Course Project

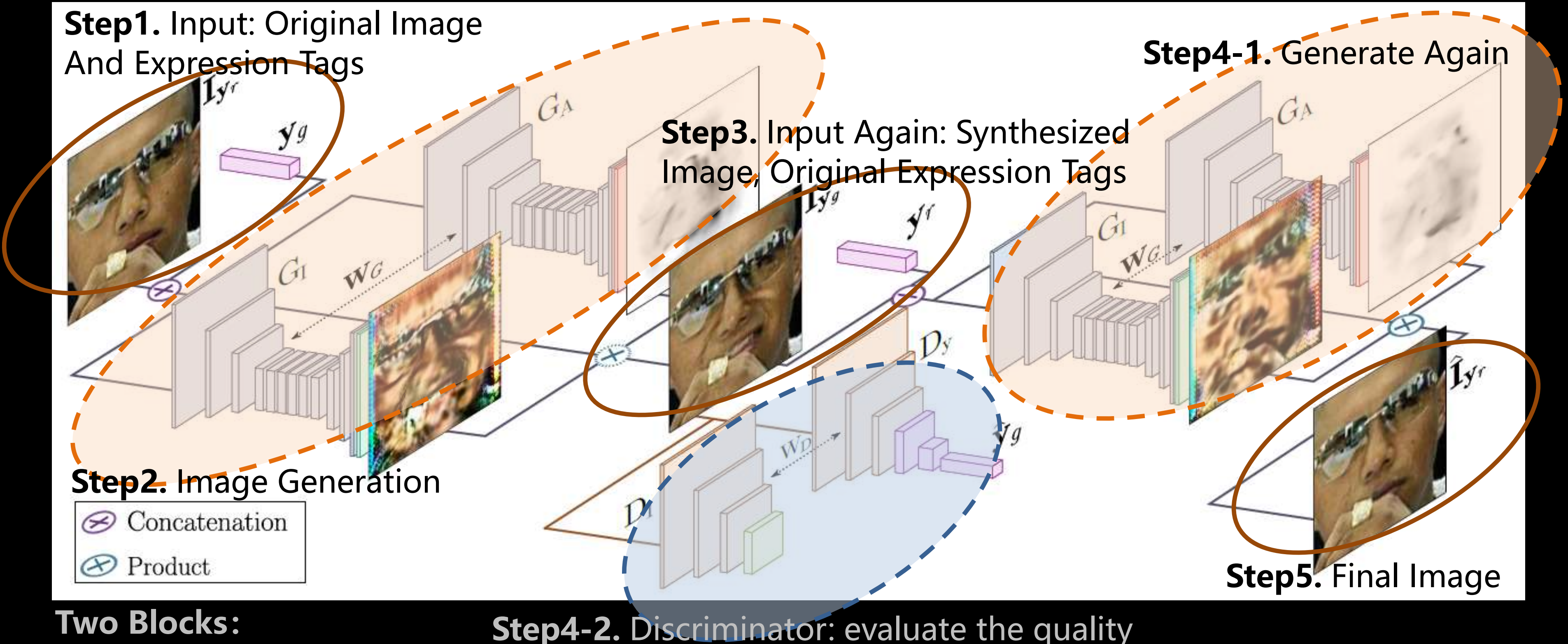
GANimation on TensorLayer
AND
Thoughts of a Better Train Dataset

艺术学院　朱峰　1801221169
软件与微电子学院　要曙丽　1901210768

Review

- **StarGAN** can only generate a discrete number of expressions, determined by the content of the dataset.
- This paper introduces a novel GAN conditioning scheme based on **Action Units (AU) annotations,** which describes in a continuous manifold the anatomical facial movements defining a human expression.
- We leverage on the recent **EmotioNet dataset**, which consists of one million images of facial expressions (we use 200,000 of them) of emotion in the wild annotated with discrete AUs activations.
- Additionally, we propose a fully unsupervised strategy to train the model, that only requires images annotated with their activated AUs, and exploit **Attention Mechanisms** that make our network robust to changing backgrounds and lighting conditions.

Framework



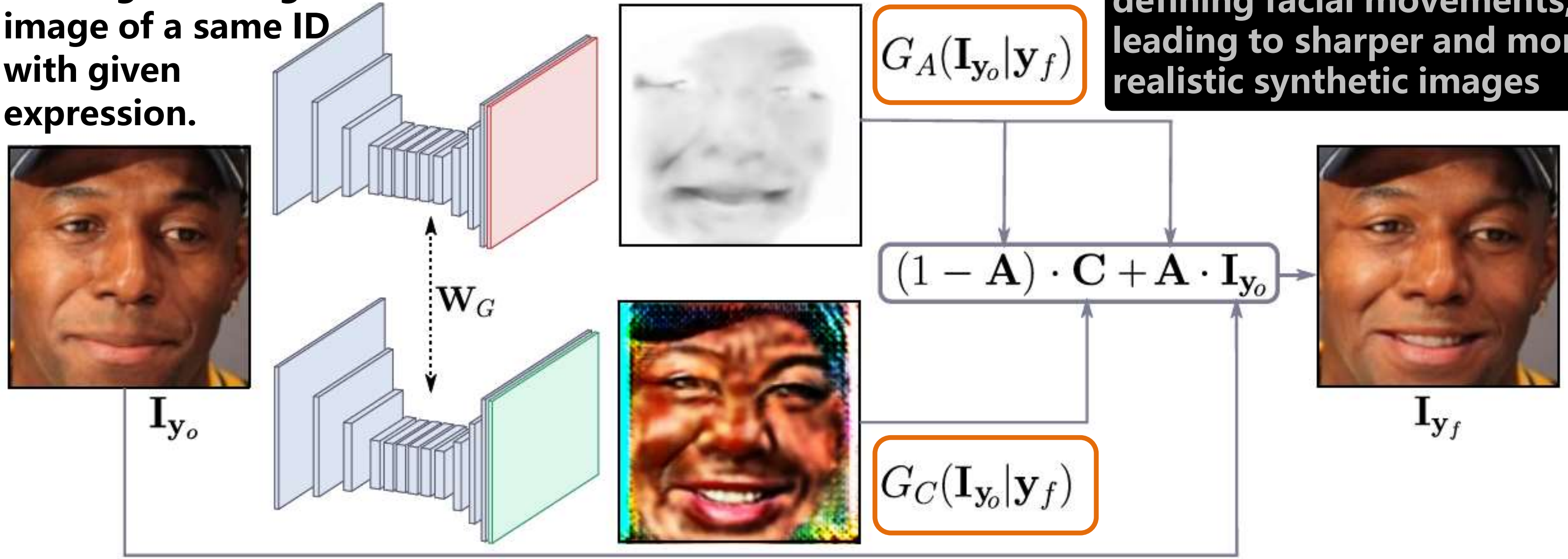
Two Blocks:
Generator: GA, GI
Discriminator: DI, Dy

