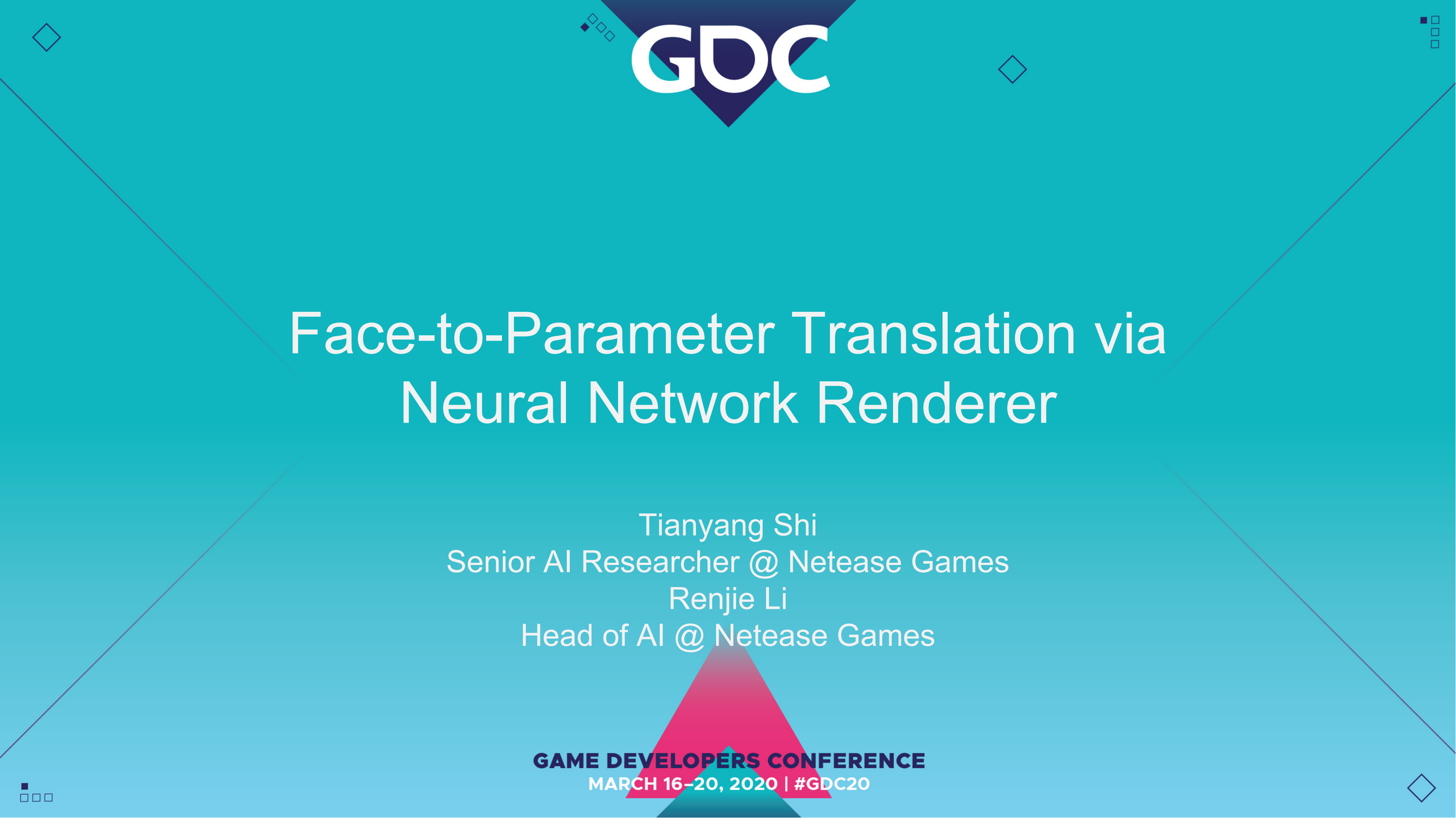
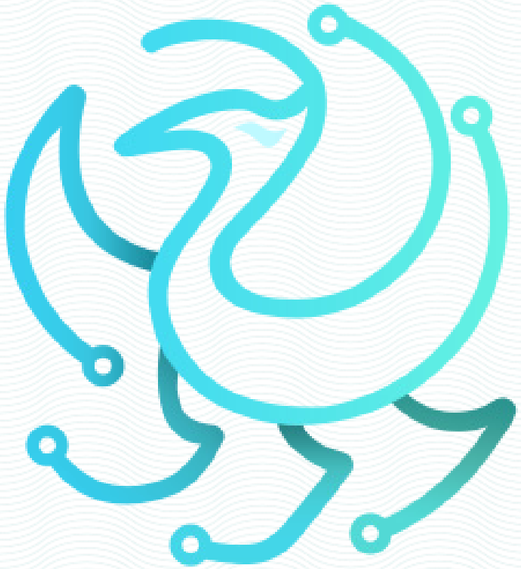


Face-to-Parameter Translation via Neural Network Renderer

Tianyang Shi
Senior AI Researcher @ Netease Games
Renjie Li
Head of AI @ Netease Games

GAME DEVELOPERS CONFERENCE
MARCH 16–20, 2020 | #GDC20

Fuxi AI Lab



FUXI-Lab

NetEase Games

+

- Established in Sep. 2017
- First Game AI Research Lab in China
- Vision: Enlighten Games with AI

Fuxi AI Lab

NetEase Games Passion of gamers | FUXI-Lab

Reinforcement Learning

User Profiling

Computer Vision & Graphics

Virtual Human

Natural Language Processing

Big Data & Computing Platform

Talks from Fuxi AI Lab

Talk
The AI knows you better than yourself: mining player pursuits in Justice Online – AI Summit
Full body animation generation for expressive NPCs – Machine Learning Summit
Building an intelligent game testing system in Netease MMORPG games – Machine Learning Summit
Face-to-parameter translation via neural network renderer - Core
Applying reinforcement learning to develop game AI in Netease games - Core

Overview

- Introduction
- Face-to-Parameter Translation
- Discussion

Introduction

Game Character Customization

- This system allows players to edit the profiles of their in-game characters.

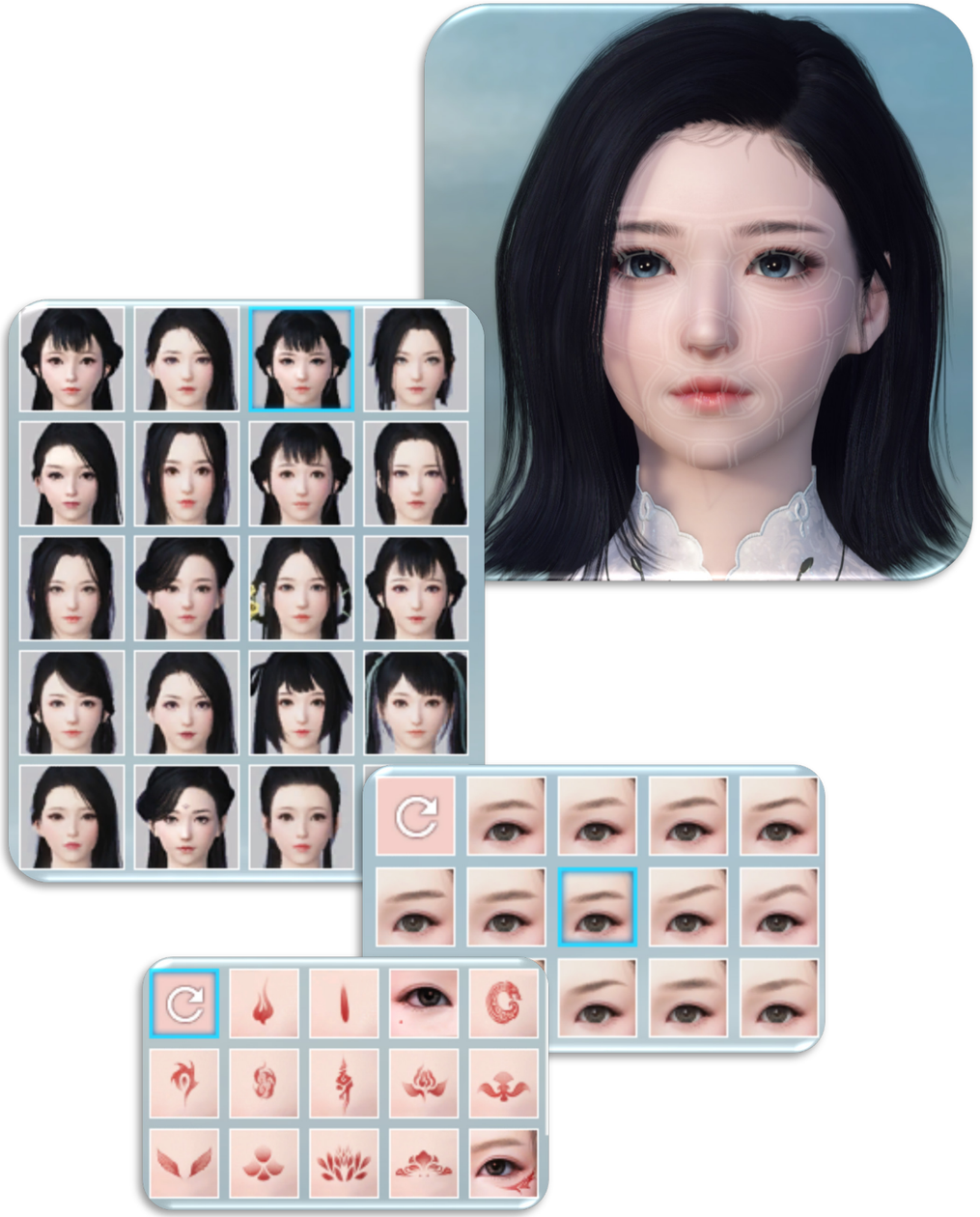
What attributes can be edited?

