Train a Variational Autoencoder (VAE) using perceptual loss

trains the VAE for 50000 steps using perceptual loss and the 20170512-110547 model. The model is trained on the CASIA-WebFace dataset

1.训练 VAE train_vae.py

```
python src/generative/train_vae.py \
src.generative.models.dfc_vae \
~/datasets/casia/casia_maxpy_mtcnnpy_182 \
src.models.inception_resnet_v1 \
~/models/export/20170512-110547/model-20170512-110547.ckpt-250000 \
--models_base_dir ~/vae/ \
--reconstruction_loss_type PERCEPTUAL \
--loss_features 'Conv2d_1a_3x3,Conv2d_2a_3x3,Conv2d_2b_3x3' \
--max_nrof_steps 50000 \
--batch_size 128 \
--latent_var_size 100 \
--initial_learning_rate 0.0002 \
--alfa 1.0 \
--beta 0.5
```

1 VAE 模型的名字 dfc_vae

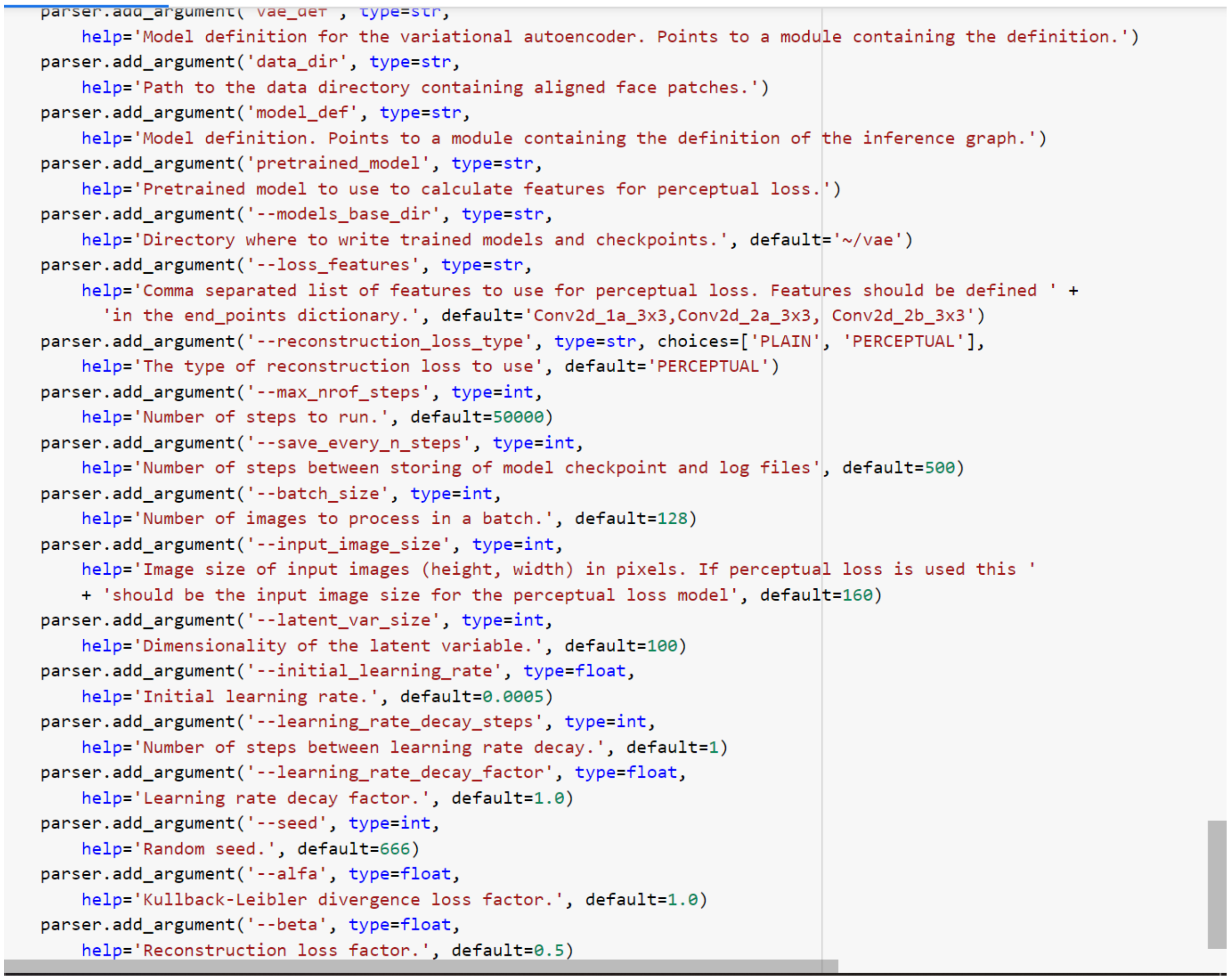
2.训练用的数据集的地址

3.restnet of the inference graph

4.导入 pretrained 的模型节点到 dfc_vae（需要训练为什么还要 pretrained 的 VAE?因为需要用于计算 features for p loss）