Step4-2. Discriminator: evaluate the quality of the generated image and its expression

Attention-based Generator

Attention mask A:
the generator can focus exclusively on the pixels defining facial movements, leading to sharper and more realistic synthetic images

Goal: generating image of a same ID with given expression.

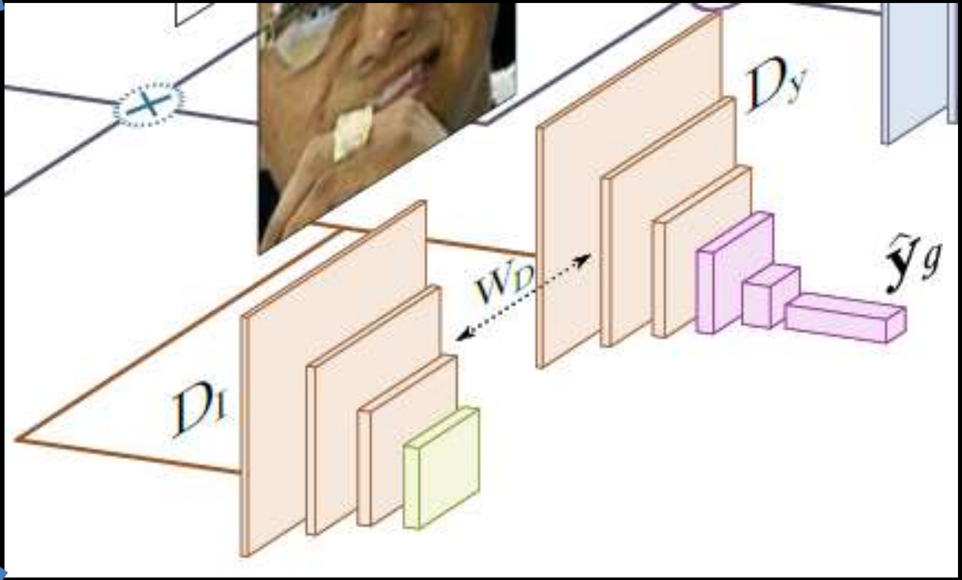


Two Blocks:
Generator: GA, GI
Discriminator: DI, Dy

Color mask C: the generator does not need to render static elements

Discriminator

Goal: Evaluate the quality of the generated image (Real Photo? Given ID?) and its expression (Given tag?)



Two Blocks:
Generator: GA, GI
Discriminator: DI, Dy

Loss Function

Goal of Generator: Generating image of a same ID with given expression.

Goal of Discriminator: Evaluate the quality of the generated image (Real Photo? Given ID?) and its expression (Given tag?)

Gradient penalty

1 Image Adversarial Loss

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] - \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(\mathbf{I}_{\mathbf{y}_o})] + \boxed{\lambda_{\mathrm{gp}}\mathbb{E}_{\tilde{I}\sim\mathbb{P}_{\tilde{I}}}\left[(\|\nabla_{\tilde{I}} D_{\mathrm{I}}(\tilde{I})\|_2 - 1)^2\right]}$$

2 Attention Loss

$$\lambda_{\mathrm{TV}}\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\sum_{i,j}^{H,W}\left[(\mathbf{A}_{i+1,j} - \mathbf{A}_{i,j})^2 + (\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j})^2\right]\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[\|\mathbf{A}\|_2] \quad (2)$$

3 Conditional Expression Loss

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathbf{y}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)) - \mathbf{y}_f\|_2^2\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathbf{y}}(\mathbf{I}_{\mathbf{y}_o}) - \mathbf{y}_o\|_2^2\right]. \quad (3)$$

4 Identity Loss

$$\mathcal{L}_{\mathrm{idt}}(G, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o, \mathbf{y}_f) = \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|G(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)|\mathbf{y}_o) - \mathbf{I}_{\mathbf{y}_o}\|_1\right]. \quad (4)$$

Full Loss

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\mathrm{I}}(G, D_{\mathrm{I}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) + \lambda_{\mathrm{y}}\mathcal{L}_{\mathrm{y}}(G, D_{\mathrm{y}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g) \quad (5)\\ &+ \lambda_{\mathrm{A}}\left(\mathcal{L}_{\mathrm{A}}(G, \mathbf{I}_{\mathbf{y}_g}, \mathbf{y}_r) + \mathcal{L}_{\mathrm{A}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g)\right) + \lambda_{\mathrm{idt}}\mathcal{L}_{\mathrm{idt}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g).\end{aligned}$$

Implementation Details

Generator: built from **CycleGAN;** slightly modified by substituting the last convolutional layer with two parallel convolutional layers, one to regress the **color mask C** and the other to dene the **attention mask A.**

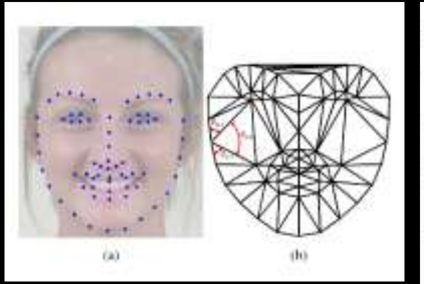
Discriminator: adopted the **PatchGan** architecture, with the **gradient penalty** computed with respect to the entire batch.

