Predictive BMI with Conditional Adversarial Autoencoder

Alejandro Bravo
abravo16@stanford.edu

Edward Guzman
eguzman3@stanford.edu

John Romano
jromano@stanford.edu

March 2019

Abstract

Generative Adversarial Networks (GAN) and related architectures such as Conditional Adversarial Autoencoders (CAAE) have been widely used for predictive tasks such as facial aging and rejuvenation. In addition to facial aging, such generative images are potentially useful for a number of health applications, such as predictive Body Mass Index (BMI) image generation. However, we were not able to find any models that performed this task. We present a model that uses a CAAE in order to generate facial images at different BMI levels. We achieved a mean BMI difference of 6.44 between generated images and their labels.

1 Introduction

Body Mass Index (BMI) is a "simple, inexpensive, and noninvasive surrogate measure of body fat" that is "an appropriate measure for screening for obesity and its health risks" [1]. It requires only two parameters to calculate (height and weight), and since it measures excess weight rather than excess fat, it has clinical limitations [1]. There has been machine learning work done with BMI, specifically the use of VGG-Face network for regression of facial images to BMI [2]. This work seems to only be limited to regression, however, and not other tasks such as generation. As such, given the success of Generative Adversarial Networks (GAN) and other related architectures such as Conditional Adversarial Autoencoders (CAAE) in tasks such as facial age prediction [3][4], it is of interest to explore generation of facial images at different BMI levels. Such a generator could be useful for medical professionals to provide more complete information and/or projections to their patients. The input to our algorithm are images of human faces across a wide variety of BMI labels. We then use a CAAE to output a series of 10 images at different BMI levels for each input face [5].

2 Related Work

2.1 Face aging and rejuvenation

Face aging and rejuvenation is an exciting application of conditional generative adversarial networks that has grown in popularity with recent advances in natural image generation. Previously, common face aging techniques were categorized into 'prototyping' and 'modeling' methods. Prototyping methods estimate average faces in different age buckets and take the differences between these average faces to find aging patterns, which are then applied to input images to artificially age or rejuvenate them. Such methods do not take into account distinct facial details among individuals and therefore produce unrealistic results [3][4]. Modeling methods tend to produce more realistic images by modeling the biological and physical patterns of aging, focusing on muscle changes, wrinkles, and facial structure. These methods are substantially more costly to implement, however, as they require collecting data from many individuals at various points in their lives. They are also computationally expensive compared to prototyping methods [7].

With recent advances in natural image generation, conditional generative adversarial networks have been utilized more frequently in place of these older methods [3][4]. Such implementations have utilized datasets containing cropped images of faces labeled with the ages of the subjects in each photo. With this data, they have made use of "Identity-Preserving" models, which generate convincing images of faces conditioned on age while also preserving the identity of the individual in the input image. We were inspired by these techniques to explore the possibility of generating altered photos of individuals' faces using variables other than age.

2.2 BMI classification

Our decision to generate images of individuals at varying BMI's was partially inspired by Kocabey et al.'s work in using computer vision techniques to estimate an individual's BMI using only a photo of their face [2]. This approach utilized a dataset containing cropped images of faces labeled by BMI. This dataset format fit well with the identity-preserving methods used in face-aging/rejuvenation studies, so

we decided to pursue a novel approach of using a CAAE to generate altered images of an individual's face, preserving the original identity and conditioned on varying BMIs. A CAAE is similar to a GAN in that it has a generator and discriminator, but it is different in that unlike a GAN, it does not start training from noise but from Z-vectors from the encoder [5].

3 Dataset and Features

3.1 UTKFace Dataset

The CAAE we are using is trained on the publicly-available UTKFace dataset [6]. This is a large-scale single-face dataset with a large range of ages (up to 116 years old). The dataset consists of over 23,000 images covering providing large variation in poses, facial expressions, and ambient lighting. Each image is labeled by age, gender, and ethnicity. More specifically, we are using a variant of the UTKFace dataset that contains images that are cropped and center-aligned around each face. The images in this dataset variant have an RGB color space and a resolution of 200x200.

