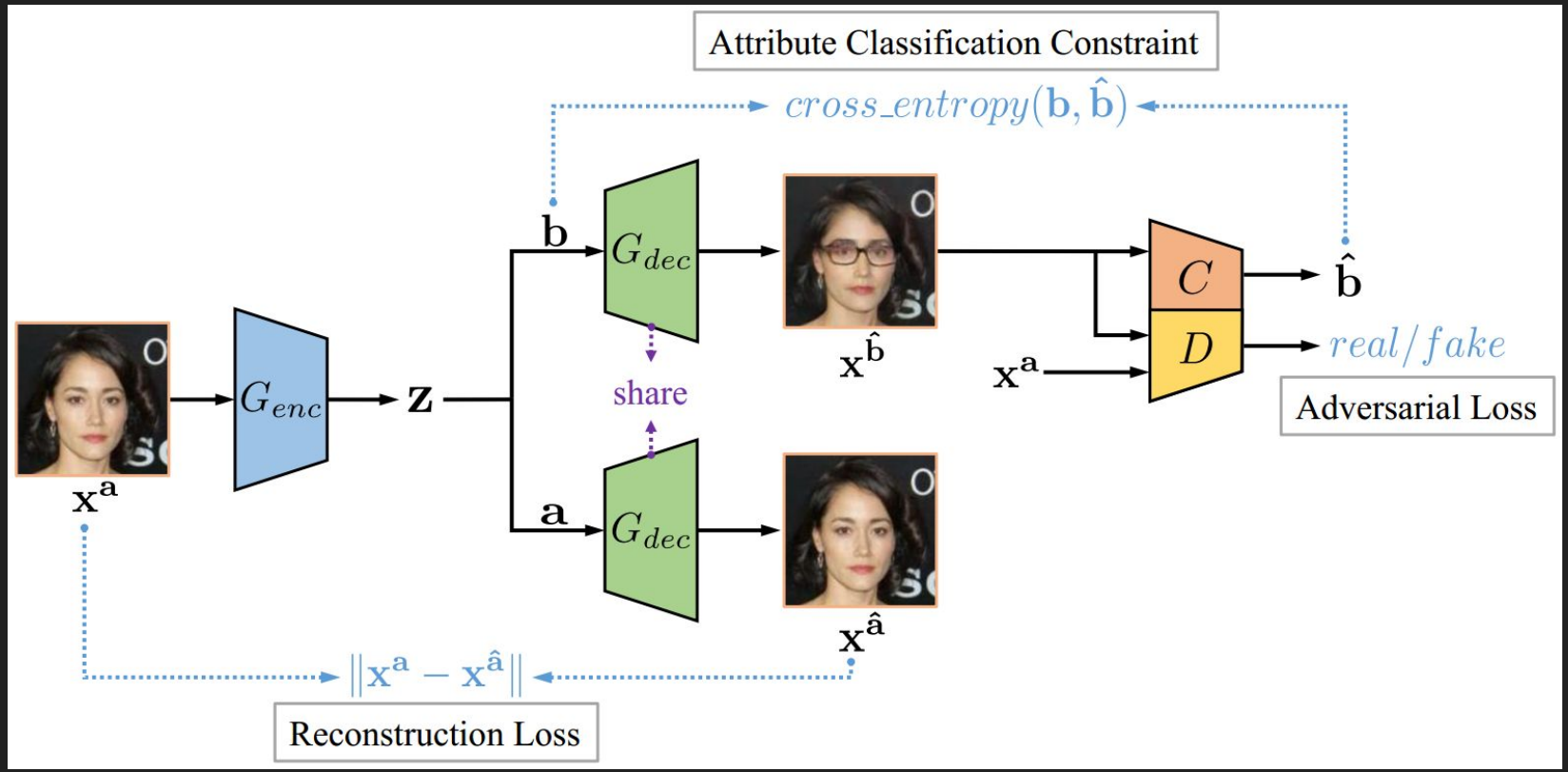
Facial Attribute Editing using Neural Networks