Other Details: The model is trained on the **EmotioNet dataset**. We use a subset of **200,000 samples (over 1 million)** to reduce training time. We use Adam with learning rate of 0.0001, beta1 0.5, beta2 0.999 and batch size 25. We train for 30 epochs and linearly decay the rate to zero over the last 10 epochs. Every 5 optimization steps of the critic network we perform a single optimization step of the generator. The model takes two days to train with a single GeForce GTX 1080 Ti GPU.

EmotioNet (CVPR 2016) - Overview

EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild

SHAPE: Angles + Lengths ⟶ Action Units



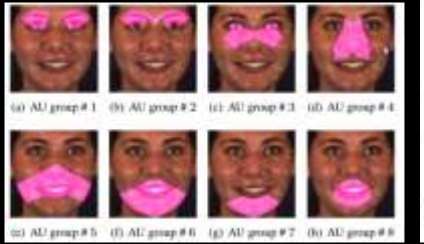
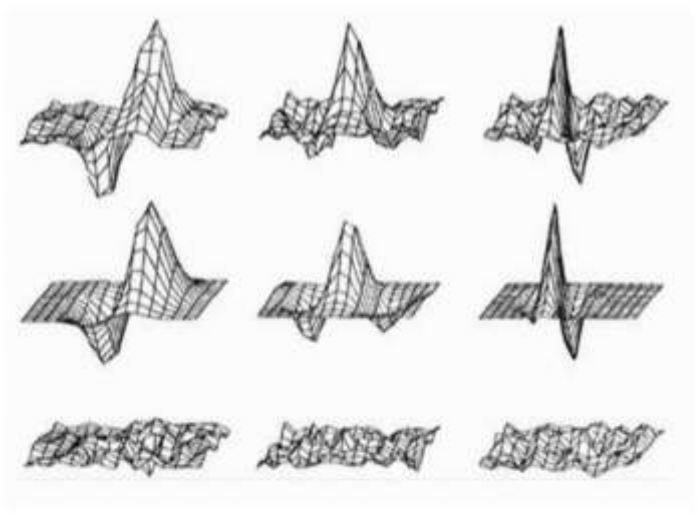


Table 2. AU partition rule

AU group	AU NO	RoI NO
#1* (∈# 2)	AU 1, AU 2, AU 5, AU 7	1, 2, 5, 6, 8, 9, 12, 13, 40, 41, 42, 43
#2	AU 4	1, 2, 3, 4, 5, 6, 8, 9, 12, 13, 40, 41
#3	AU 6	16, 17, 18, 19, 42, 43
#4 (∈ # 3)	AU 9	10, 11, 17, 18, 22, 23
#5 (∈ # 6)	AU 10, AU 11, AU 12, AU 13, AU 14, AU 15	21, 22, 23, 24, 25, 26, 27, 28, 37
#6 (∈ # 5)	AU 16, AU 20, AU 25, AU26, AU 27	25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37
#7 (∈ # 6)	AU 17	29, 30, 31, 32, 33, 34, 35, 36
#8 (∈ # 5, # 6)	AU 18, AU 22, AU 23, AU 24, AU 28	26, 27, 29, 30, 31, 32, 37

Note: Symbol * means the corresponding AU group have symmetrical regions. Symbol ∈ indicates the "label fetch".

SHADE: Gabor Filter

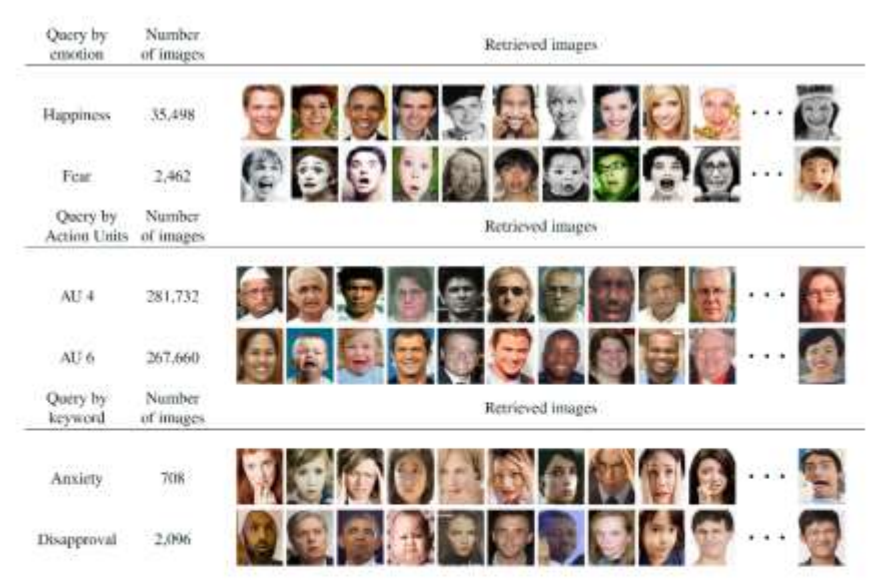


Shoulder Pain Database
Denver Intensity of Spontaneous Facial Action (DISFA) dataset
database of Compound Facial Expressions of Emotion (CFEE)