- Facial shape
- Hairstyles
- More makeups

  e.g. Eyebrow, Eyeshadow,

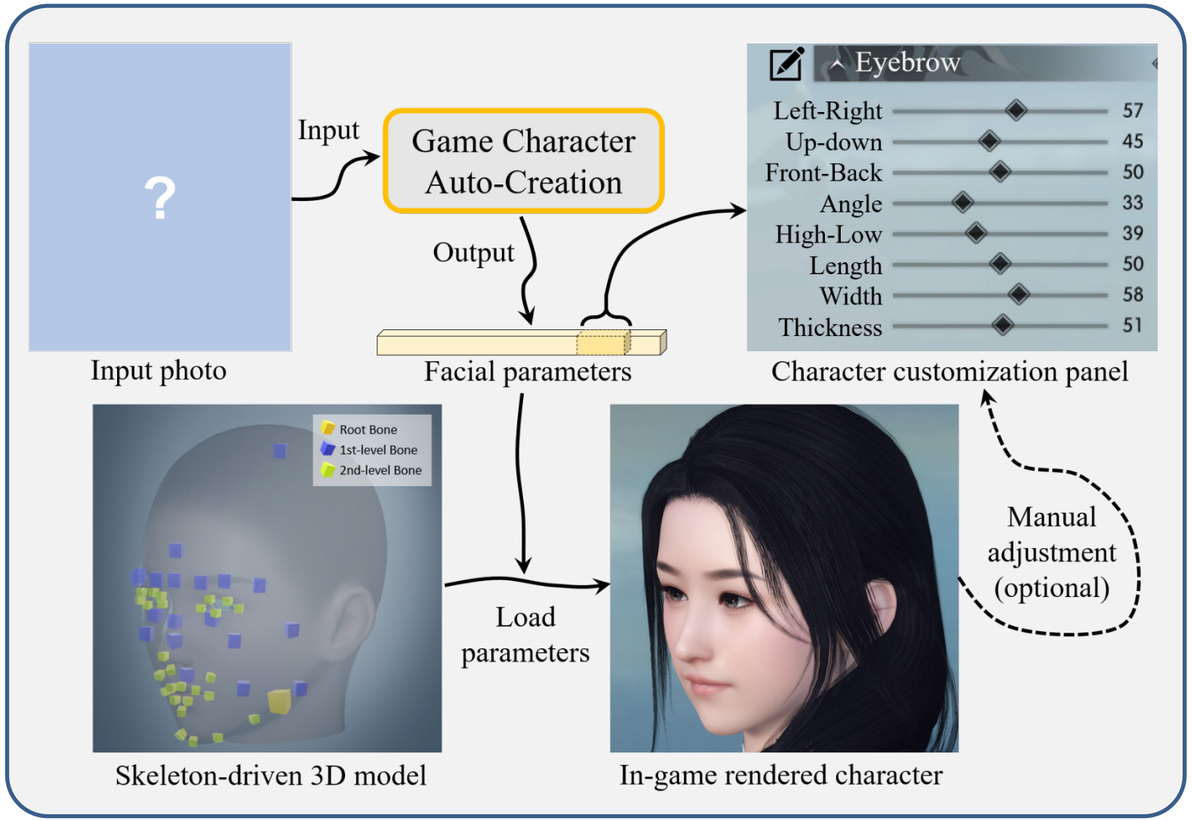
  Pupil color, Blusher, Tattoo,

  …



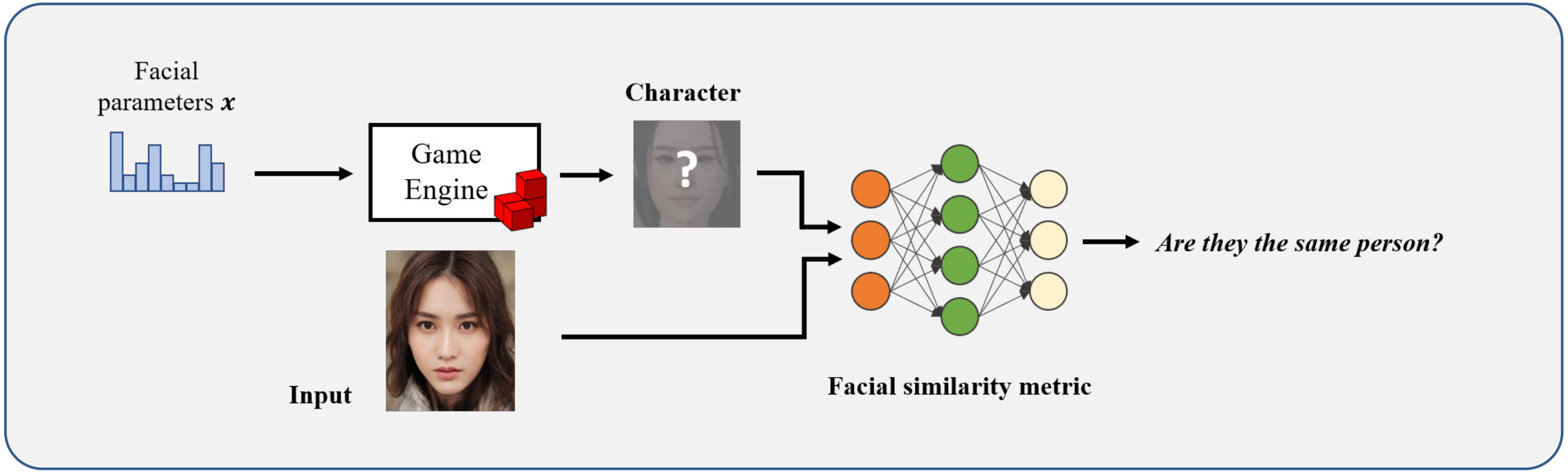
Our goals

- Generate a set of parameters based on the input photo
- These parameters can be recognized by the game
- The created character looks like the input photo
- Players are allowed to further edit the created character



