업데이트
  - (a) 가짜 이미지가 들어갔을 때 : -1보다 작아야 전체가 0이된다
  - (b) 진짜 이미지가 들어갔을 때 : 1보다 커져야 전체가 1이 된다
- 실제 이미지가 높은 r score를 갖고 가짜는 낮은 r score를 갖도록 hinge loss 최소화
- Embedder와 Generator 파라미터 업데이트 후 Discriminator가 업데이트 되면서 학습

# 03. METHODS

## 3.3 Few-shot learning by fine-tuning : Generator

$$\hat{\mathbf{e}}_{\text{NEW}} = \frac{1}{T} \sum_{t=1}^T E(\mathbf{x}(t), \mathbf{y}(t); \phi), \quad (7)$$

- 메타러닝이 끝나면 이제 새로운 사람의 talking head를 만들어 낼 수 있음
  - 새로운 사람의 embedding을 생성할 때에는 **메타러닝 단계에서 학습이 끝난 embedder**를 사용
    - 새로운 화자가 들어오면 그 화자에 대해서 임베딩벡터를 구하게 된다
    - 기존 generator 의 파라미터인  $\psi$ 와  $P$  를 그대로 사용하여 원하는 사람의 얼굴과는 다른 얼굴이 생성: identity gap
- > **fine-tuning 단계에서 해결 가능**

- generator  $G(y(t), \hat{\mathbf{e}}_{\text{NEW}}; \psi, P)$  대신  $G'(y(t); \psi, \psi')$  를 사용
  - $\psi$  : person generic parameters (global한 값)
  - $\psi'$  : person specific parameters (특정 사람에 대한 값)
- $\psi'$  는 메타러닝 때와 달리 직접 학습되는 파라미터 (initialize  $\psi' = P\hat{\mathbf{e}}_{\text{NEW}}$   $P$ 를  $e$ 에 프로젝션한 것)
  - $P$ (projection matrix) : 임베딩 벡터를 projection하는 데에만 사용
- 따라서 메타러닝의 prior로 새로운 사람에 대한 학습을 수행

## 03. METHODS

### 3.3 Few-shot learning by fine-tuning : Discriminator

$$D'(\hat{\mathbf{x}}(t), \mathbf{y}(t); \theta, \mathbf{w}', b) = V(\hat{\mathbf{x}}(t), \mathbf{y}(t); \theta)^T \mathbf{w}' + b. \quad (8)$$

- discriminator는  $D'(x(t), y(t); \theta, w', b)$ 를 통해 realism score를 계산 ( $w'$  : 새로운 화자)
- ConvNet part  $V(x(t), y(t); \theta)$ 와 bias  $b$ 는 메타러닝의 결과
- realism score는 메타러닝 단계와 비슷하게 입력을 N차원 벡터로 임베딩한 후 계산

## 03. METHODS

### 3.3 Few-shot learning by fine-tuning : Loss functions

generator의 파라미터  $\psi$  와  $\psi'$  는 아래 loss를 최소화하도록 학습

$$\begin{aligned}\mathcal{L}'(\psi, \psi', \theta, \mathbf{w}', b) = \\ \mathcal{L}'_{\text{CNT}}(\psi, \psi') + \mathcal{L}'_{\text{ADV}}(\psi, \psi', \theta, \mathbf{w}', b),\end{aligned}\tag{9}$$

discriminator의 파라미터는 아래 loss를 최소화하도록 학습

$$\begin{aligned}\mathcal{L}'_{\text{DSC}}(\psi, \psi', \theta, \mathbf{w}', b) = \\ \max(0, 1 + D(\hat{\mathbf{x}}(t), \mathbf{y}(t); \psi, \psi', \theta, \mathbf{w}', b)) + \\ \max(0, 1 - D(\mathbf{x}(t), \mathbf{y}(t); \theta, \mathbf{w}', b)).\end{aligned}\tag{10}$$