3.2 VisualBMI Dataset

The BMI dataset we are using was developed using data from the now defunct VisualBMI project and consists of user-submitted images on social media. More specifically, the dataset was created by Kocabey et al. as part of their efforts to build a BMI classifier based on a single face input [2]. Access to this dataset was obtained by reaching out to the team via email and requesting access to the dataset and code described in their paper. This dataset consists of around 4,200 single-face images. The images are RGB images with some variation in lighting and image resolutions. Each image is labeled by BMI and gender. Some preprocessing was required to crop and center-align each face image in the dataset. This was accomplished using the open source Python Autocrop utility [12].

3.3 Data Augmentation

The VisualBMI dataset, while useful, was limited to only a few thousand images. We used a number of data augmentation techniques to multiply our dataset by a factor of 10. For each of the 4206 input images, we generated the following augmentations using the albumentations library in Python [13]:

1. Random Brightness Change
2. Random Contrast Change
3. Grayscale
4. Horizontal Flip
5. Hue-Saturation-Value Color Shift
6. Combination of 1 and 4
7. Combination of 2 and 4
8. Combination of 3 and 4
9. Combination of 4 and 5
10. Combination of 2 and 5

4 Methods

4.1 Initial Approach

We originally had the idea of using a GAN for facial generation and then applying BMI regression on the generated images to explore the accuracy. We began by adapting a GAN implementation in PyTorch to the UTKFace dataset to get a first model up and running [10]. This model worked well on the MNIST dataset that it was designed for, but when we applied it to the UTKFace dataset, we received high discriminator losses and noisy, unintelligible output for many epochs. It was at this point that we decided to pivot to using the CAAE directly for variable BMI facial generation. We used a CAAE this time that was implemented for the purposes of facial aging and were able to get it to produce results, shown below [11].

4.2 Transfer Learning Approach

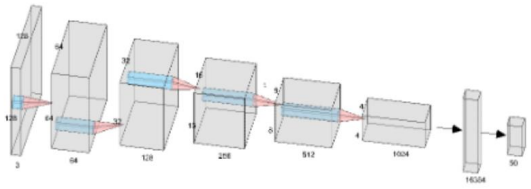
Our baseline model comes from Github user mattans' PyTorch implementation of the algorithm in [11]. This model was written in Python 3.7 and PyTorch 0.4.1. This network consists of an encoder, which transforms RGB images to Z vectors, a generator which transforms Z vectors to RGB images, a discriminator that measures and forces uniform distribution on the encoder's output, and another discriminator that measures and forces realistic properties on the generator's output.

The pre-trained model included with the PyTorch implementation was trained on the UTKFace dataset for 200 epochs and uses a latent vector size of 100. Starting from this pre-trained model, we trained on our augmented VisualBMI dataset for 200 epochs allowing all of the model's parameters to change. Training was done on GPU-enabled AWS instances.

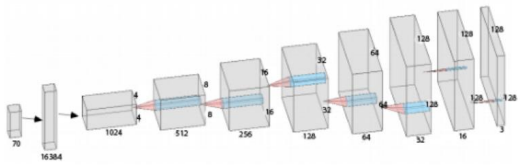
Originally, the conditional autoencoder grouped ages into 10 age buckets of approximately 10 years each. However, when applying transfer learning to the BMI generation, we needed to group our BMI-labeled images into buckets for the autoencoder to train on. We decided to keep the same number of buckets and decided on BMI buckets in a way that would spread our dataset evenly between the buckets. A Python script was used to determine which BMI ranges to use and also to label our dataset based on these new buckets for use with the conditional autoencoder. The buckets we used grouped BMIs into the following ranges: (0-23), (23-25), (25-27), (27-29), (29-31), (31-33), (33-36), (36-39), (39-44), (44+).

4.3 Network Architecture

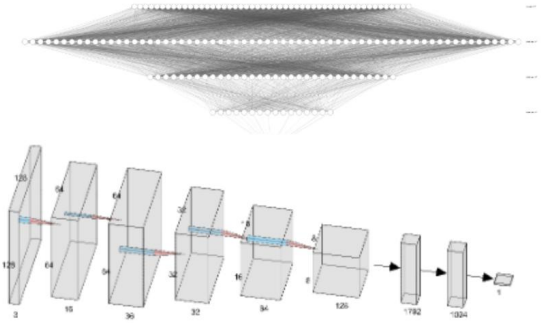
Encoder: 5 convolutional layers and a fully connected layer. Face images of dimension 128x128x3 are converted to latent vectors of configurable size (default size of 50).