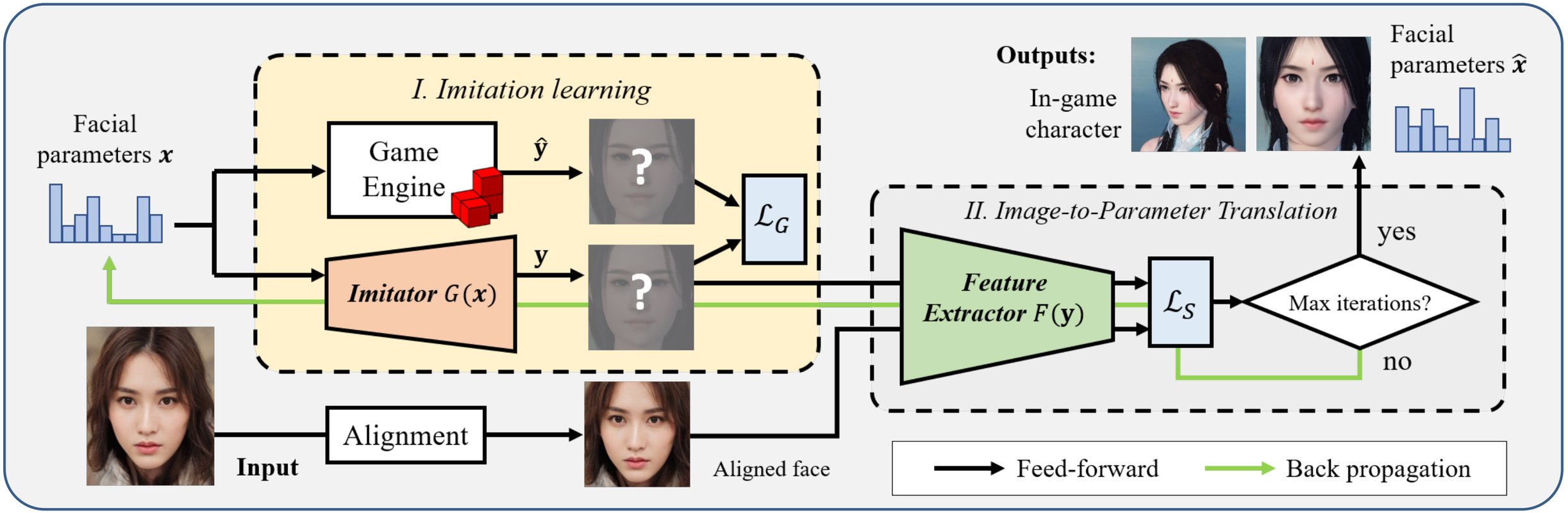
Face-to-Parameter Translation

Challenges



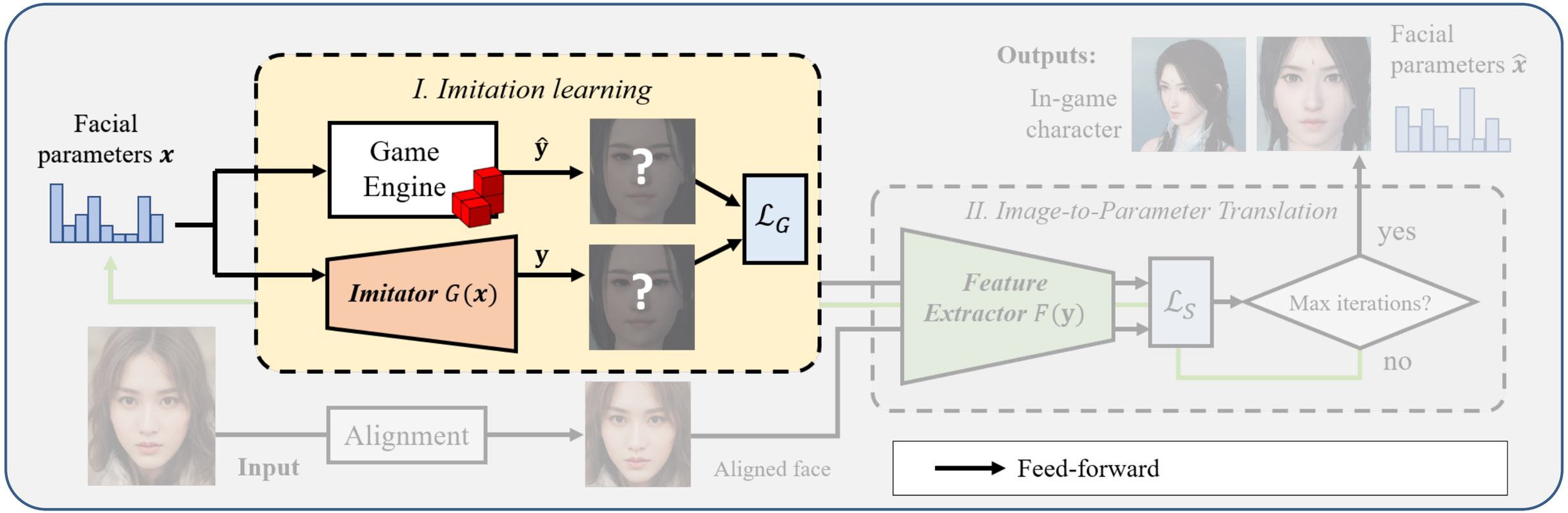
- Game engine is non-differentiable
- Define a metric to measure the similarity between input photo and created character

Methodology



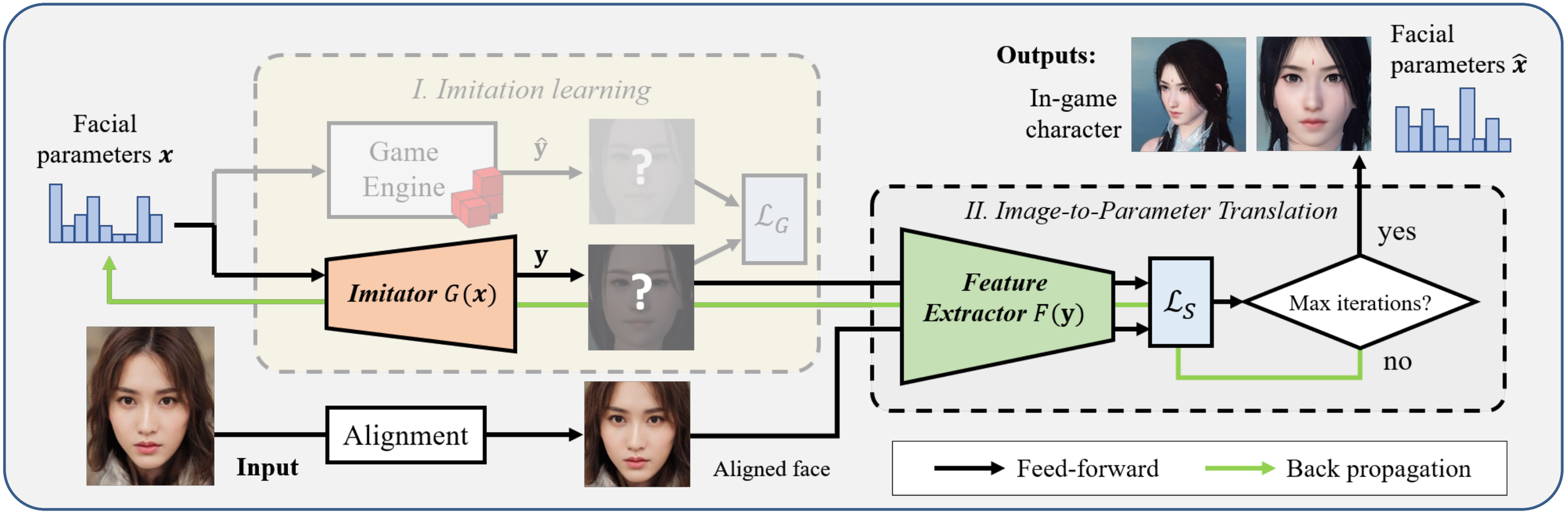
- Phase 1: Train a generative neural network as a renderer
- Phase 2: Define the facial similarity and auto-create characters by the gradient descent method

Methodology



- Phase 1: Train a generative neural network as a renderer
- Phase 2: Define the facial similarity and auto-create characters by the gradient descent method

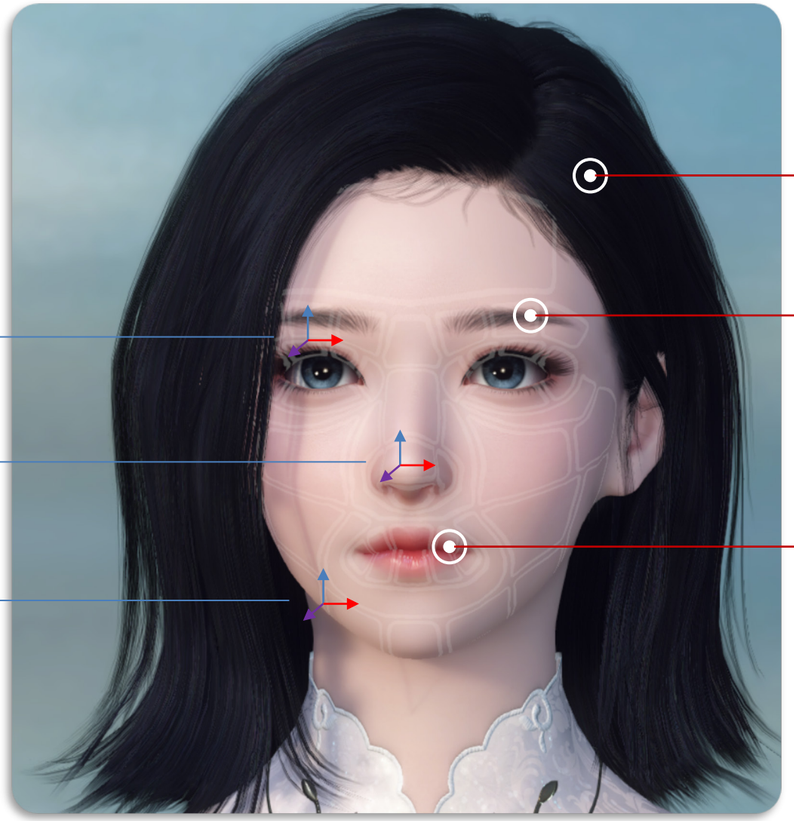
Methodology



- Phase 1: Train a generative neural network as a renderer
- Phase 2: Define the facial similarity and auto-create characters by the gradient descent method

Phase 1: Imitation learning

Facial parameters



Hairstyle

Eyebrow

Lipstick

Eye bone

Nose bone

Jaw bone

…

Normalized in [0,1]

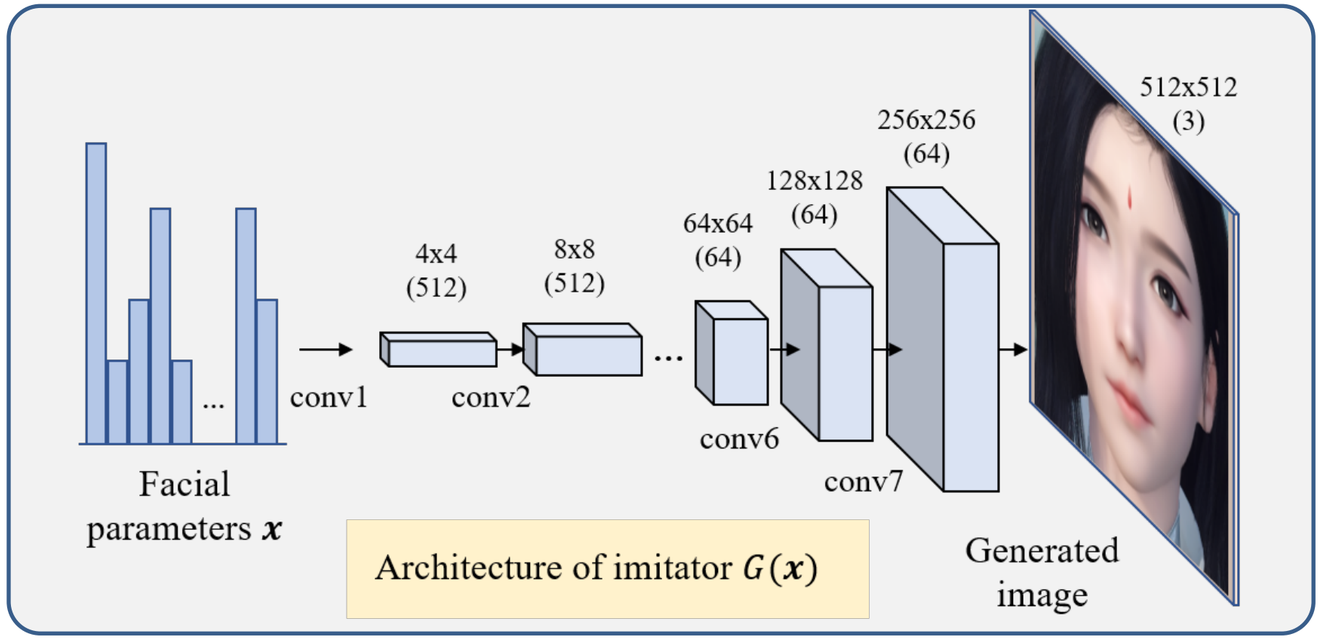
One-hot Encoding

x: continuous parameters | discrete parameters $\in [0,1]$

208d

102d

Imitation learning



- **Structure**
  - Input: facial parameters ($\boldsymbol{x} \in \mathcal{R}^{208+102}$)
  - Output: game character facial image ($G(\boldsymbol{x}) \in \mathcal{R}^{512\times512\times3}$)
  - 8 transposed convolution layers attached with ReLU and BN layers
  - Feature size is doubled after each convolution layer