5.训练好的模型放的地址

6.其他设置



```
parser.add_argument('vae_def', type=str,
    help='Model definition for the variational autoencoder. Points to a module containing the definition.')
parser.add_argument('data_dir', type=str,
    help='Path to the data directory containing aligned face patches.')
parser.add_argument('model_def', type=str,
    help='Model definition. Points to a module containing the definition of the inference graph.')
parser.add_argument('pretrained_model', type=str,
    help='Pretrained model to use to calculate features for perceptual loss.')
parser.add_argument('--models_base_dir', type=str,
    help='Directory where to write trained models and checkpoints.', default='~/vae')
parser.add_argument('--loss_features', type=str,
    help='Comma separated list of features to use for perceptual loss. Features should be defined ' +
      'in the end_points dictionary.', default='Conv2d_1a_3x3,Conv2d_2a_3x3, Conv2d_2b_3x3')
parser.add_argument('--reconstruction_loss_type', type=str, choices=['PLAIN', 'PERCEPTUAL'],
    help='The type of reconstruction loss to use', default='PERCEPTUAL')
parser.add_argument('--max_nrof_steps', type=int,
    help='Number of steps to run.', default=50000)
parser.add_argument('--save_every_n_steps', type=int,
    help='Number of steps between storing of model checkpoint and log files', default=500)
parser.add_argument('--batch_size', type=int,
    help='Number of images to process in a batch.', default=128)
parser.add_argument('--input_image_size', type=int,
    help='Image size of input images (height, width) in pixels. If perceptual loss is used this '
    + 'should be the input image size for the perceptual loss model', default=160)
parser.add_argument('--latent_var_size', type=int,
    help='Dimensionality of the latent variable.', default=100)
parser.add_argument('--initial_learning_rate', type=float,
    help='Initial learning rate.', default=0.0005)
parser.add_argument('--learning_rate_decay_steps', type=int,
    help='Number of steps between learning rate decay.', default=1)
parser.add_argument('--learning_rate_decay_factor', type=float,
    help='Learning rate decay factor.', default=1.0)
parser.add_argument('--seed', type=int,
    help='Random seed.', default=666)
parser.add_argument('--alfa', type=float,
    help='Kullback-Leibler divergence loss factor.', default=1.0)
parser.add_argument('--beta', type=float,
    help='Reconstruction loss factor.', default=0.5)
```

2. Calculate attribute vectors

```
python src/generative/calculate_attribute_vectors.py \
src.generative.models.dfc_vae \
~/vae/20170708-150701/model.ckpt-50000 \
~/datasets/celeba/img_celeba \
~/datasets/celeba/list_attr_celeba.txt \
~/vae/20170708-150701/attribute_vectors.h5 \
--batch_size 128 \
--image_size 160 \
--latent_var_size 100
```

1. VAE 的名称
2. 第一步训练的节点
3. Dataset 的地址
4. 输出的 Attribute 的名字
5. 输出的 attribute vector 文件地址

```
def parse_arguments(argv):
    parser = argparse.ArgumentParser()

    parser.add_argument('vae_def', type=str,
        help='Model definition for the variational autoencoder. Points to a module containing the definition.',
        default='src.generative.models.dfc_vae')
    parser.add_argument('vae_checkpoint', type=str,
        help='Checkpoint file of a pre-trained variational autoencoder.')
    parser.add_argument('data_dir', type=str,
        help='Path to the directory containing aligned face patches for the CelebA dataset.')
    parser.add_argument('annotations_filename', type=str,
        help='Path to the annotations file',
        default='/media/deep/datasets/CelebA/Anno/list_attr_celeba.txt')
    parser.add_argument('output_filename', type=str,
        help='Filename to use for the file containing the attribute vectors.')
    parser.add_argument('--batch_size', type=int,
        help='Number of images to process in a batch.', default=128)
    parser.add_argument('--image_size', type=int,
        help='Image size (height, width) in pixels.', default=64)
    parser.add_argument('--latent_var_size', type=int,
        help='Dimensionality of the latent variable.', default=100)
    parser.add_argument('--seed', type=int,
        help='Random seed.', default=666)

    return parser.parse_args(argv)
```

3. Add smile to a face

```
python src/generative/modify_attribute.py \
src.generative.models.dfc_vae \
~/vae/20170708-150701/model.ckpt-50000 \
~/vae/20170708-150701/attribute_vectors.h5 \
~/vae/20170708-150701/add_smile.png
```

1. 打开 VAE
2. 打开第一步的节点
3. 打开第二步的 vector
4. 添加自己想添加的 attribute 到自己想添加的图片中

```
def parse_arguments(argv):
    parser = argparse.ArgumentParser()

    parser.add_argument('vae_def', type=str,
        help='Model definition for the variational autoencoder. Points to a module containing the definition.')
    parser.add_argument('vae_checkpoint', type=str,
        help='Checkpoint file of a pre-trained variational autoencoder.')
    parser.add_argument('attributes_filename', type=str,
        help='The file containing the attribute vectors, as generated by calculate_attribute_vectors.py.')
    parser.add_argument('output_image_filename', type=str,
        help='File to write the generated image to.')
    parser.add_argument('--latent_var_size', type=int,
        help='Dimensionality of the latent variable.', default=100)
    parser.add_argument('--seed', type=int,
        help='Random seed.', default=666)

    return parser.parse_args(argv)
```