Laten Explorer Documentation

Introduction

The goal of this project is to create a pipeline which uses StyleGAN and SVM models to find latent directions that represent “pain” in human faces latent space. This pipeline involves creating a dataset of embedded human face images and their pain classifications, train a SVM classifier to find the “pain” directions in the latent space of StyleGAN human faces, create GAN inversion of the original image while moving the embedding to the desired direction causing the output image to be edited with more or less “pain”.

This project uses code from the articles:

- For the GAN inversion we found that the best inversion method can be implemented using the code from the PTI article:
  PTI: Pivotal Tuning for Latent-based editing of Real Images **.**
- For the SVM boundaries classification (train_boundery API) we used ***InterFaceGAN - Interpreting the Latent Space of GANs for Semantic Face Editing***

Other useful articles on the subject:

[1] Patrick Lucey et al. 2011: PAINFUL DATA: The UNBC-McMaster Shoulder Pain Expression Archive Database

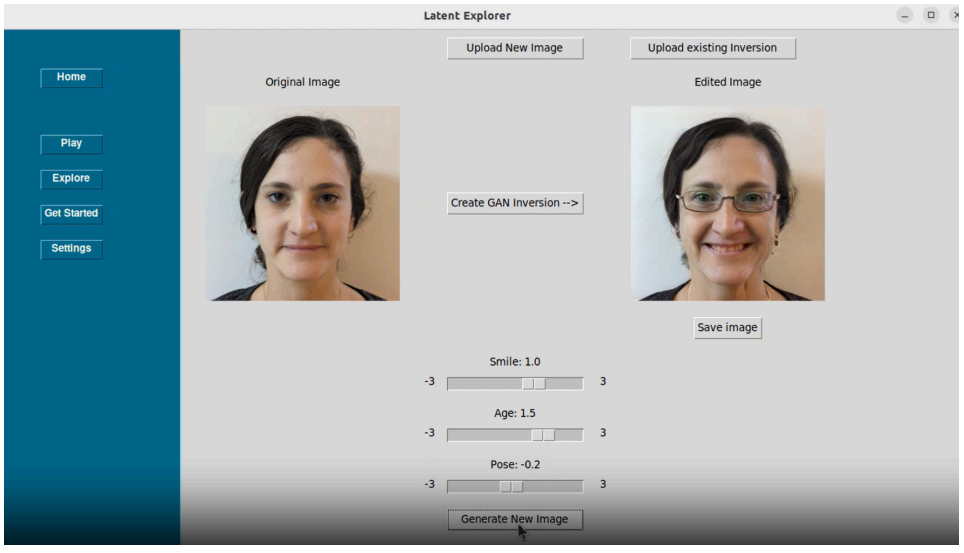
[2] Boneh-Shitrit+, M. Feigelstein, A. Bremhorst,+, S. Amir, T. Distelfeld, Y. Dassa, S.Yaroshevsky, S.Riemer, I. Shimshoni, DS. Mills, and A. Zamansky: “Explainable automated recognition of emotional canine facial expressions: The case of the Positive Anticipation and Frustration. Scientific Reports 12, 22611 Q1, IF: 4.996 (2022).

Module Description

1. GUI and API for Executing the PTI model - “Play” page
    a. **Introduction**

    This module brings a nice GUI to experience, play and test the tweaking and editing of real images with the trained latent direction using the PTI architecture. This feature is deployed on the “Play” page in the tkinter app we named “Latent Explorer”. The pipeline of this page is:

    i. Upload an image of human faces.
    ii. Generate GAN inversion of the uploaded image. This process uses the PTI architecture to train a new generator from a pre-trained one specifically for the current image.
    iii. Choose directions in the latent space and use the scroll bars to set their values - currently supported smile, age, pose.
    iv. Generate the edited image using the new generator model.



    b. **Code Location**

    The app frontend is located in the folder *latent_explorer/gui/*

    The app backend is located in the folder *latent_explorer/api/*

The app database with the direction tensors and model weights is located in the folder
latent_explorer/database/

To run the app locally just execute the following script
latent_explorer/gui/main_app.py

2. Code for creating and preparing the dataset of pain images.
    a. **Introduction**

        This module is the dataset preparation file, It contains several parts, each performs another step in preparing the dataset to be able to run the SVM on it and get the new direction.