Generative Adversarial Networks

Contents

- Intro to Model and How it works
- Training process
- Loss Metrics
- Graphs
- Generated Samples

Model


Attribute Classification Constraint
$cross\_entropy(\mathbf{b}, \hat{\mathbf{b}})$
$\mathbf{x}^{\mathbf{a}}$
G_{enc}
$\mathbf{z}$
$\mathbf{b}$
G_{dec}
$\mathbf{x}^{\hat{\mathbf{b}}}$
share
$\mathbf{a}$
G_{dec}
$\mathbf{x}^{\hat{\mathbf{a}}}$
$\mathbf{x}^{\mathbf{a}}$
C
D
$\hat{\mathbf{b}}$
$real/fake$
Adversarial Loss
$\|\mathbf{x}^{\mathbf{a}} - \mathbf{x}^{\hat{\mathbf{a}}}\|$
Reconstruction Loss

Intro

- *Networks:*
  - G-enc
    - Encodes the Image to Latent space(Z)
  - G-dec
    - Decodes the Image to original Image Conditioned on specified attributes
  - C & D
    - Attribute Classification constraint works on logits captured at end Layer
    - Real/Fake Detector
- *Loss functions*
  - Reconstruction Loss which preserves the original attributes
  - Adversarial Loss for Real like Sample (WGAN)
  - Cross entropy for Attribute Classification

Training Process

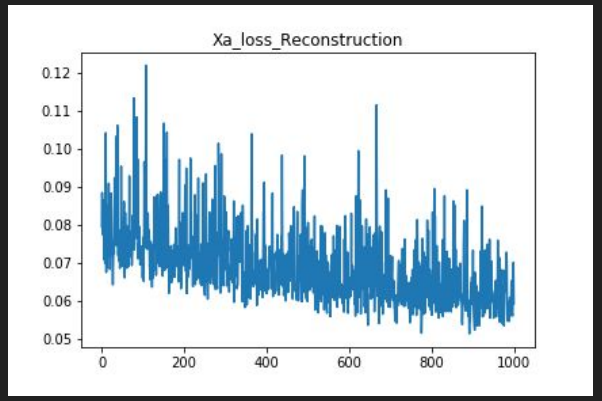
- Load the batch of images (Image[?,128,128,3]+Attribute[?,len(Desired_Attributes)])
- Generate Valid Random Attribute vectors
- Main Three Steps
  - Generate samples for Original and Random Attribute
    - $Generated\_image_{original}$=Gdec{Genc(Xa),original Attribute vector}
    - $Generated\_image_{random}$=Gdec{Genc(Xa),random Attribute vector}
  - Discriminate samples for Original and Random Attribute
    - Xa_logits_{gan},Xa_logits_{attr}=D(Xa) , C(Xa)
    - Xb_logits_{gan},Xb_logits_{attr}=D($Generated\_image_{random}$) , C($Generated\_image_{random}$)
  - Calculate Reconstruction loss
- Define Loss functions Three steps
  - D-loss = (Xa_logits_{gan} - Xb_logits_{gan}) +10.0*GP+Xa_loss_{attr} (More in Next Slide)
  - G-loss = Avg(Xb_logits_{gan})+10.0*Xb_loss_{attr}+100*Reconstruction_loss (More in next Slide)
- Define Optimizers (Adam)
  - D-step = tf.train.AdamOptimizer(lr, beta1=0.5).minimize(d_loss, var_list=d_var)
  - G-step = tf.train.AdamOptimizer(lr, beta1=0.5).minimize(g_loss, var_list=g_var,global_step=global_step)
- Run for epoch in n_epochs:
  - Run for batch in n_batches:
    - Run D-step : G_step = 5 : 1