Generator: 7 deconvolutional layers and a fully connected layer. Labeled Z vectors in a latent space are transformed into face images of dimensions 128x128x3.



Discriminators: One discriminator on Z with 4 fully-connected layers, and a second on images with 4 convolutional layers and 2 fully connected layers.



5 Experiments / Results / Discussion

5.1 Hyperparameters

Our model was trained for 200 epochs using the following hyperparameters:

1. Batch Size: 64
2. Learning Rate: 2e-4
3. Weight Decay: 1e-5
4. Beta 1: 0.9
5. Beta 2: 0.999
6. Latent (Z) Vector Size: 100

5.2 Loss

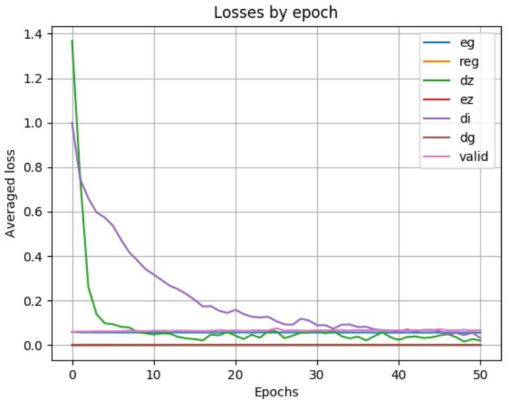
To calculate the loss on the encoder-decoder, our model uses a mean absolute error loss function. Specifically, we are using PyTorch's implementation of L1Loss. According to the PyTorch documentation [14], this loss function can be described as an average over the following quantity:

$$l_n = |x_n - y_n|$$

For loss functions on the discriminators our model uses binary cross entropy (BCE). Specifically, we are using PyTorch's implementation of BCEWithLogitsLoss. According to the PyTorch documentation [14], this loss function can be described as an average over the following quantity:

$$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right]$$

The following is a plot of the different losses used graphed over the last 52 epochs of training.



5.3 Metrics

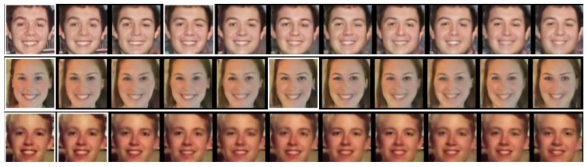
Qualitative and quantitative methods were employed to evaluate the output of our model. From visual observation, we were able to see that the model produced favorable results when input images were clear and aligned with subjects facing the camera. We also observed that the model outputted better results when generating images that increase an individual's BMI rather than decrease it. This may have resulted from the fact that our dataset contained significantly more images of individuals at high BMI's (30+) than lower BMI's (< 20), so our model was able to better learn features of high-BMI individuals.

To quantitatively evaluate our results, we adopted a method similar to that used by G. Antipov, M. Baccouche, and J.L. Dugelay in their face aging study [3]. Using the Face-to-BMI classifier, we computed the mean difference between what BMI an image was classified to have and what BMI range an image belongs to. This mean difference was 6.44 for our generated images, only 2.5 points more than the mean difference for the original input images.

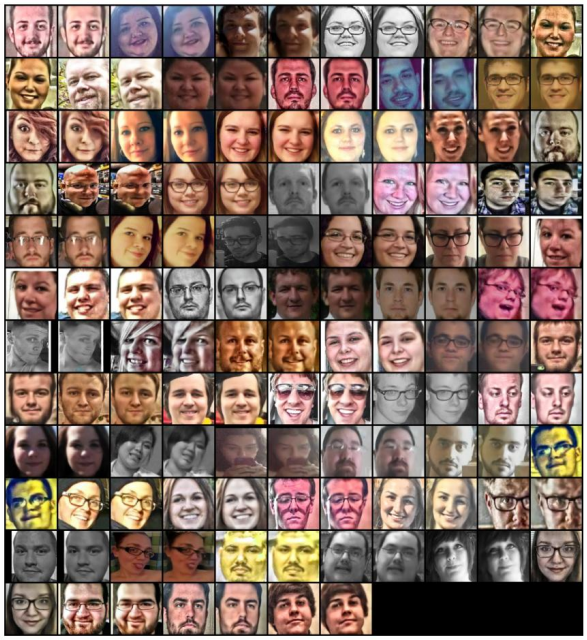
5.4 Sample Results

Below are 3 example BMI output progressions. The input is the leftmost image and all other images are outputted by our model. The single image outlined in white in each example corresponds to the model's

generation of the input face at the individual's actual BMI.



