Convert Male portraits to female and vice-versa

Alshammari Amal Salem S (아말) - M2016719
Alghamdi Amjad Ahmed A (암자드) - M2016721

Convert Male portraits to female and vice-versa

The purpose of this Machine-learning project is to show that it's possible to automatically transform pictures of faces in useful and fun ways. The way this is done is by filtering the type of faces used as inputs to the model and the types of faces used as the desired target. The exact same architecture can be used to transform masculine faces into feminine ones, or vice versa, just by switching the source and target images used during training.

- Requirements:
    i. Python (3.5+)
    ii. TensorFlow (r0.12+)
    iii. GPU (optional for faster training)

-Face Recognition

Training examples of a Image

Used with permission.

It also has other potential applications, such as vanity filters that make people look more attractive. This would be done by selecting only attractive faces as the target population. More experimentation will be required.

How it works :

Here is what we learned in this project. About Machine Learning and many other functionality. Read about Machine Learning on internet.

* Tune the architecture to the nature of the problem

This project takes 80x100 pixel images as inputs and produces images of the same size as outputs. In addition to that both of those images are faces, which means that the input and output distributions are very similar.

It would be possible to do the same in this project by progressively encoding the 80x100 pixel input image into a 1x1 latent embedding and later expanding it to an image again. However, since we know that the input and output distributions are very similar we don't need to encode the input image all the way down into 1x1 pixels.

In the final architecture the encoder only has two pooling layers. Increasing the number of layers actually lowered the quality of the outputs in the sense that they no longer resembled the person in the source image. The goal was to produce an output that was clearly recognizable as the same person, and that necessarily requires making relatively small changes to the source material.

Train Model

Training the model for about 60 to 80 batches. Need to adjust “train time” depending on how many batches/hour your system can train.

-Training the model *:*

```
import numpy as np
import os.path
import tensorflow as tf
import time

import dm_arch
import dm_input
import dm_utils

FLAGS = tf.app.flags.FLAGS
```

```
def _save_image(train_data, feature, gene_output, batch, suffix, max_samples=None):
    """Saves a picture showing the current progress of the model"""

    if max_samples is None:
        max_samples = int(feature.shape[0])
    td  = train_data

    clipped  = np.clip(gene_output, 0, 1)
    image    = np.concatenate([feature, clipped], 2)

    image    = image[:max_samples,:,:,:]
    cols     = []
    num_cols = 4
samples_per_col = max_samples//num_cols

    for c in range(num_cols):
        col   = np.concatenate([image[samples_per_col*c + i,:,:,:] for i in range(samples_per_col)], 0)
        cols.append(col)

    image   = np.concatenate(cols, 1)

    filename = 'batch%06d_%s.png' % (batch, suffix)
    filename = os.path.join(FLAGS.train_dir, filename)

    dm_utils.save_image(image, filename)


def _save_checkpoint(train_data, batch):
    """Saves a checkpoint of the model which can later be restored"""
    td = train_data

    oldname = 'checkpoint_old.txt'
    newname = 'checkpoint_new.txt'

    oldname = os.path.join(FLAGS.checkpoint_dir, oldname)
    newname = os.path.join(FLAGS.checkpoint_dir, newname)

    # Delete oldest checkpoint
    try:
        tf.gfile.Remove(oldname)
        tf.gfile.Remove(oldname + '.meta')
    except:   pass
 # Rename old checkpoint
    try:
        tf.gfile.Rename(newname, oldname)
        tf.gfile.Rename(newname + '.meta', oldname + '.meta')
    except:
        pass

    # Generate new checkpoint
    saver = tf.train.Saver()
    saver.save(td.sess, newname)

    print("    Checkpoint saved")
def train_model(train_data):
```

```
    """Trains the given model with the given dataset"""
    td  = train_data
    tda = td.train_model
    tde = td.test_model

    dm_arch.enable_training(True)
    dm_arch.initialize_variables(td.sess)

    # Train the model
    minimize_ops = [tda.gene_minimize, tda.disc_minimize]
    show_ops     = [td.annealing, tda.gene_loss, tda.disc_loss, tda.disc_real_loss, tda.disc_fake_loss]

    start_time = time.time()
    step       = 0
    done       = False
    gene_decor = " "


print('\nModel training...')
    step = 0
    while not done:
        # Show progress with test features
        if step % FLAGS.summary_period == 0:
            feature, gene_mout = td.sess.run([tde.source_images, tde.gene_out])
            _save_image(td, feature, gene_mout, step, 'out')

        # Compute losses and show that we are alive
        annealing, gene_loss, disc_loss, disc_real_loss, disc_fake_loss = td.sess.run(show_ops)
        elapsed   = int(time.time() - start_time)/60
        print('  Progress[%3d%%], ETA[%4dm], Step [%5d], temp[%3.3f], %sgene[%-3.3f], *disc[%-3.3f] real[%-3.3f]
fake[%-3.3f]' %
              (int(100*elapsed/FLAGS.train_time), FLAGS.train_time - elapsed, step,
               annealing, gene_decor, gene_loss, disc_loss, disc_real_loss, disc_fake_loss))

        # Tight loop to maximize CPU utilization
        if step < 200:

            gene_decor = " "
            for _ in range(10):
                td.sess.run(tda.disc_minimize)
        else:

            gene_decor = "*"
            for _ in range(2):
                td.sess.run(minimize_ops)
                td.sess.run(tda.disc_minimize)
                td.sess.run(tda.disc_minimize)
                td.sess.run(tda.disc_minimize)
        step += 1
# Finished?
        current_progress = elapsed / FLAGS.train_time
        if current_progress >= 1.0:
            done = True

        # Decrease annealing temperature exponentially
```

```
    if step % FLAGS.annealing_half_life == 0:
        td.sess.run(td.halve_annealing)

    # Save checkpoint
    #if step % FLAGS.checkpoint_period == 0:
    #    _save_checkpoint(td, step)

_save_checkpoint(td, step)
print('Finished training!')
```