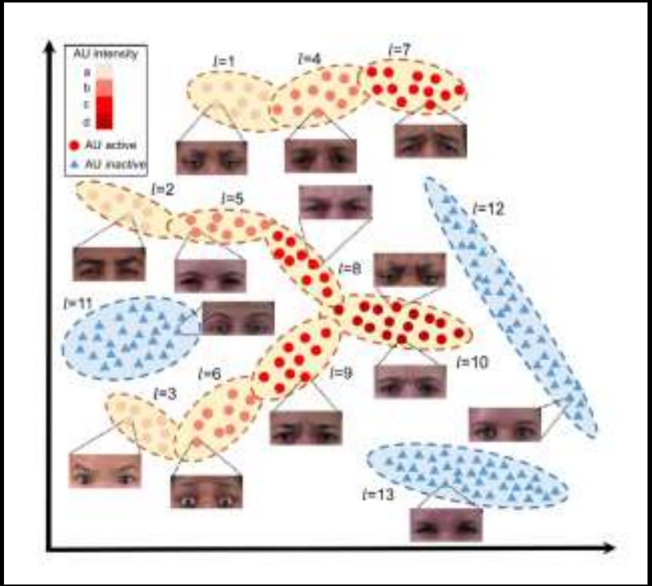
→ **Automatic Annotator** —1 Million Pics from WordNet→ **EmotioNet**

EmotioNet - 4 Limits

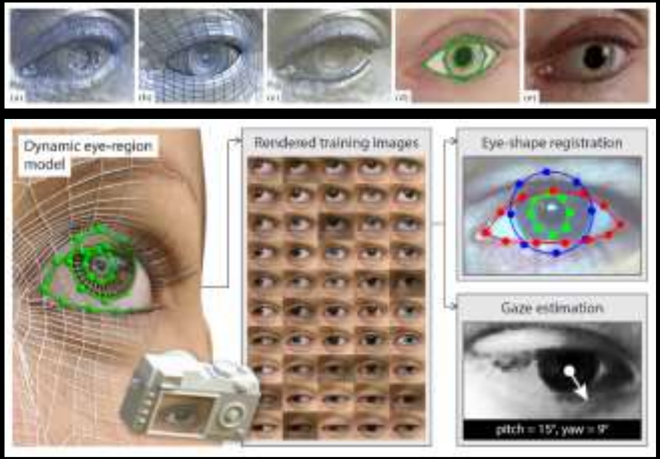
1 Lack of Various Directions
2 No Occlusion



3 Inaccurate Annotation
4 Discrete AU Intensity



Synthetic Data vs. EmotioNet



~~Lack of~~ Various Directions

~~No~~ Occlusion

~~In~~accurate Annotation

~~Discrete~~ AU Intensity

Body Pose Estimation

R. Okada and S. Soatto, "Relevant feature selection for human pose estimation and localization in cluttered images," in *ECCV*, 2008.

J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from a single depth image," in *CVPR*, 2011.

Object Detection/ Recognition

L. Fu and L. B. Kara, "Neural network-based symbol recognition using a few labeled samples," *Computers & Graphics*, vol. 35, no. 5, pp. 955–966, 2011.

J. Yu, D. Farin, C. Krger, and B. Schiele, "Improving person detection using synthetic training data," in *ICIP*, 2010.

J. Liebelt and C. Schmid, "Multi-view object class detection with a 3d geometric model," in *CVPR*, 2010, pp. 1688–1695.

X. Peng, B. Sun, K. Ali, and K. Saenko, "Exploring invariances in deep convolutional neural networks using synthetic images," *arXiv preprint*, 2014.

Facial Landmark Localization

T. Baltrušaitis, P. Robinson, and L.-P. Morency, "3D constrained local model for rigid and non-rigid facial tracking," in *CVPR*, 2012.

L. A. Jeni, J. F. Cohn, and T. Kanade, "Dense 3D Face Alignment from 2D Videos in Real-Time," *FG*, 2015.

And Super Resolution, Frame Interpolation... etc.

Implementation - 2 Possible Methods

No.1 3D Reconstruction + Render

confidence	AU01	AU02	AU04	AU05	AU06	AU07	AU09
0.975	0.02	0.73	0.48	0.36	0	0	0.31
AU14	AU15	AU17	AU20	AU23	AU25	AU26	AU45
0	0.96	0.74	0.44	0.05	0	0.16	0

confidence	AU01	AU02	AU04	AU05	AU06	AU07	AU09	AU10
0.975	0.09	0	0	0.18	0.25	0.6	0	0.37
AU12	AU14	AU15	AU17	AU20	AU23	AU25	AU26	AU45
0.53	0.57	0	0	0	0.28	0.95	0.32	0

Various Directions	Y
Occlusion	Y
Accurate Annotation	N
AU Intensity	Continuous
Data Production	Efficient