The following is an example of many input faces (left) and the images reconstructed (right) from their latent vector representation.



6 Conclusion / Future Work

Here we show a successful application of transfer learning on an existing conditional adversarial autoencoder (CAAE). We took a CAAE trained for age progression/regression of face images and applied transfer learning to train a model that given an input face image and BMI, would predict how that person would look for different BMIs.

Due to time and resource constraints, we settled on a simple mean difference metric for qualitatively evaluating the output of our CAAE. Given more time we would have liked to explore other metrics used to evaluate autoencoders. Our model uses hyperparameters that are standard for GANs and related architectures; however, in future work it would be worth exploring how tuning these may improve our model. Because we found that the amount of publicly available BMI-labeled face images was limited, it would be interesting to collect labeled data for use in future work.

7 Contributions

John Romano: During our initial training, we encountered high losses and blurry output images that we suspected was due to uncentered face images. Used the autocrop library to crop and center the faces, which fixed the problem. Also made a data augmentation script using the albumentations Python library that sent us from 4200 to 42000 images (imageconvert.py).

Edward Guzman: Requested the VisualBMI dataset from Kocabey, E., et al. Helped set up the AWS instance and ensure that CUDA was being used correctly for faster training. Modified the PyTorch implementation of the autoencoder to support batch image testing for faster testing. Wrote Python scripts used to label our augmented images, group them into evenly distributed buckets, and process the output images of our autoencoder for evaluation.

Alejandro Bravo: Set up AWS instance and monitored training and testing. Wrote scripts to apply the BMI classifier to all input images and all output images and publish the results to a CSV file for ease of analysis on a spreadsheet. Did quantitative analysis with spreadsheet.

7.1 Github Repos

https://github.com/eguzman3/BMI-CAAE
https://github.com/jromano87/face-aging

References

[1] United States of America, Department of Health and Human Services, Centers for Disease Control and Prevention. (n.d.). Body Mass Index: Considerations for Practitioners. https://www.cdc.gov/obesity/downloads/bmiforpactitioners.pdf

[2] Kocabey, E., et al.(2017). Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media. Retrieved from https://arxiv.org/pdf/1703.03156.pdf

[3] Antipov, G., Baccouche, M., & Dugelay, J. (2017). Face Aging With Conditional Generative Adversarial Networks. Retrieved from https://arxiv.org/abs/1702.01983

[4] Wang, Z., Tang, X., Luo, W., & Gao, S. (n.d.). Face Aging with Identity-Preserved Conditional Generative Adversarial Networks. Retrieved from http://openaccess.thecvf.com/content_cvpr_2018/papers/Wang_Face_Aging_With_CVPR_2018_paper.pdf

[5] Zhang, Z., Song, Y., & Qi, H. (n.d.). Age Progression/Regression by Conditional Adversarial Autoencoder. Retrieved from https://arxiv.org/abs/1702.08423

[6] Song, Y., & Zhang, Z. (n.d.). UTKFace. Retrieved from http://aicip.eecs.utk.edu/wiki/UTKFace

[7] Tiddeman, B., Burt, M., & Perrett, D. (2001). Prototyping and transforming facial textures for perception research. Retrieved from https://ieeexplore.ieee.org/document/946630

[8] N.Ramanathan and R.Chellappa. Modeling age progression in young faces. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, IEEE, 2006.

[9] Kemelmacher-Shlizerman, I., Suwajanakorn, S., & Seitz, S. M. (2014). Illumination-Aware Age Progression. 2014 IEEE Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2014.426. Retrieved from https://dl.acm.org/citation.cfm?id=2679839

[10] Erik Linder-Noren, PyTorch-GAN, 2018, GitHub Repository, URL: https://github.com/eriklindernoren/PyTorch-GAN

[11] mattans, PyTorch Implementation of Age Progression/Regression by Conditional Adversarial Autoencoder, 2019, GitHub Repository, URL: https://github.com/mattans/AgeProgression

[12] Francois Leblanc, autocrop, 2019, GitHub Repository, URL: https://github.com/leblancfg/autocrop

[13] Alexander Buslaev, Albumentations, 2018, GitHub Repository, URL: https://github.com/albu/albumentations

[14] https://pytorch.org/docs/stable/nn.html