MIE324 - Introduction to Machine Intelligence

TIMEMACHINET FINAL REPORT

December 2, 2018
Word Count: 1855
(excluding title page, table of contents, and bibliography)

Woo Sik Kim, 1003379952
Anup Deb, 1003260384
University of Toronto

Contents

1 Introduction

1.1 Goal

TimeMachiNet is an application that aims to predict the progression and regression of a person's face as a function of age. Given an image of a person at an arbitrary age, the application will predict how the person looked like in the past and how the person will look like in the future. Specifically, the output will consist of 10 predictions (as images) corresponding to 10 different age ranges (31-40 years, 41-50 years, etc.) without compromising the particular features of the person's face.

1.2 Motivation

This technology can be used in multiple applications where an age transformation is needed, such as in facial prediction of wanted people, age-invariant verification, recreational use (personal curiosity), or even assisting in the search for missing people.

2 Sources of Data

The dataset for the neural network in this project is the UTKFace dataset [3] which contains 20,000 (200 x 200 px) aligned and cropped facial images of people of various race, gender and age. A few sample images are shown in Figure 1.

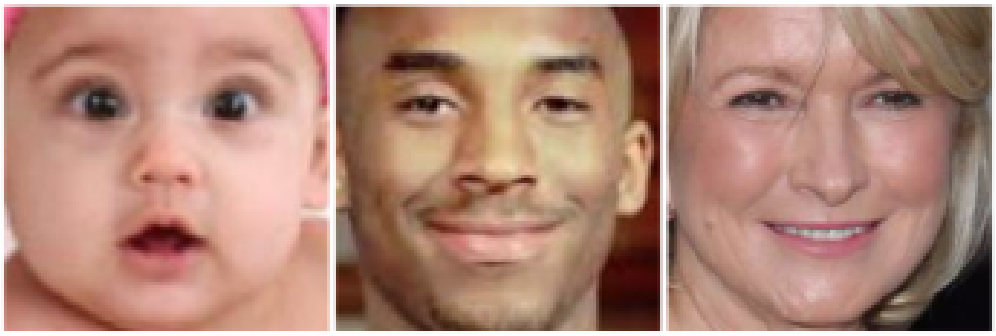

Figure 1: Sample images from UTKFace dataset.

Some statistics about the dataset were determined and we found that it contains an almost equal number of males and females. On the other hand, there are more images of people below the age of 30 than people above the age of 70, as shown in Figure 2. The race distribution of the dataset is skewed towards the white race. Furthermore, some of the age labels were blatantly incorrect in which case we either removed or relabeled them.

To reduce computational cost, the images were normalized and down-sampled to 128 x 128 px (resized and centre-cropped). Then the images were labeled with the appropriate age ranges (one of 0-5, 6-10, 11-15, 16-20, 21-30, 31-40, 41-50, 51-60, 61-70, 70+) that were chosen to mitigate the in-balance of age in the dataset.