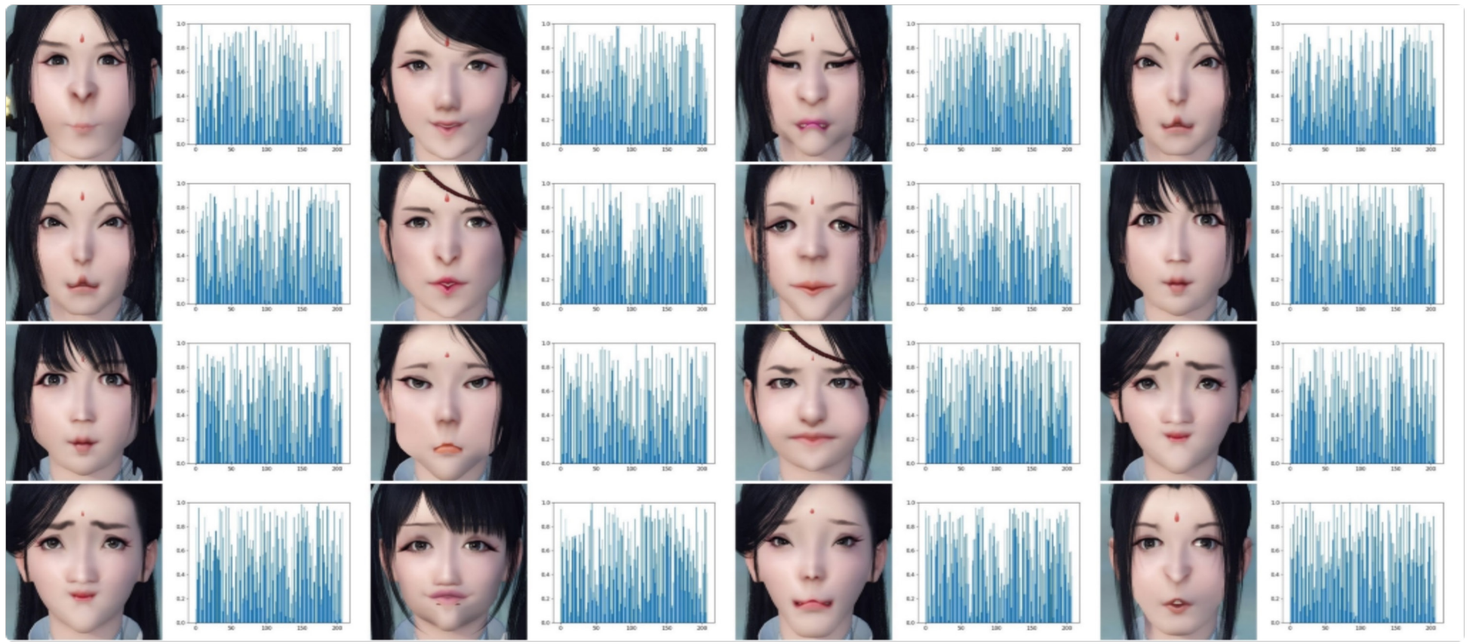
Imitation learning

- **Data**
  - Randomly generate facial parameters obeying uniform distribution

$$x_{continuous} \sim \mathcal{U}(0,1)$$

$$x_{discrete} \sim \mathcal{U}\{0,1,\dots,N\}$$

  - Render facial parameters in game (20000 pictures for each kind of character)
  - Sample pairs:

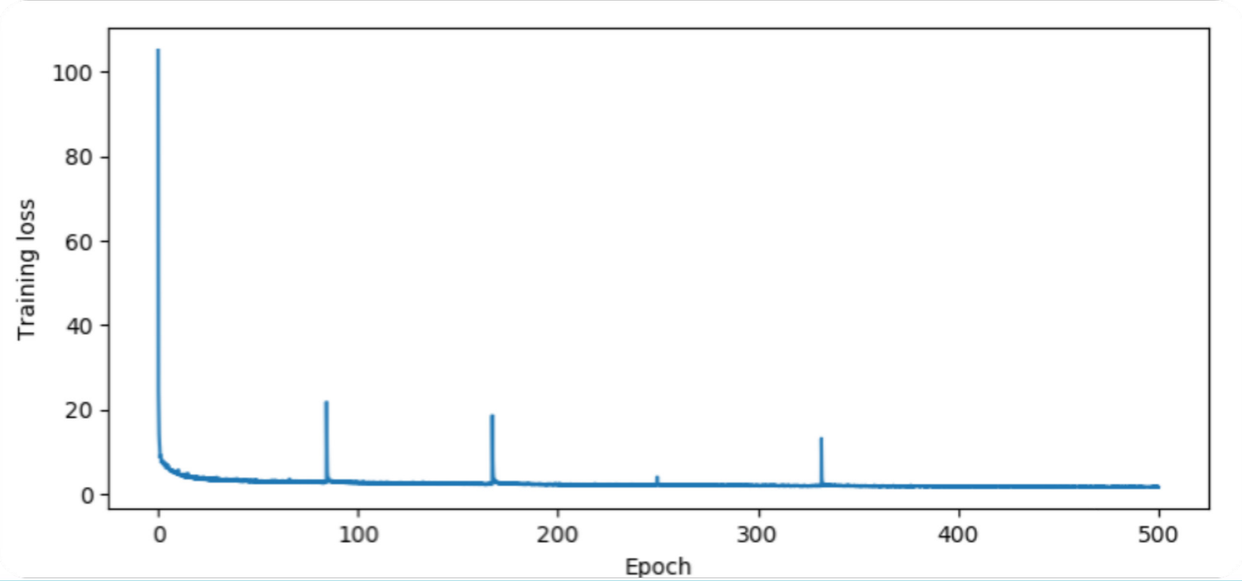


Imitation learning

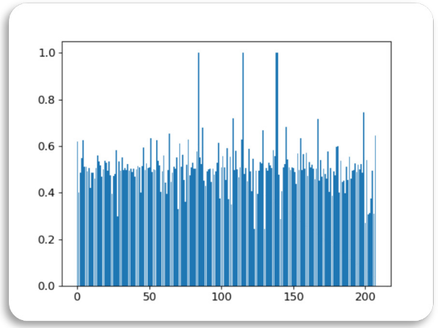
- **Training**
  - Target function:

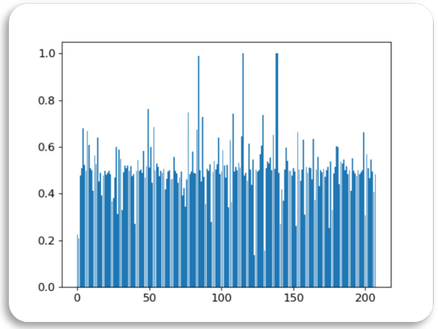
$$\min_{G} E_{\boldsymbol{x}\sim \mathrm{u}(\boldsymbol{x})}[\|G(\boldsymbol{x}) - Engine(\boldsymbol{x})\|_1]$$

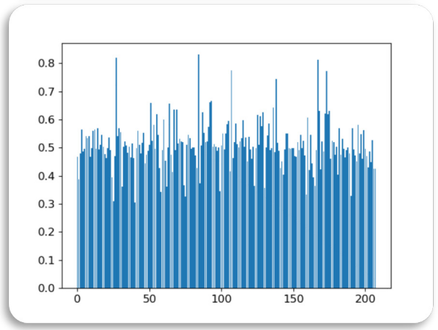
  - Optimizer: Stochastic Gradient Descent with momentum
  - Configuration: learning rate = 0.01, momentum = 0.9, batch size = 16, maximum epoch = 500, learning rate decay = 10% per 50 epoch