        This file was made specifically for the "Shoulder pain dataset" and will need to be modified to fit any other dataset.

        the "Shoulder pain dataset" structure: 40 subjects, each has a few videos saved as separate "frames". Each frame has a txt file containing the facs score for that frame.

        The SVM is designed to take inputs in the form of 2 vectors: a vector of the latent encoding for each image and a vector with corresponding scores.

    b. the steps:

        1.create json file with "patient_name", "video_name","frame" and pspi_score" keys

        2. add "facs score" key

        3. add "pain" key

        4. split to train validation and test and add statistics

        5. copy the images to 3 folders based on the split

        6. Create vectors - after the embedding of the images in the folders, create the 2 output vectors of latent codes and score. This exist also as an API.

    c. **Code Location**

        latent_explorer/database/dataset_prepration/data_proccecing.ipynb

    d. **Call**

        To prepare the “shoulder pain dataset” you will need to copy the images and the “images” and “Frame labels” directory to where you run the file. Then run through steps 1-5. Then you will need to use the API of the embeddings calculation. Only once you have the embeddings you can run section 6 and get the 2 vectors you need to use for the new direction API.

3. API for calculate our dataset embeddings
    a. **Introduction**

        This module takes a folder with input face images and returns a folder with the embedded images represented as tensors in the latent space. The process is:

        i. Preprocess the images - align and crop each image
        ii. Create Dataloader of the processed images
        iii. Load the pretrained Generator
        iv. Loop over the images and embed each image using PTI projector.

    b. **Code Location**

        latent_explorer/api/explore/calc_inversion.py

    c. **API Call**

        To embed all images into latent vectors, you need to call -

        calc_inversiones(
        input_images_dir *str,*
        processed_images_dir *str,*
        embbeding_images_dir *str*
)

        Input Variables -

        Input_images_dir - path to local folder which holds the original images dataset.
        processed_images_dir - path to local folder which will hold the processed images dataset.
        embbeding_images_dir- path to local folder which will hold the embedding vectors dataset.

4. API for converting embedded images and scores into 2 vectors with latent codes and scores
    a. **Introduction**

        This module takes the embedded images and their scores and creates 2 vectors: one is a vector of all the latent codes and the second is the corresponding scores.

    b. **Code Location**

        latent_explorer/api/explore/create_vectors.py

    c. **API Call**

        create_vectors (

        Embeddings_folder *str,*

        Frame_labels_json_file_path *str,*

        output_dir *str*

)

        Input Variables -

        Embeddings_folder - path to local folder containing the embedded images in .pt format.
        Frame_labels_json_file_path - path to local .json file which holds dataset information including a “frame” key that corresponds to each image name and for each frame a “score” key. In this case the score is a “pain” key and is binary, but can be otherwise in other datasets and need to be adjusted according to the data.
        Output_dir - where you want the output vectors to be saved

5. API to train a new direction in the latent space according to the embeddings and their classification.

    a. **Introduction**

    This module finds the orthogonal direction of a SVM classifier in a latent space of embedded images and saves this direction as a pytorch tensor. This direction tensor can be later used to edit images with the PTI GUI that we've created.

    b. **Code Location**

    latent_explorer/api/explore/train_boundary.py

    c. **API Call**

    To train the SVM classifier and save the directions as tensor you need to run the following module -

    train_and_save_boundaries(
    latent_code_path *str,*
    latent_scores_path *str,*
    boundary_tensor_path *str*
)

    Input Variables -

    latent_code_path - path to local .npy file which holds the embeddings vectors of the images datasets.
    latent_scores_path - path to local .npy file which holds the label vector. This is the classification vector of each embedding vector in the latent code.
    boundary_tensor_path - path to local .pt file which will hold the output direction vector in the latent space.