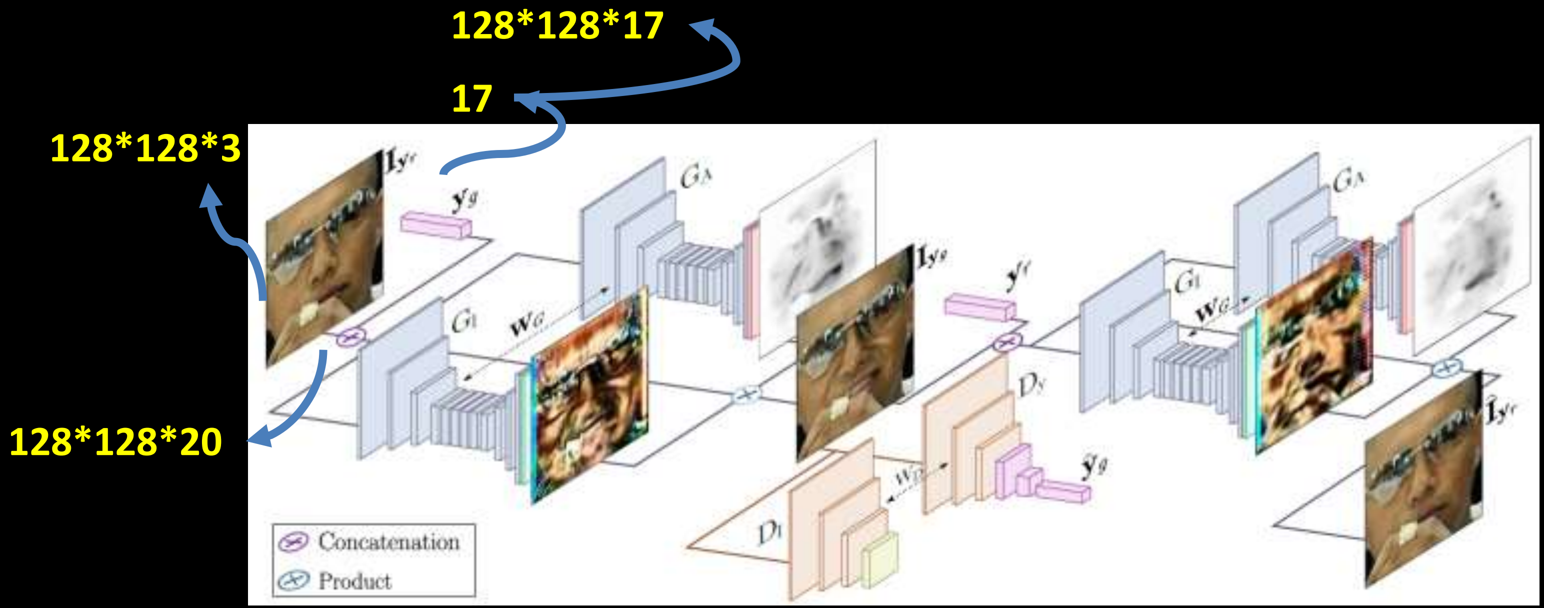
confidence	AU01	AU02	AU04	AU05	AU06	AU07	AU09	AU10
0.875	0.11	0.27	0	0.33	0	0	0	0
AU12	AU14	AU15	AU17	AU20	AU23	AU25	AU26	AU45
0	0	0	0.77	0	0.82	0	0	0

Implementation - 2 Possible Methods

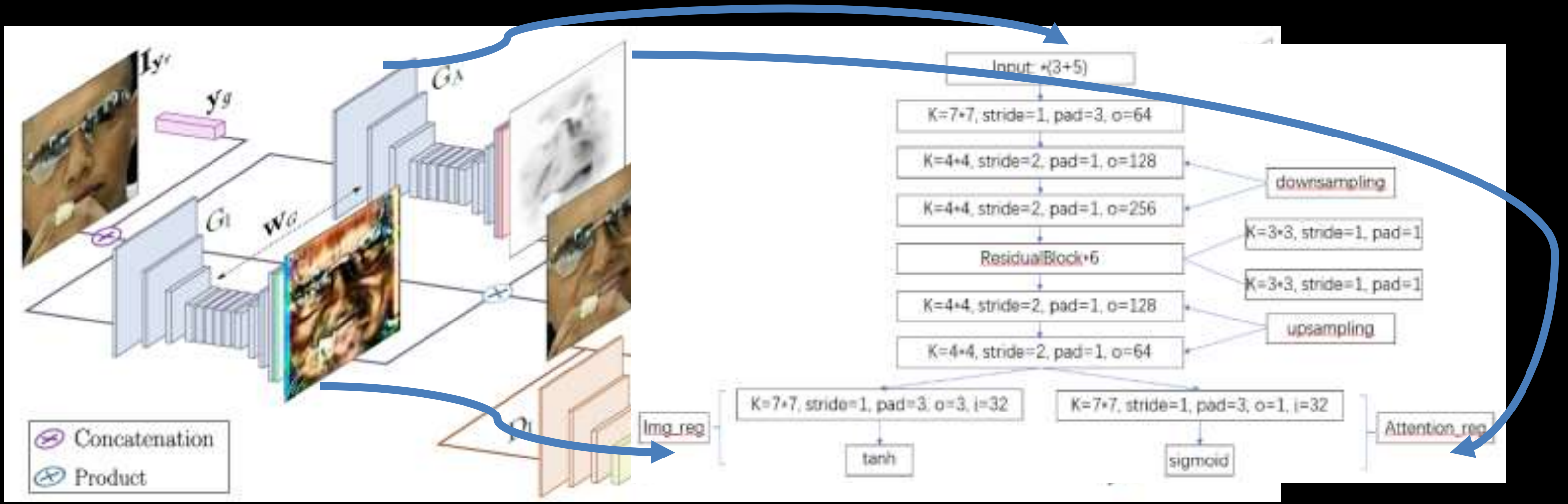
No.2 Pure CG Project

Various Directions	Y
Occlusion	Y
Accurate Annotation	Y
AU Intensity	Continuous
Data Production	Inefficient