# 03. METHODS

---

## 3-4. Implementation details

- downsampling, upsampling layer를 **batch normalization** 대신 **instance normalization**을 사용
- embedder와 discriminator의 ConvNet 부분은 모두 **residual downsampling block** 구조
- 모든 네트워크의 각 레이어에 **spectral normalization** 과 **self-attention blocks**을 사용
- learning rate
  - embedder, generator :  $5 * 10^{-5}$
  - discriminator :  $2 * 10^{-4}$

# EXPERIMENTS

# 04. EXPERIMENTS

---

## 사용한 데이터셋

### - VoxCeleb1

- baseline study나 ablation study를 위해 사용
- 대부분의 연구들이 이 데이터셋을 위주로 이용

### - VoxCeleb2

- VoxCeleb1에 비해 약 10배가 넘는 비디오를 갖고 있다

# 04. EXPERIMENTS

## 실험 결과

Method (T)	FID↓	SSIM↑	CSIM↑	USER↓
VoxCeleb1				
X2Face (1)	45.8	<b>0.68</b>	<b>0.16</b>	0.82
Pix2pixHD (1)	<b>42.7</b>	0.56	0.09	0.82
Ours (1)	43.0	0.67	0.15	<b>0.62</b>
X2Face (8)	51.5	<b>0.73</b>	<b>0.17</b>	0.83
Pix2pixHD (8)	<b>35.1</b>	0.64	0.12	0.79
Ours (8)	38.0	0.71	<b>0.17</b>	<b>0.62</b>
X2Face (32)	56.5	<b>0.75</b>	0.18	0.85
Pix2pixHD (32)	<b>24.0</b>	0.70	0.16	0.71
Ours (32)	29.5	0.74	<b>0.19</b>	<b>0.61</b>
VoxCeleb2				
Ours-FF (1)	<b>46.1</b>	0.61	<b>0.42</b>	<b>0.43</b>
Ours-FT (1)	48.5	<b>0.64</b>	0.35	0.46
Ours-FF (8)	42.2	0.64	<b>0.47</b>	0.40
Ours-FT (8)	<b>42.2</b>	<b>0.68</b>	0.42	<b>0.39</b>
Ours-FF (32)	40.4	0.65	<b>0.48</b>	0.38
Ours-FT (32)	<b>30.6</b>	<b>0.72</b>	0.45	<b>0.33</b>

### FID : Frechet-inception distance

- 주로 인지적 사실성(perceptual realism)을 측정하는 metric

### SSIM : Structured similarity

- ground truth 이미지와의 low-level 유사도를 측정

### CSIM : Cosine similarity

- identity mismatch 정도를 측정하기 위해 SOTA 얼굴인식기로 생성된 embedding vector 사이의 cosine similarity를 측정