- ***Uniqueness***

Using in this project the following:

1. Decision Theory
2. Search & Optimization
3. Decision processes
4. Deep Learning
5. Machine Learning – Classification
6. Machine Learning – Clustering
7. Natural Language Processing

- **Machine Learning Problems:**

A. *Choosing the Training Experience*
   a. Choosing the Training Experience
      i. Sometimes straightforward
         1. Text classification, disease diagnosis
      ii. Sometimes not so straightforward
         1. Chess playing
   b. Other Attributes
      i. How the training experience is controlled by the learner?
      ii. How the training experience represents the situations in which the performance of the program is measured?

B. *Choosing the Target Function*
   C. Choosing the Target Function
      a. What type of knowledge will be learned?
      b. How it will be used by the program?
   D. Reducing the Learning Problem
      a. From the problem of improving performance P at task T with experience E
      b. To the problem of learning some particular target functions

E. *Solving Real World Problems*

F. What Is the Input?
    a. Features representing the real world data
G. What Is the Output?
    a. Predictions or decisions to be made
H. What Is the Intelligent Program?
    a. Types of classifiers, value functions, etc.
I. How to Learn from experience?
    a. Learning algorithms

J. *Feature Engineering*
    K. Representation of the Real World Data
        a. Features: data's attributes which may be useful in prediction
    L. Feature Transformation and Selection
        a. Select a subset of the features
        b. Construct new features, e.g.
            i. Discretization of real value features
            ii. Combinations of existing features
    M. Post Processing to Fit the Classifier
        a. Does not change the nature

N. Intelligent Programs
    O. Value Functions
        a. Input: features
        b. Output: value
    P. Classifiers (Most Commonly Used)
        a. Input: features
        b. Output: a single decision
    Q. Sequence Labeling
        a. Input: sequence of features
        b. Output: sequence of decisions