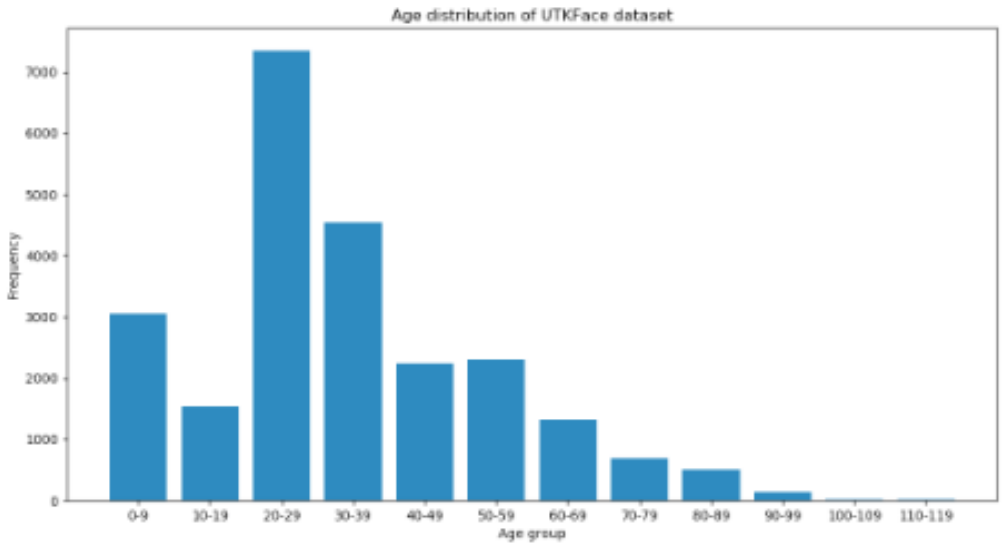

Figure 2: Age distribution of UTKFace dataset.

3 Description of Overall Software Structure

3.1 Software Structure

TimeMachiNet consists of a client and server side. On the client side, users will have the option to select an input image by browsing their computer. Once an image is submitted, a `GET` request is sent to the API back-end which calls `use_model.py` to pre-process the input image. The new image is then passed to the pre-trained models to generate the 10 images that are displayed on the client side for users to observe. See Figure 3.

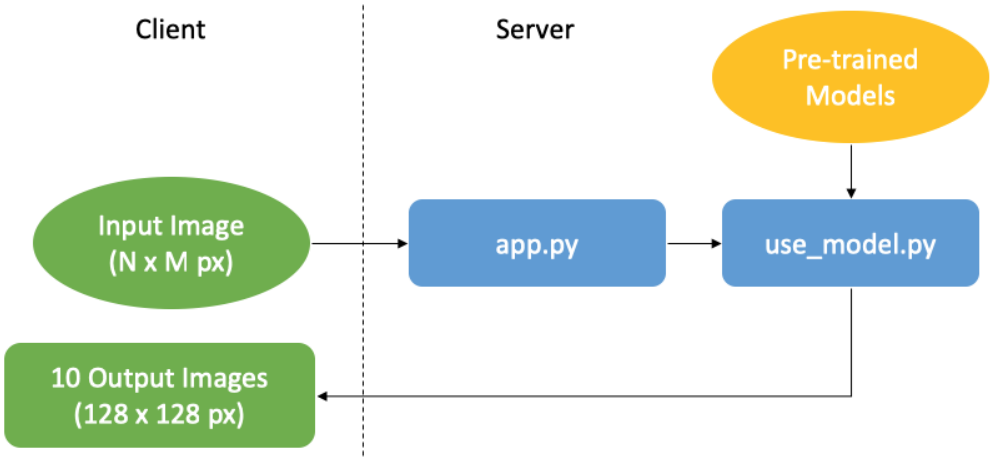

Figure 3: The structure of TimeMachiNet's software.

3.2 The Model: Conditional Adversarial Autoencoder

We use an architecture known as the Conditional Adversarial Autoencoder, inspired from 'Age Progression/Regression by Conditional Adversarial Autoencoder' by Z. Zhang et. al [1]. It consists of an encoder, two discriminators, and a generator, as shown in Figure 4.