Imitation learning







Input facial parameters x

Imitator
$G(x)$

Imitator
$G(x)$

Imitator
$G(x)$

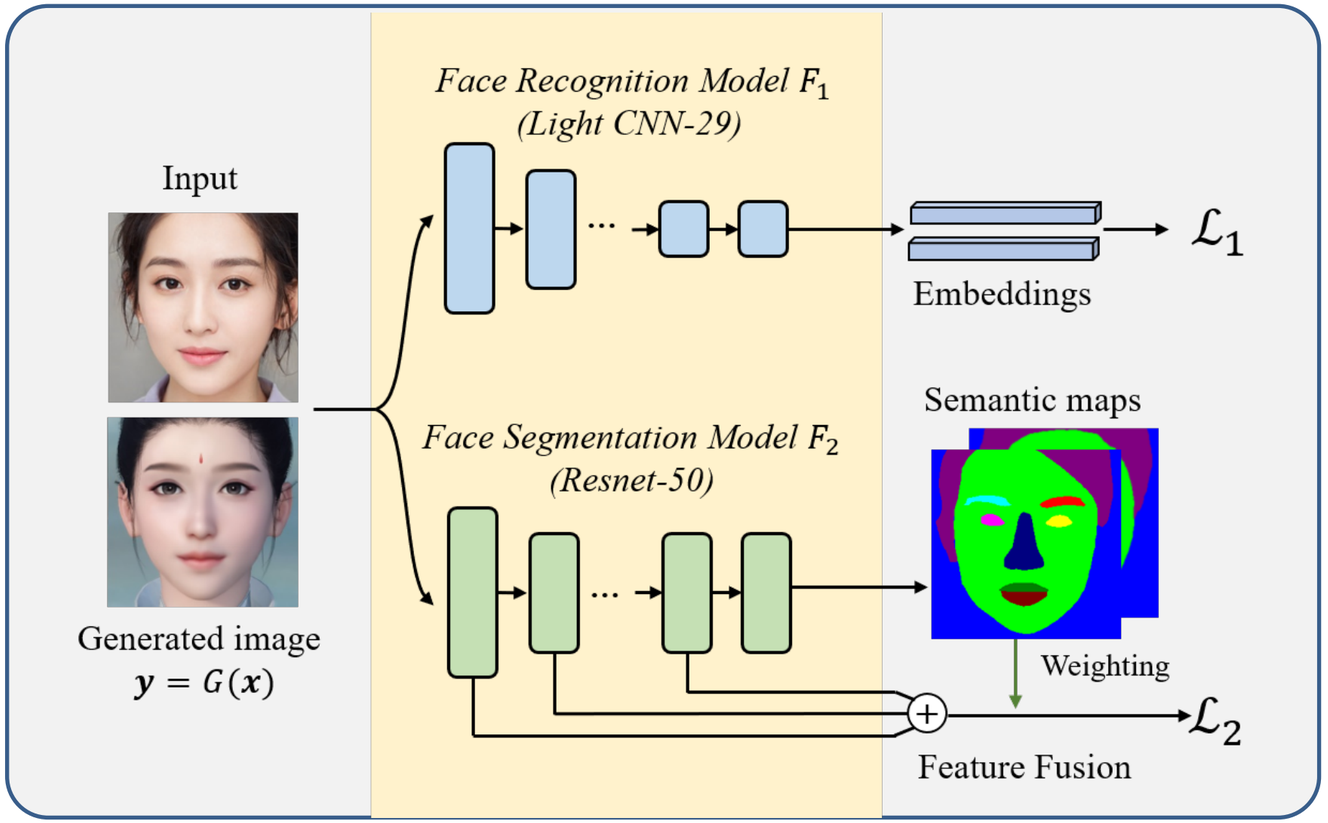
Generated face

Ground truth

* None of the three pairs are selected from the imitator dataset

Phase 2: Face-to-Parameter Translation

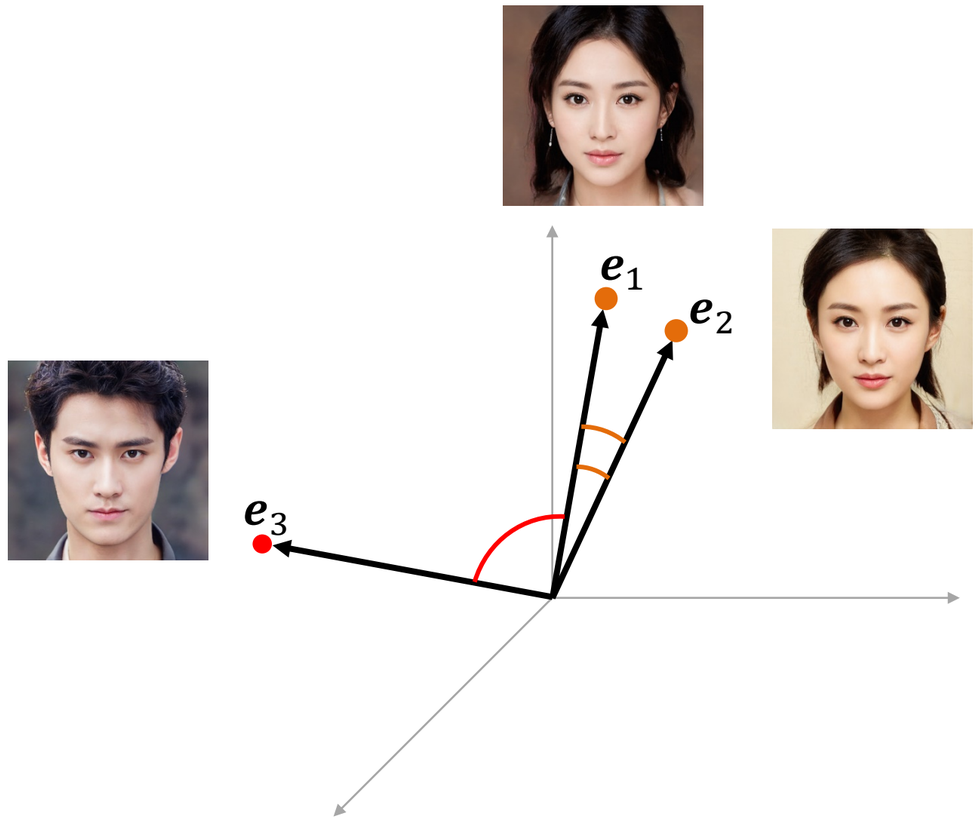
Facial Similarity Measurement



- Use discriminative networks to model human perception
- Input: facial photo (y_r) & generated image ($G(x)$)
- Output: identity loss ($\mathcal{L}_1$) and content (shape) loss ($\mathcal{L}_2$)

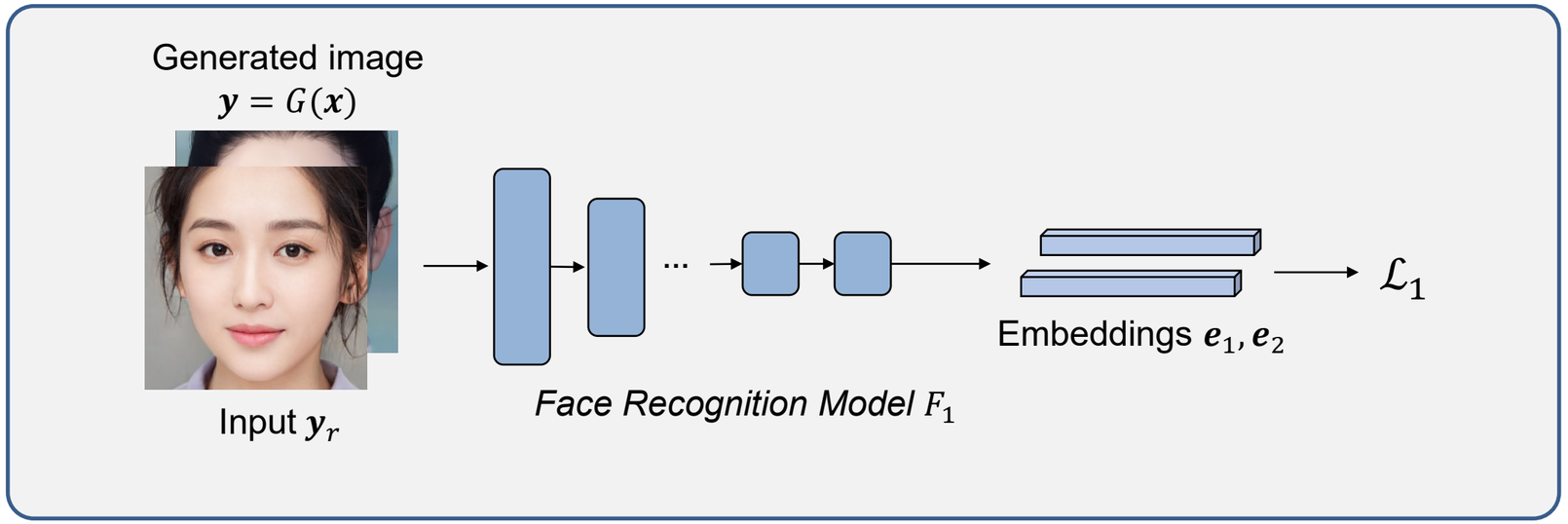
Face Recognition Model F_1

- **Facial embedding**
  - A vector used on identity information representation
  - Photos of the same person share similar embeddings



Face Recognition Model F_1

- **Model**
  - Pretrained neural network – Light CNN-29v2*
  - Target: Global face information extraction
  - Input: 128x128 gray images (input & generated ones)
  - Output: 256d facial embeddings
  - Fast (~1ms on GTX1080Ti) and accurate (99.43% on LFW**)



* Wu *et al.,* TIFS2018, https://github.com/AlfredXiangWu/LightCNN
** Labelled Faces in the Wild (A public face verification dataset)