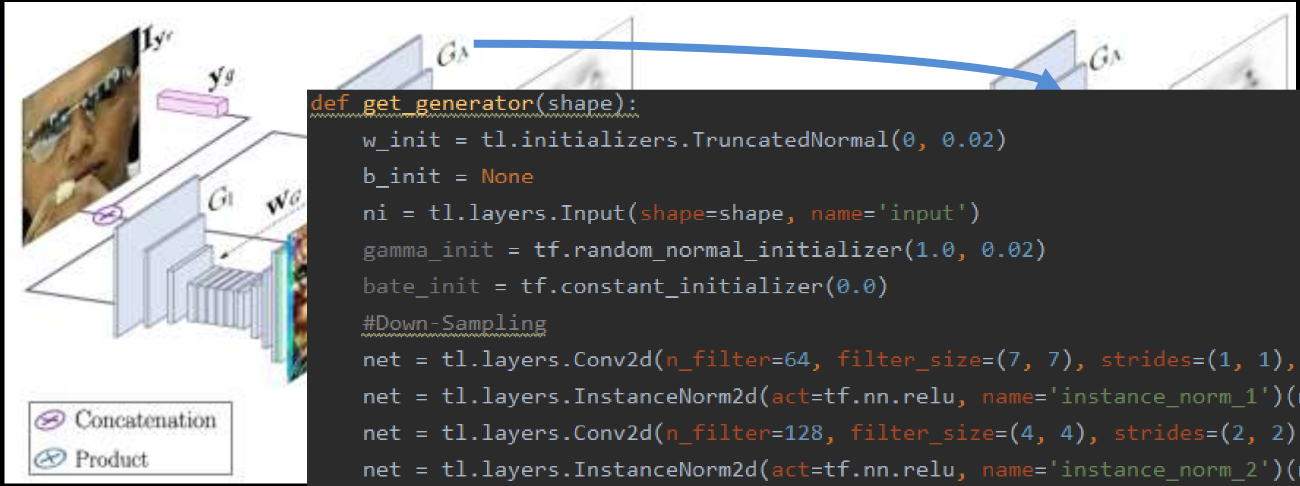
GANimation Architecture



GANimation Architecture

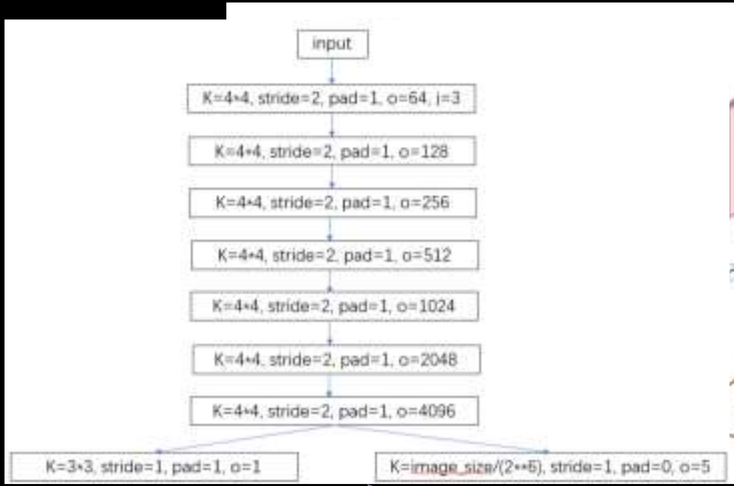


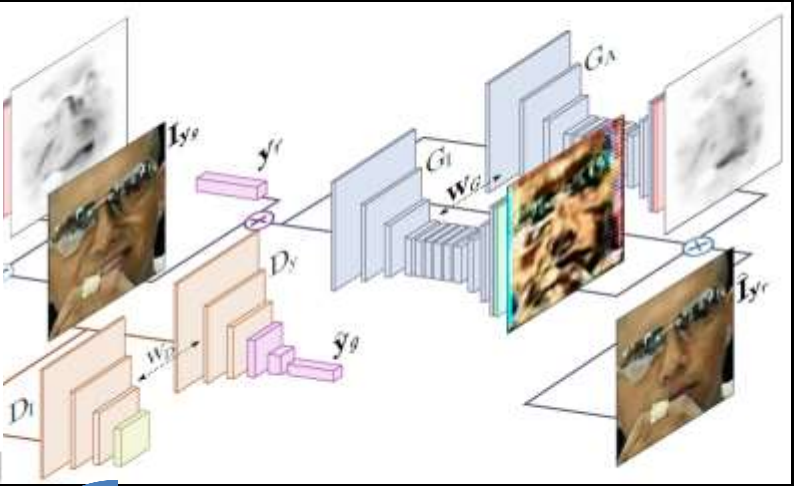
GANimation on TensorLayer - Generator




```
def get_generator(shape):
    w_init = tl.initializers.TruncatedNormal(0, 0.02)
    b_init = None
    ni = tl.layers.Input(shape=shape, name='input')
    gamma_init = tf.random_normal_initializer(1.0, 0.02)
    bate_init = tf.constant_initializer(0.0)
    #Down-Sampling
    net = tl.layers.Conv2d(n_filter=64, filter_size=(7, 7), strides=(1, 1), W_init=w_init, b_init=b_init, padding=
    net = tl.layers.InstanceNorm2d(act=tf.nn.relu, name='instance_norm_1')(net)
    net = tl.layers.Conv2d(n_filter=128, filter_size=(4, 4), strides=(2, 2), W_init=w_init, b_init=b_init, name='c
    net = tl.layers.InstanceNorm2d(act=tf.nn.relu, name='instance_norm_2')(net)
    net = tl.layers.Conv2d(n_filter=256, filter_size=(4, 4), strides=(2, 2), W_init=w_init, b_init=b_init, name='c
    net = tl.layers.InstanceNorm2d(act=tf.nn.relu, name='instance_norm_3')(net)
    #resblock
    for i in range(1, 7):...
    #Up-Sampling
    net = tl.layers.DeConv2d(n_filter=128, filter_size=(4, 4), strides=(2, 2), W_init=w_init, name='deconv1')(net)
    net = tl.layers.InstanceNorm2d(act=tf.nn.relu, name='instance_norm_4')(net)
    net = tl.layers.DeConv2d(n_filter=64, filter_size=(4, 4), strides=(2, 2), W_init=w_init, name='deconv2')(net)
    net = tl.layers.InstanceNorm2d(act=tf.nn.relu, name='instance_norm_5')(net)
```