### USER : User study

- 사람들에게 물어보는 것. 3개를 주면서 2개는 진짜, 1개는 가짜

FF : 임베딩 제외 파인튜닝

FT : 전부 파인튜닝

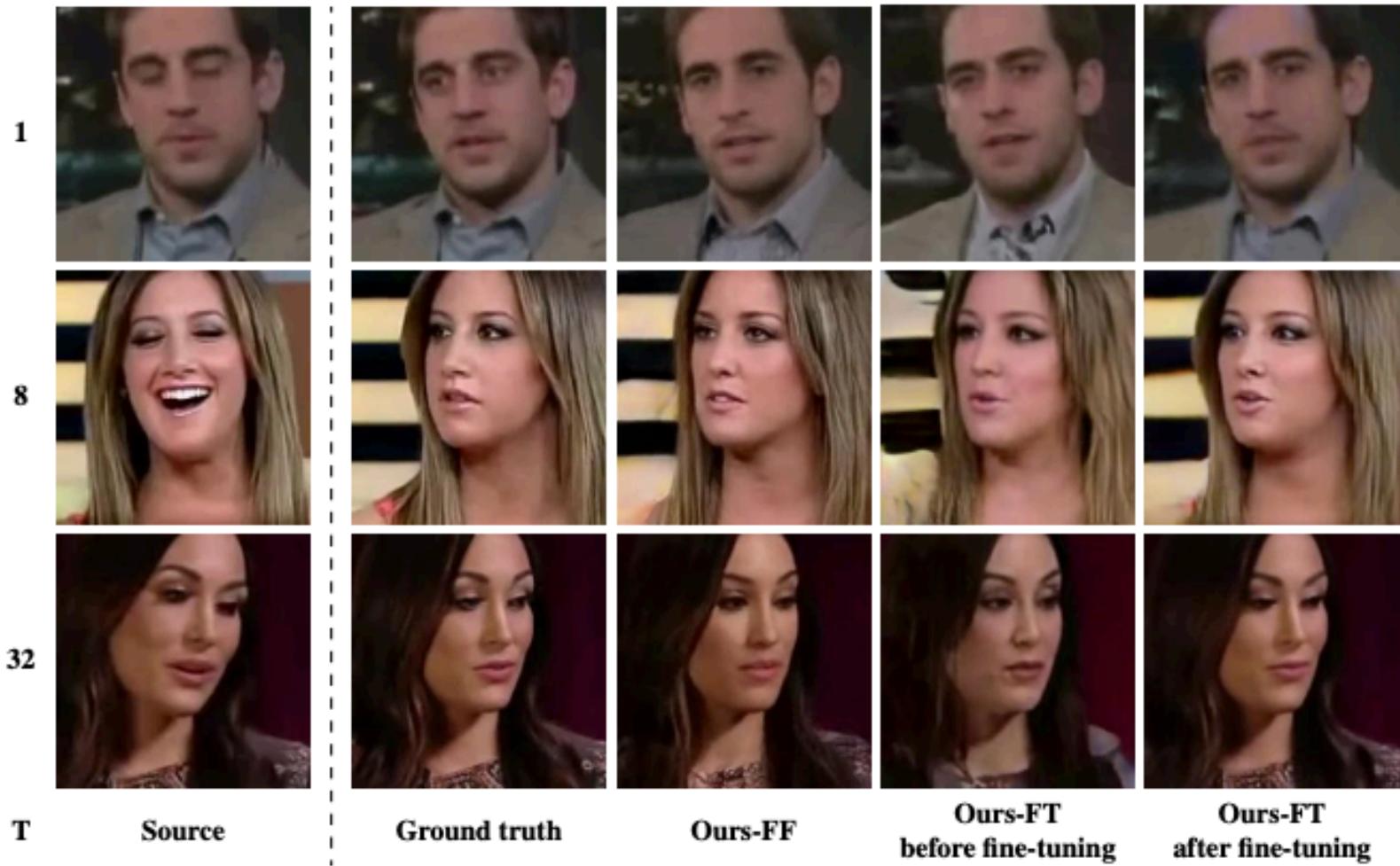
## 04. EXPERIMENTS

다른 모델과의 비교



# 04. EXPERIMENTS

Fine tuning 비교



# 04. EXPERIMENTS

## puppeteering results



source의 랜드마크가 아닌 다른 사람의 랜드마크를 사용하여 talking head를 생성

# CONCLUSION

# 05. CONCLUSION

## 결론 및 한계

### 결론

- 실제와 매우 비슷한 'talking head'를 생성하는 GAN의 메타러닝을 위한 프레임워크를 제시
- 아주 작은 샘플만 가지고 잘 만들 수 있다 (32개면 더 좋고)

### 한계

- 표현의 한계
  - gaze에서 사람 시선에 대한 랜드마크가 없어서 이상하게 나온다던가...
- 랜드마크 자체를 adaptation을 하진 않기 때문에 완전 다른 사람을 집어넣으면 이상하게 나올 수 있음
  - 같은 사람에 대한 랜드마크를 넣어야지 자연스럽다
  - 랜드마크를 adaptation할 수 있으면 딴사람이어도 충분히 할 수 있지 않을까..