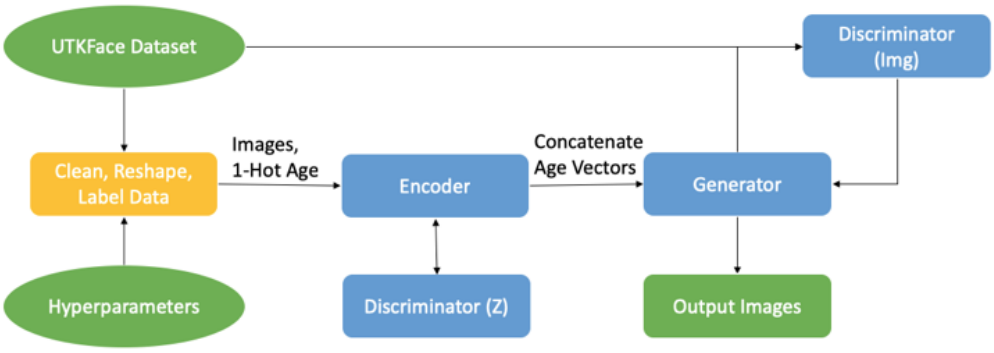

Figure 4: Basic framework of the Conditional Adversarial Autoencoder.

In training mode, we first load the path to our dataset and hyperparameters for our model (learning rate, betas, batch size, and number of epochs). We then pre-process the image data by extracting age, resizing and normalizing the images. The processed images are first passed to an encoder, E, which produces a feature (latent) vector corresponding to the image. The latent vectors play the role of encapsulating the facial features (personality) of the input face, regardless of age. The first discriminator, D_z, is coupled to the encoder and its purpose is to regularize the latent vectors to be uniform distributed, smoothing the age transformation for faces not used to train [1]. The latent vectors are concatenated with the one-hot encoded age labels and passed into the generator, G, which generates versions of the input image from the latent vectors, conditioned on age. The generator is coupled to a discriminator, D_{img} that tries to discriminate between real images and the generated reconstructions at the same age, thus training the generator to create realistic images [4]. In inference mode, we encode the input image and run the generator on all possible age options to generate the predicted face for the person's lifetime.

3.2.1 Encoder

The encoder is a convolutional neural network that receives a 128 x 128 px image as input and outputs a 50-element vector. It consists of 4 convolutional layers, 1 linear layer, and ReLU activations. A convolutional neural network is appropriate here because it is the architecture of choice for identifying features in images. Each convolutional layer has a kernel size of 4 x 4, stride of 2 (to reduce computational cost and allow the network to learn the appropriate downsampling of the image instead of manually pooling) and padding of 1.

A key insight is that the neural network is 'learning' a high dimensional manifold of faces (each dimension represents a feature of the image) and the encoder is learning to represent this manifold with a space of simple 1-dimensional feature (latent) vectors. By keeping age independent of the other facial features, we can 'move' through the latent space in the direction of increasing or decreasing age and project the vector back on the manifold

to recover the desired facial image. Mathematically, if we denote the manifold, M and an image x where $x \in M$, then $z = E(x)$ represents the latent vector, and we can concatenate a one-hot age label, l, to get $[z, l]$, a point in latent space. We can 'move' in the latent space by changing l and generate a corresponding image $y = G(z, l)$ conditioned on l. This is shown in Figure 5. We would like to impose a uniform distribution on the encoder's output so that space of feature vectors (representing personality) becomes populated evenly, therefore smoothing the aging process for people that the model has not trained on. We use the encoder as the input to our generator rather than random noise because we want the output to retain the identity of the person. We train E by maximizing the loss of D_z, given by $log(D_z(E(x)))$.