GANimation Architecture



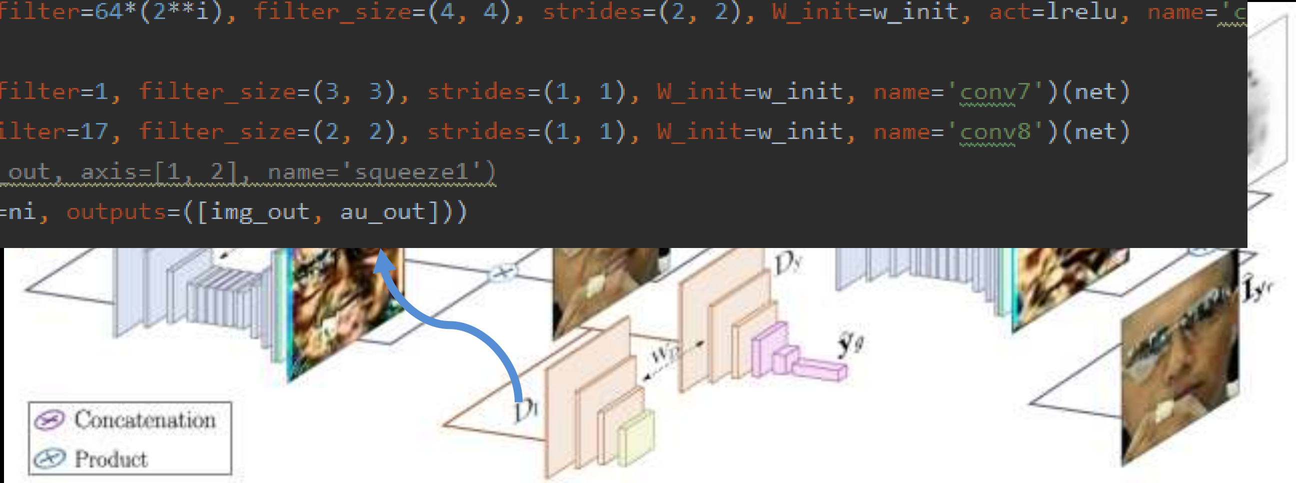


GANimation on TensorLayer - Discriminator

```
def get_discriminator(input_shape):
    w_init = tl.initializers.TruncatedNormal(0, 0.02)
    lrelu = lambda x: tf.nn.leaky_relu(x, 0.01)

    ni = tl.layers.Input(input_shape, name='input')
    net = ni
    for i in range(6):
        net = tl.layers.Conv2d(n_filter=64*(2**i), filter_size=(4, 4), strides=(2, 2), W_init=w_init, act=lrelu, name='c

    img_out = tl.layers.Conv2d(n_filter=1, filter_size=(3, 3), strides=(1, 1), W_init=w_init, name='conv7')(net)
    au_out = tl.layers.Conv2d(n_filter=17, filter_size=(2, 2), strides=(1, 1), W_init=w_init, name='conv8')(net)
    #au_out = tf.squeeze(input=au_out, axis=[1, 2], name='squeeze1')
    return tl.models.Model(inputs=ni, outputs=([img_out, au_out]))
```



Loss Function

Goal of Generator: Generating image of a same ID with given expression.

Goal of Discriminator: Evaluate the quality of the generated image (Real Photo? Given ID?) and its expression (Given tag?)

Gradient penalty

1 Image Adversarial Loss

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] - \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(\mathbf{I}_{\mathbf{y}_o})] + \lambda_{\mathrm{gp}}\mathbb{E}_{\tilde{I}\sim\mathbb{P}_{\tilde{I}}}\left[(\|\nabla_{\tilde{I}}D_{\mathrm{I}}(\tilde{I})\|_2 - 1)^2\right]$$

2 Attention Loss

$$\lambda_{\mathrm{TV}}\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\sum_{i,j}^{H,W}\left[(\mathbf{A}_{i+1,j}-\mathbf{A}_{i,j})^2+(\mathbf{A}_{i,j+1}-\mathbf{A}_{i,j})^2\right]\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[\|\mathbf{A}\|_2] \quad (2)$$

3 Conditional Expression Loss

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathbf{y}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] - \mathbf{y}_f\|_2^2\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathbf{y}}(\mathbf{I}_{\mathbf{y}_o}) - \mathbf{y}_o\|_2^2\right]. \quad (3)$$

4 Identity Loss

$$\mathcal{L}_{\mathrm{idt}}(G,\mathbf{I}_{\mathbf{y}_o},\mathbf{y}_o,\mathbf{y}_f) = \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[\|G(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)|\mathbf{y}_o) - \mathbf{I}_{\mathbf{y}_o}\|_1]. \quad (4)$$

Full Loss

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\mathrm{I}}(G, D_{\mathrm{I}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) + \lambda_{\mathrm{y}}\mathcal{L}_{\mathrm{y}}(G, D_{\mathrm{y}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g) \quad (5)\\ &+ \lambda_{\mathrm{A}}\left(\mathcal{L}_{\mathrm{A}}(G, \mathbf{I}_{\mathbf{y}_g}, \mathbf{y}_r) + \mathcal{L}_{\mathrm{A}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g)\right) + \lambda_{\mathrm{idt}}\mathcal{L}_{\mathrm{idt}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g).\end{aligned}$$