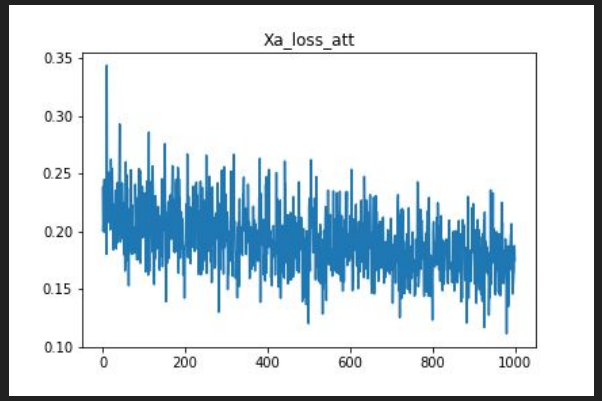
Xa: Original Image

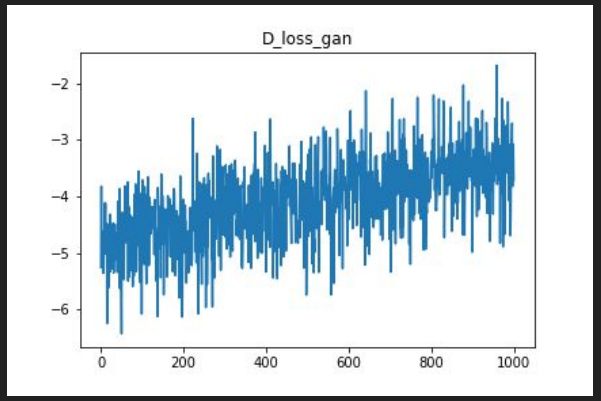
Loss Metrics

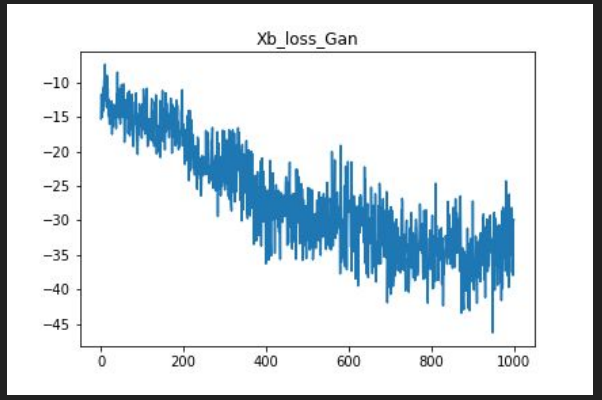
- WGAN-GP
  - $L_D^{WGAN} = E[D(Xa)] - E[D(Generated\_image_{random})]$
  - $L_G^{WGAN} = E[D(Generated\_image_{random})]$
  - Gradient Penalty $= \lambda * (\|\nabla_{\boldsymbol{x}}(D_{w}(\boldsymbol{x}))\|_2 - 1)^2$
  - Total_loss = $L_D^{WGAN} + L_G^{WGAN}$ + Gradient Penalty
- Xa_loss_{attr} : tf.losses.sigmoid_cross_entropy(a, Xa_logit_{attr})
- Xb_loss_{attr} : tf.losses.sigmoid_cross_entropy(a, Xa_logit_{attr})

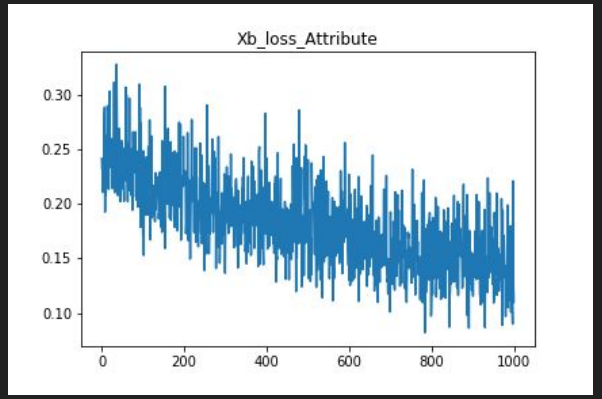
Graphs


Xa_loss_Reconstruction
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0
200
400
600
800
1000