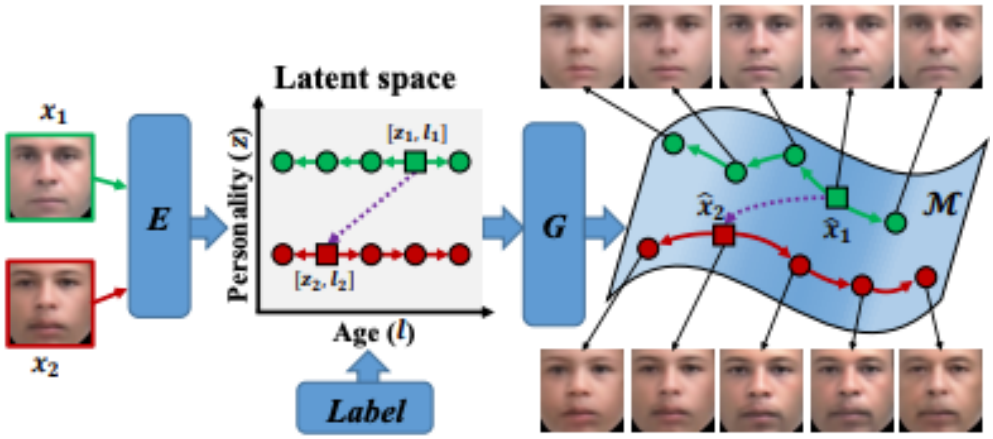

Figure 5: Outline of transformations performed by encoder and generator [1].

3.2.2 Generator

The generator is a de-convolutional neural network that performs the operation of transforming a latent vector back into an image. The generator is learning a mapping between vectors in the latent space back to the manifold of faces. The architecture of the generator consists of 1 fully connected linear layer and 5 de-convolutional layers, with ReLU and Tanh (at the end) activations. Note that the pixels in the images and the latent vectors are normalized to [-1, 1] to improve training speed. The kernel size, number of input and output channels of the de-convolutional layers mirror the opposite process of the encoder. Ultimately, we want to be able to generate a person's face conditioned on one of the 10 possible age ranges. The loss of the generator, G, can be expressed mathematically as a weighted average of $\|x - G(E(x), l)\|$ and $log(D_z(G(z)))$. The first term is the reconstruction loss between the input image and the generated image at the original age group and the second term represents the fact that we want to fool the discriminator into thinking the generated images are real.

3.2.3 Discriminator on Encoder

The discriminator on the encoder, D_z, consists of 4 linear layers (with ReLU activations) and a Sigmoid activation on the output layer to produce a scalar value. D_z takes in the latent vector z produced by the encoder and trains the encoder so that it produces latent vectors that form a uniform distribution. This procedure ensures that they evenly span the latent space. Without this condition, the model will not be able to generalize well to new inputs as the model will not have the experience from previous similar images to define an appropriate mapping between the manifold and latent space for the new input, which will result in unrealistic outputs. The loss function of D_z is $BCE(D_z(z^\star), 1) + BCE(D_z(E(x), 0)$ where $z^\star$ is a random uniformly distributed vector and x is the input image [1]. This loss function forces E to produce uniformly distributed vectors as its outputs.

3.2.4 Discriminator on Generator

The discriminator on the generator, D_{img}, consists of 2 convolutional layers followed by 2 linear layers (with ReLU activations) and a Sigmoid activation on the output layer to produce a scalar. D_{img} predicts a number close to 1 if it predicts that the input image is real (from dataset) and 0 otherwise so its loss function is $BCE(D_{img}(x), 1) + BCE(D_{img}(G(E(x))), 0)$. This makes sense because the first term corresponds to the real image and should be close to 1 (to minimize loss) and the second term corresponding to the generated image should be close to 0. The purpose of the discriminator is to force the generator to generate more realistic images.

Figure 6: Training losses through 50 epochs.

4 Training, Validation and Test

We found that Adam quickly converged to reasonable qualitative results so it was used instead of SGD. The losses for the various components of the neural network are shown in Figure 6. Most of the losses (E, G, D_{img}, D_z) are small in magnitude and stay relatively constant. Notice that G_{img} loss increases as the number of epochs increases. This indicates that D_{img} is finding it difficult to guess real versus reconstructed images.

The qualitative results of the training process (using a validation image) is shown in Figure 7, where age is increasing from left to right. It is clear that the aging process is slightly evident even after 10 epochs, although the generated images are slightly blurry. On the other hand, the aging process after 50 epochs is much more evident and realistic.