Face Recognition Model F_1

- **Identity loss $\mathcal{L}_1$**
  - Definition

$$\mathcal{L}_1(\boldsymbol{x}) = 1 - \cos < F_1(\boldsymbol{y}_r), F_1\big(G(\boldsymbol{x})\big) >$$

  - Cosine distance

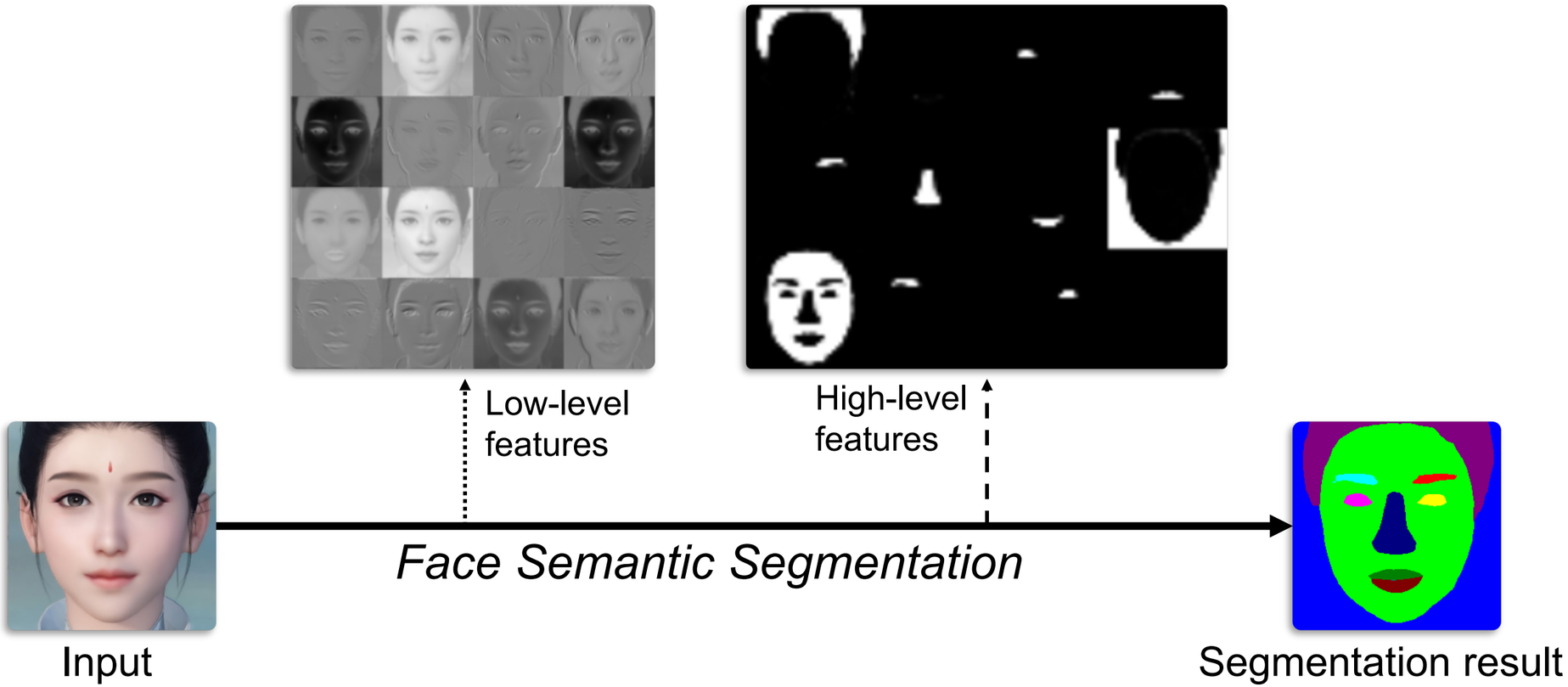
$$\cos(\boldsymbol{e}_1, \boldsymbol{e}_2) = \frac{< \boldsymbol{e}_1, \boldsymbol{e}_2 >}{\|\boldsymbol{e}_1\|_2 \|\boldsymbol{e}_2\|_2}$$

  - Optimal object

$$\min_{\boldsymbol{x}} \mathcal{L}_1(\boldsymbol{x})$$

Face Semantic Segmentation Model F_2

- **Facial features**
  - A group of 2D features used on facial detail representation



Face Semantic Segmentation Model F_2

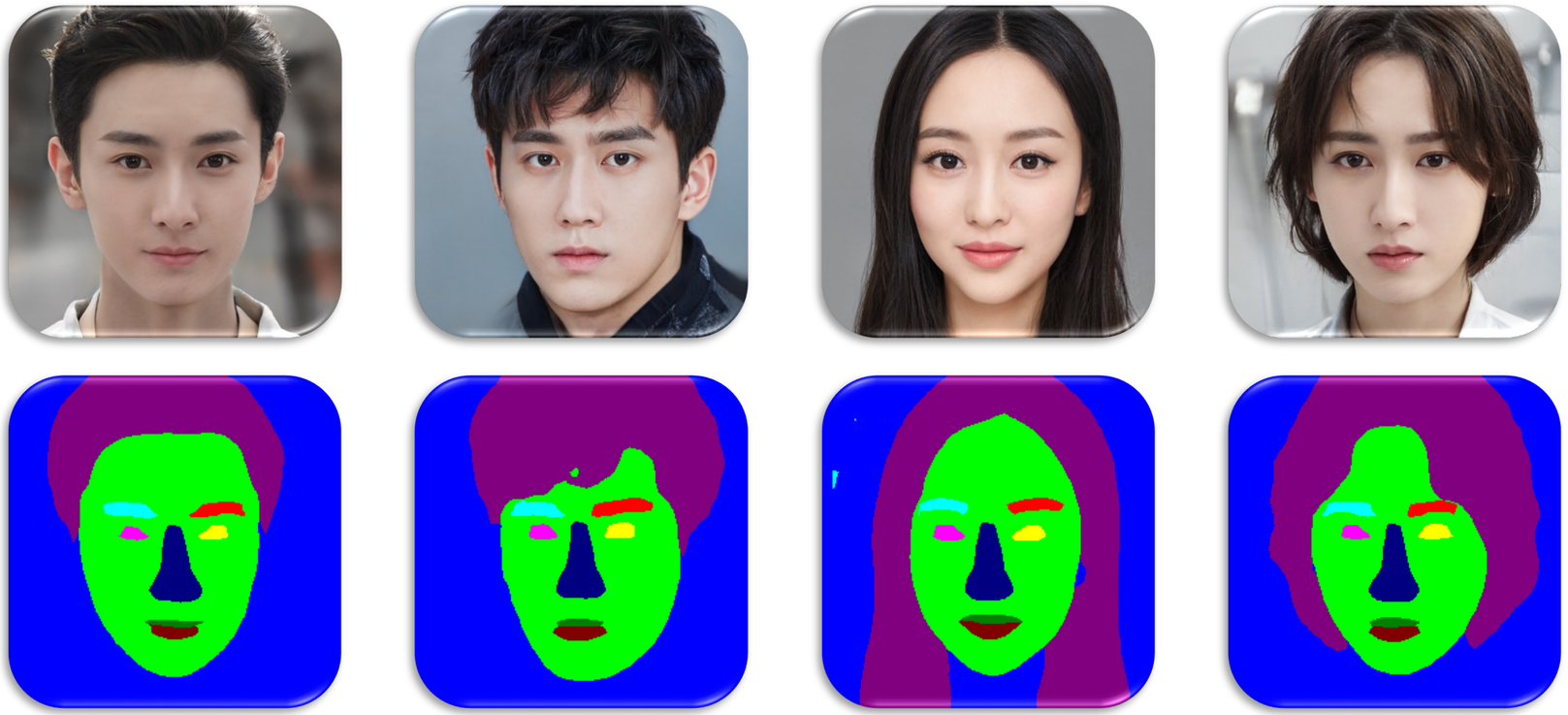
- **Model**
  - Target: Local face information extraction
  - Input: 256x256 RGB images
  - Output: facial features and semantic segmentation image
- **Structure**
  - A fully convolution network using Resnet-50 as backbone
  - Replace the last fully connected layer by 1x1 convolution layer
  - Increase output resolution from 1/32 to 1/8

	Layer	Component	Configuration	Output Resolution
Segmentation Model	Conv_1	Convolution + BN + ReLU	64x7x7 / 2	$\frac{1}{2} \times \frac{1}{2}$
	MaxPool	MaxPool	3x3 / 2	$\frac{1}{4} \times \frac{1}{4}$
	Conv_2	3 x Bottleneck	64 / 2	$\frac{1}{8} \times \frac{1}{8}$
	Conv_3	4 x Bottleneck	128 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_4	6 x Bottleneck	256 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_5	3 x Bottleneck	512 / 1	$\frac{1}{8} \times \frac{1}{8}$
	Conv_6	Convolution	11x1x1 / 1	$\frac{1}{8} \times \frac{1}{8}$

Resnet-50

Face Semantic Segmentation Model F_2

- **Training Dataset**
  - Thousands of human photos with fine label
  - 11 classes including eye, nose, mouth, etc.
  - Data augmentation by scaling and shifting