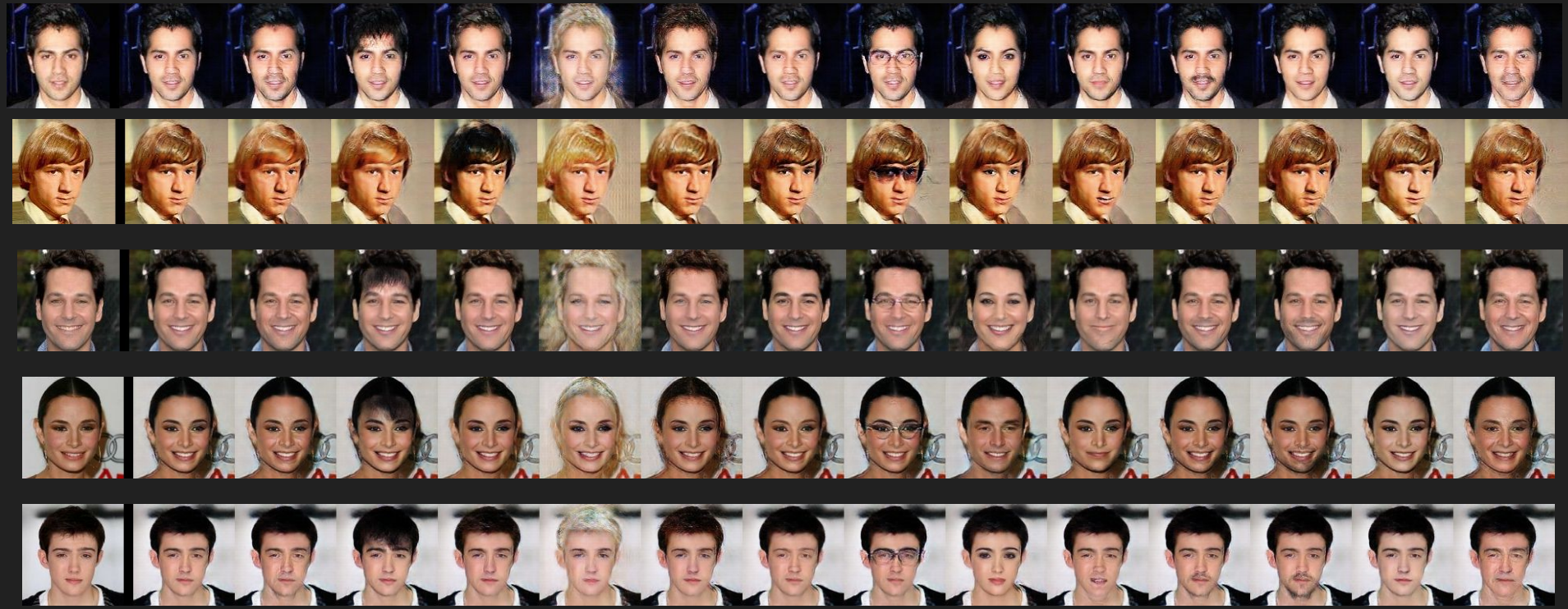

Xa_loss_att
0.35
0.30
0.25
0.20
0.15
0.10
0
200
400
600
800
1000


D_loss_gan
−2
−3
−4
−5
−6
0
200
400
600
800
1000


Xb_loss_Gan
−10
−15
−20
−25
−30
−35
−40
−45
0
200
400
600
800
1000


Xb_loss_Attribute
0.30
0.25
0.20
0.15
0.10
0
200
400
600
800
1000

Generated Samples



Original | Original | Bangs | Bald | Black Hair | Blond hair | BrownHair | Bushy EyeBrows | Spects | Gender | Mouth open (Slight) | Mustache | Beard | Pale Skin | Age

End