GANimation on TensorLayer - Loss

Loss of Generator

```
loss_g_fake_img_masked = -tf.reduce_mean(pred_fake_img_masked) * lambda_D_img
loss_g_fake_au = l2_loss(desired_au, pred_fake_au) * lambda_D_au
loss_g_cyc = l1_loss(real_img, cyc_img_masked) * lambda_cyc

loss_g_mask_fake = tf.reduce_mean(fake_mask) * lambda_mask + smooth_loss(fake_mask) * lambda_mask_smooth
loss_g_mask_cyc = tf.reduce_mean(cyc_mask) * lambda_mask + smooth_loss(cyc_mask) * lambda_mask_smooth

loss_g = loss_g_fake_img_masked + loss_g_fake_au + \
         loss_g_cyc + \
         loss_g_mask_fake + loss_g_mask_cyc
```

GANimation on TensorLayer - Loss

Loss of Discriminator

```
loss_d_img = -tf.reduce_mean(pred_real_img) * lambda_D_img + tf.reduce_mean(
    pred_fake_img_masked) * lambda_D_img
loss_d_au = l2_loss(real_au, pred_real_au) * lambda_D_au

alpha = tf.compat.v1.random_uniform([BATCH_SIZE, 1, 1, 1], minval=0., maxval=1.)
differences = fake_img_masked - real_img
interpolates = real_img + tf.multiply(alpha, differences)
gradients = tf.gradients(D(interpolates, reuse=True), [interpolates])[0]
slopes = tf.sqrt(tf.reduce_sum(tf.square(gradients), axis=1))
gradient_penalty = tf.reduce_mean((slopes - 1.) ** 2)
loss_d_gp = lambda_D_gp * gradient_penalty

loss_d = loss_d_img + loss_d_au + loss_d_gp
```

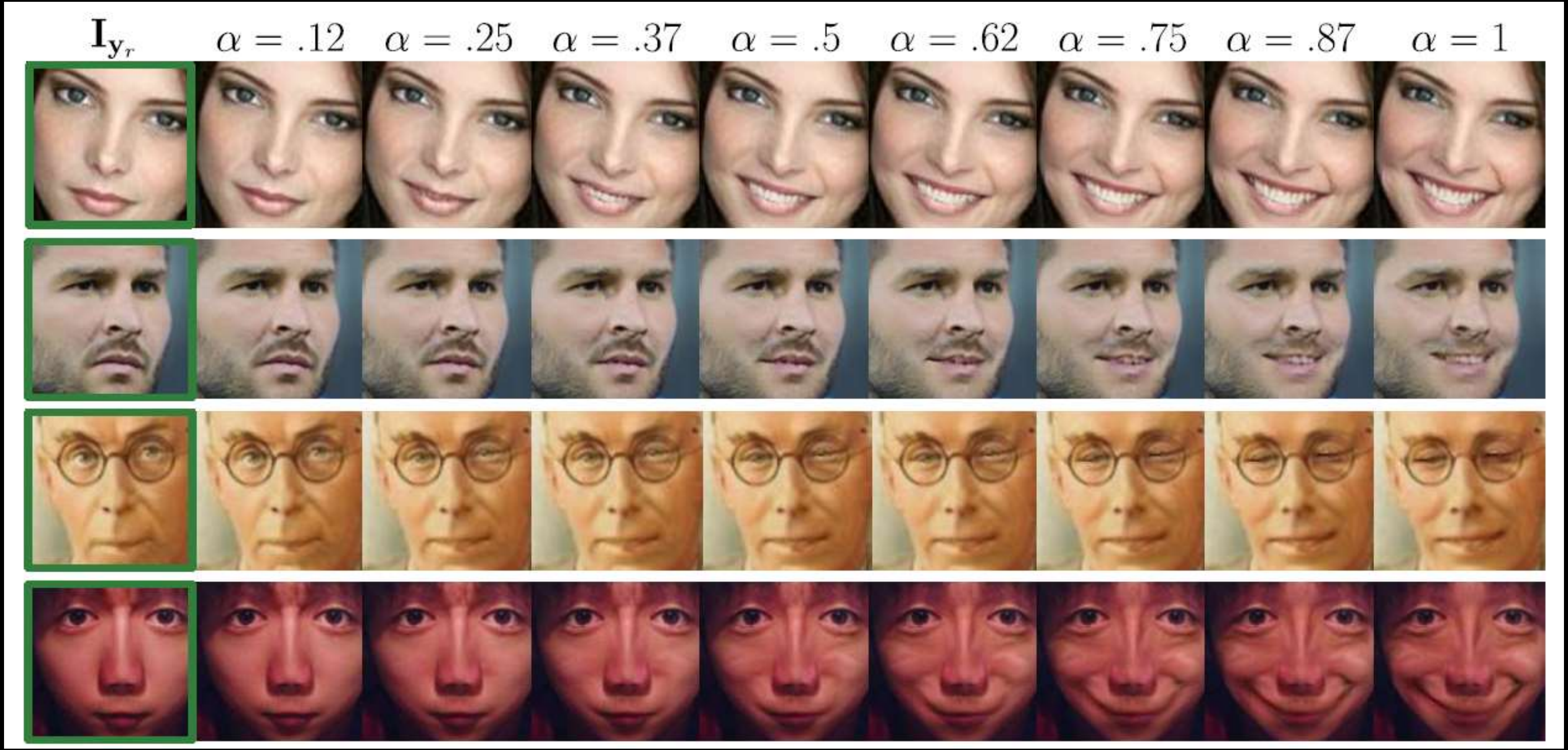
GANimation on TensorLayer - Train

Train

```
BATCH_SIZE = 25
EPOCHS = 30
lambda_D_img = 1
lambda_D_au = 4000
lambda_D_gp = 10
lambda_cyc = 10
lambda_mask = 0.1
lambda_mask_smooth = 1e-5
if e <= 21:
    lr_now = 1e-4
else:
    lr_now = 1e-5 * (EPOCHS + 1 - e)
```

Results – Single AUs' Masks

Results – Multiple AUs



Thanks.