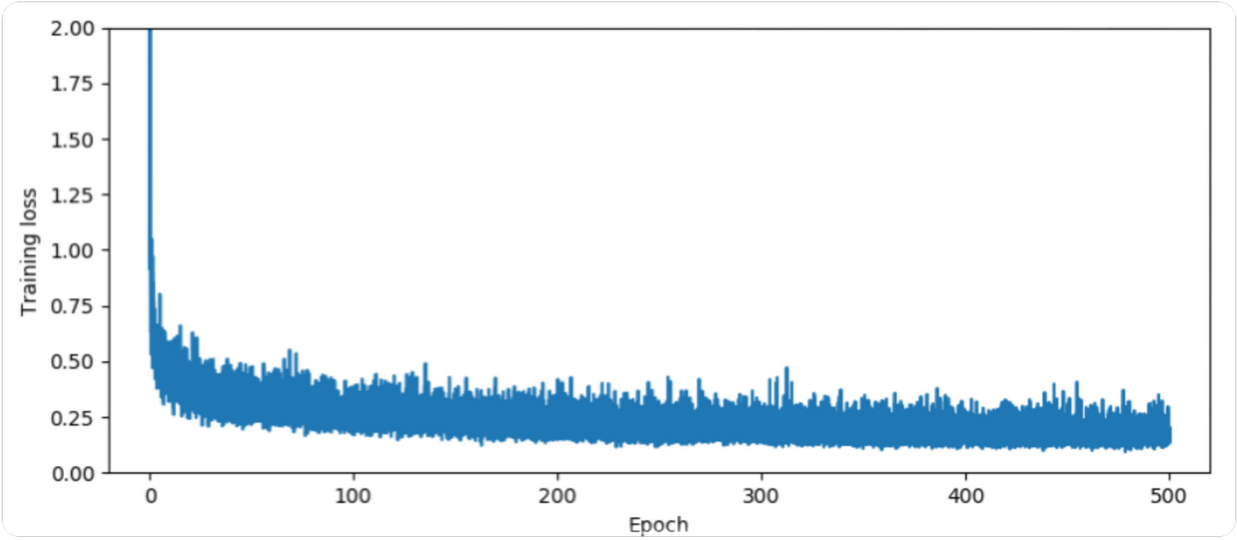


Face Semantic Segmentation Model F_2

- **Training**
  - Configuration is similar to the imitator's (except learning rate = 0.001)
  - Pixel-wise cross-entropy loss

$$L_{CE} = -\sum_{i=1}^{\mathrm{K}} y_i \log p_i$$

  - Training procedure:



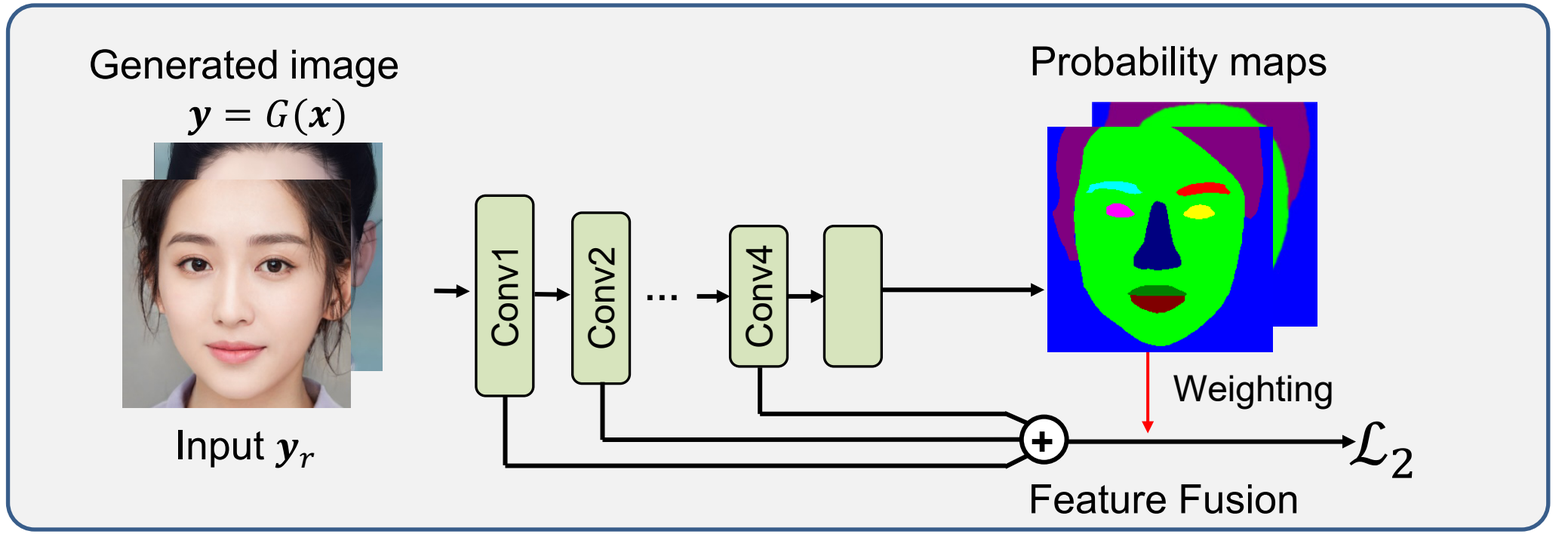
Face Semantic Segmentation Model F_2

- **Content loss**
  - Definition

$$\mathcal{L}_2(\boldsymbol{x}) = \left\| F_2(\boldsymbol{y}_r) - F_2\big(G(\boldsymbol{x})\big) \right\|_1$$

  - Use semantic probability maps to enhance facial features
  - Optimal object

$$\min_{\boldsymbol{x}} \mathcal{L}_2(\boldsymbol{x})$$



Face-to-Parameter Translation

- Overall loss function

$$\mathcal{L}_s = \alpha\mathcal{L}_1 + \mathcal{L}_2$$

- Overall optimal problem

$$\min_{\boldsymbol{x}} \mathcal{L}_s(\boldsymbol{x})$$

- Gradient descent method

  - Parameter update: $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} - \eta \frac{\partial \mathcal{L}_s(\boldsymbol{x}_{i-1})}{\partial \boldsymbol{x}_{i-1}}, \quad i = 1,2,\dots,i_{max}$

  - Gradient compute:

$$\frac{\partial \mathcal{L}_s(\boldsymbol{x})}{\partial \boldsymbol{x}} = \alpha \frac{\partial \mathcal{L}_1(\boldsymbol{x})}{\partial \boldsymbol{x}} + \frac{\partial \mathcal{L}_2(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

$$\frac{\partial \mathcal{L}_1(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial \mathcal{L}_1(\boldsymbol{x})}{\partial F_1(G(\boldsymbol{x}))} \frac{\partial F_1(G(\boldsymbol{x}))}{\partial G(\boldsymbol{x})} \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

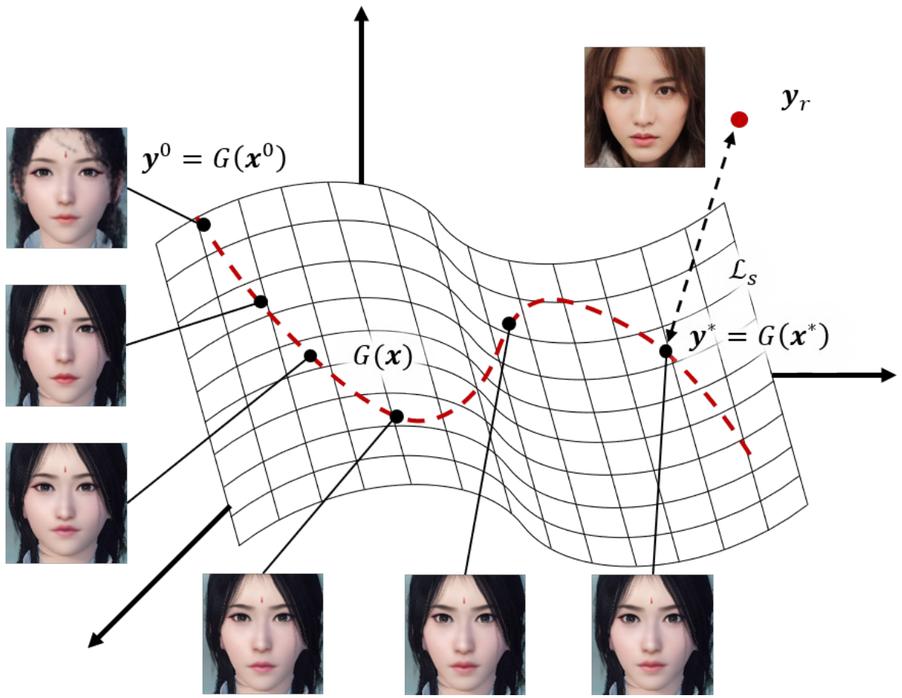
$$\frac{\partial \mathcal{L}_2(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial \mathcal{L}_2(\boldsymbol{x})}{\partial F_2(G(\boldsymbol{x}))} \frac{\partial F_2(G(\boldsymbol{x}))}{\partial G(\boldsymbol{x})} \frac{\partial G(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

Manifold searching

- Geometrically, the gradient descent procedure can be viewed as a searching on the game face manifold:

$$\Rightarrow \min_{\boldsymbol{x}} \alpha(1 - \cos < F_1(\boldsymbol{y}_r), F_1(\boldsymbol{y}) >) + \|F_2(\boldsymbol{y}_r) - F_2(\boldsymbol{y})\|_1$$

$$s.t. \;\; \boldsymbol{y} = G(\boldsymbol{x})$$



Character Auto-Creation

- Step 1: Align the user-uploaded photo based on the default game face
- Step 2: Solve the following optimization problem

$$\min_{\boldsymbol{x}} \mathcal{L}_s(\boldsymbol{x})$$

  - Step 2.1: Freeze all networks ($G\ F_1\ F_2$), initialize $\boldsymbol{x} = \boldsymbol{x}_0$, set learning rate $\eta = 10$ and max iterations $i_{max} = 10$
  - Step 2.2: Use gradient descent method to update $\boldsymbol{x}$
  - Step 2.3: When reaching the max iterations, output optimized facial parameters $\boldsymbol{x}^*$
- Step 3: Write $\boldsymbol{x}^*$ into the game and obtain the character

Aligned input

Optimizing

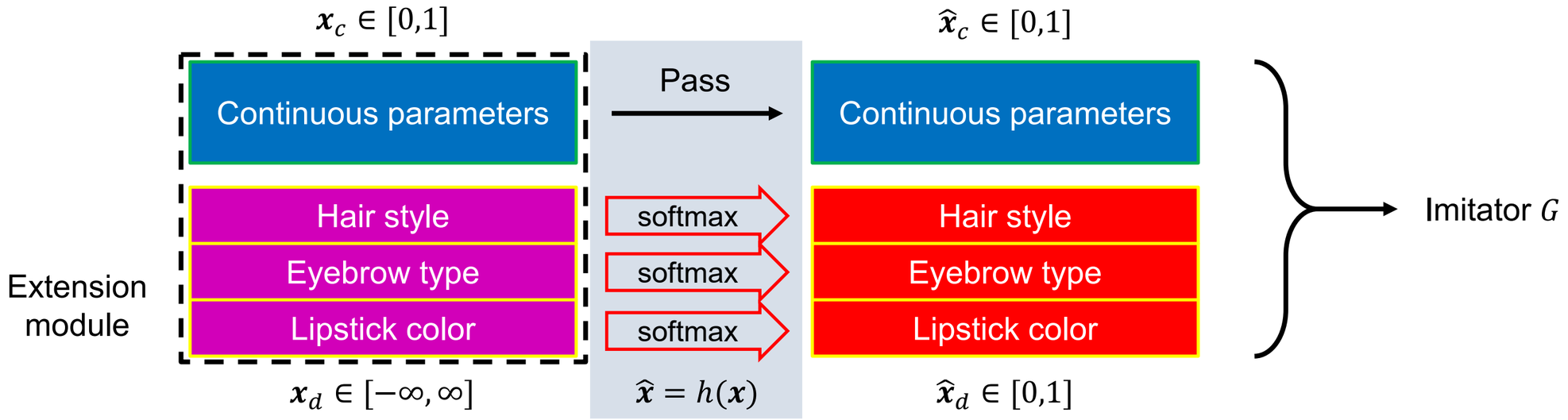
In-game character

Player editable

Changes:
Eye shadow
Lipstick color

The solution of discrete parameter optimization

- Discrete parameters are hard to optimize directly
- SoftMax function ($softmax(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}}}{\sum e^{\boldsymbol{x}}}$) is differentiable and can approximate one-hot vector by generating a probability vector

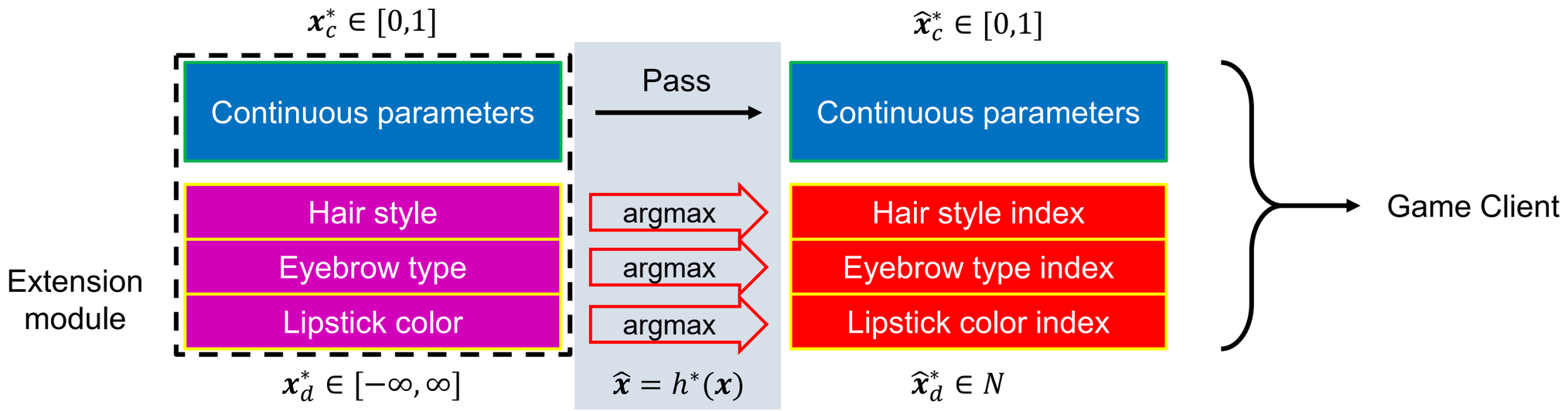


- New optimal problem

$$\min_{\boldsymbol{x}} \mathcal{L}_s\big(h(\boldsymbol{x})\big)$$

The solution of discrete parameter optimization

- When obtaining the optimized facial parameters $\boldsymbol{x}^*$, softmax function will be replaced by argmax function to get the index of makeups



Note: If $y = \text{argmax}(\boldsymbol{x})$, then y is the index of the maximum element in the vector $\boldsymbol{x}$

Environment and Performance

Action	Software	Hardware	Speed
Imitator training dataset	Windows 10, Game client	I7, RAM 16GB, GTX1060	20,000 pictures / 10 hours
Imitator training	Linux, PyTorch	E5, RAM 128GB, GTX1080Ti x4	1 day
Segmentation network training		E5, RAM 32GB, GTX1080Ti x1	1 day
Character Auto-Creation		E5, RAM 16GB, GTX1080Ti x1	1 fps

NetEase Games
Passion of gamers

FUXI-Lab

Auto-creation results

“Justice”

“Justice”

“Justice”

“Revelation Mobile”

“Revelation Mobile”

Discussion

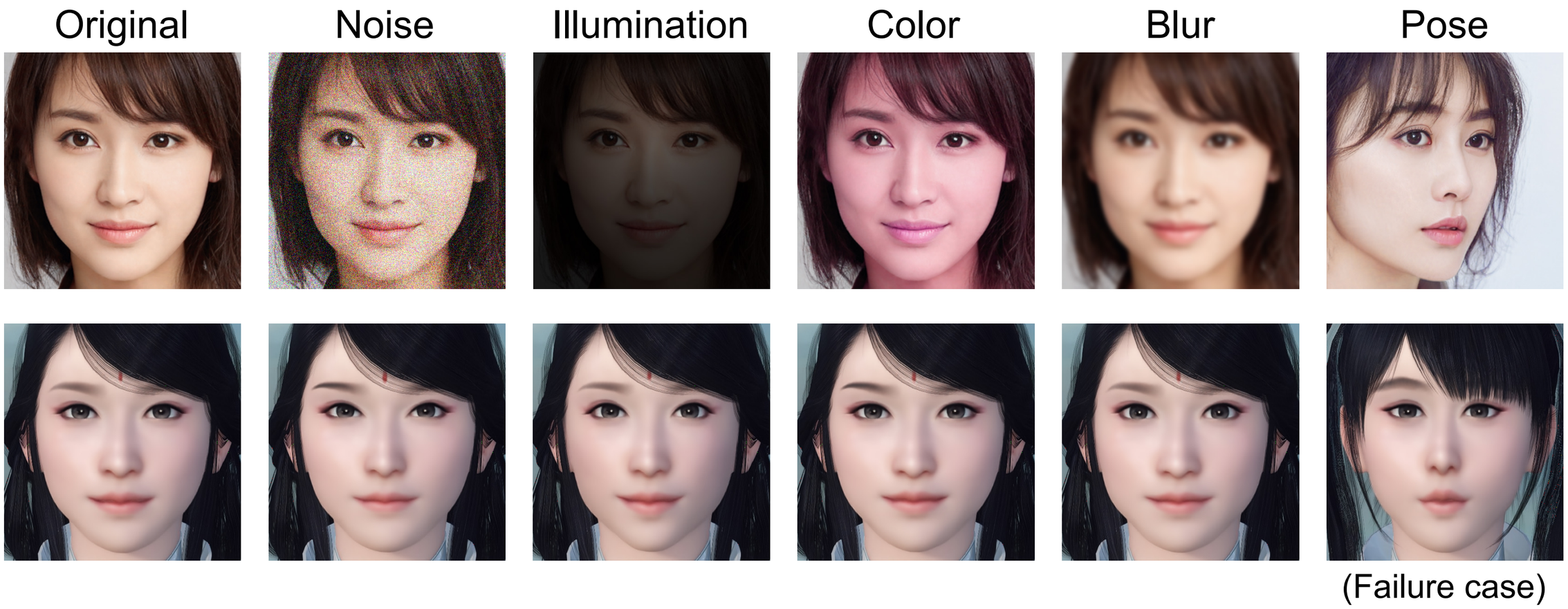
Advantages

- No labeled character data is required
- The generative network can easily fit the behavior of game engine regardless of the renderer type and 3D model structure
- Deep learning based features can represent human perception on face similarity evaluation

Limitations

- Sensitive to the pose, obstacle, etc.
- Discrete parameters are not well solved
- Human-like 3D model is required

Robustness experiment



Thanks!