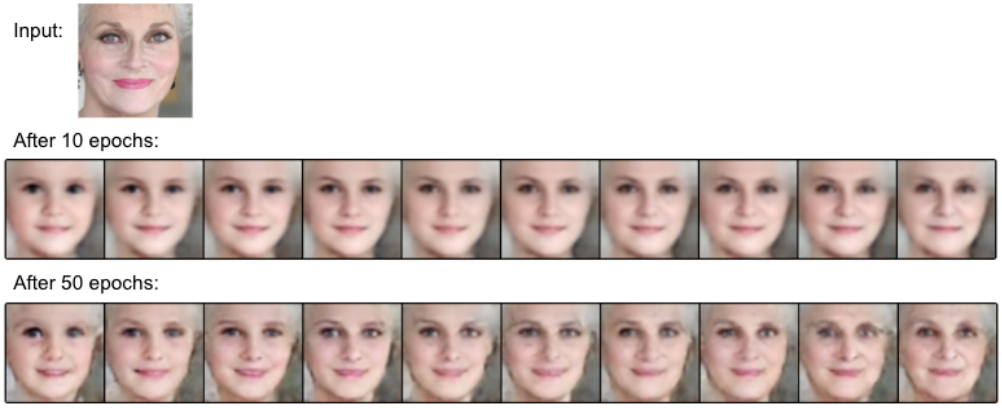

Figure 7: Sample output for image using model trained for 10 and 50 epochs.

A successful test output of the application for a middle aged man is shown in Figure 8. We can see the effects of both the progression and regression of age.

Figure 8: Sample output using final trained model.

Hence, in reasonable conditions the model performs the bidirectional age progression/regression well. However, a disadvantage is that the major differences between output images are facial shape and skin texture, but features like hair color are untouched. Due to the nature of GANs we also fail to recreate some details in the input image and produce slightly lower-quality images. The model performs unexpectedly well with non-aligned images as well so it is able to generalize as shown in Figure 9 (likely due to variation in dataset).

This example shows that there is still some distortion in the facial features for some of the predictions.

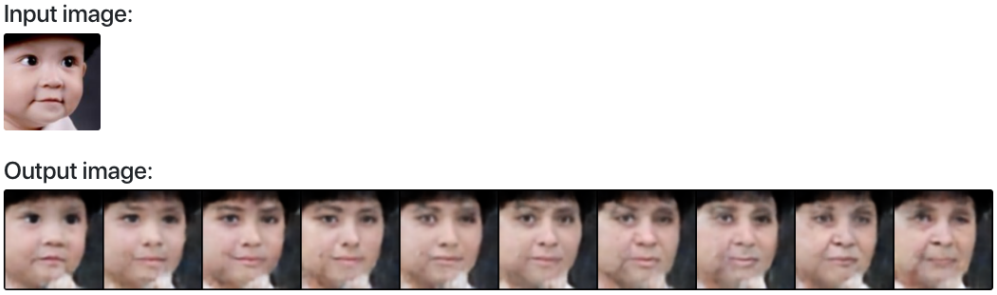

Figure 9: Sample output with non-ideal conditions.

5 Ethical Issues

There are a few ethical issues that may arise from TimeMachiNet. The application may perpetuate biases or stereotypes about the aging process of people with differing genders and ethnic groups or misrepresent reality. Additionally, aging could be a sensitive subject matter for people using the application if used recreationally for themselves or others. Lastly, if the application is used for verification or identification of any kind, inaccurate results may lead to injustices or erroneous identification.

6 Key Learnings

Some of the key learnings we had during this project are:

1. Begin training early and often, since training takes a long time.
2. GANs are tricky to train so research tips for helping to train them (e.g. strided convolution).
3. Save pre-trained models incrementally during training because the best model may not be the final one and training might time out unexpectedly.
4. Have some way to qualitatively check on the training process to gauge effectiveness quickly and know when to start over.

If we could start again we would make sure to:

1. Check thoroughly for bugs in code and pre-processing before training to save time.
2. Carefully optimize memory usage to make it possible to quickly train the model without a ridiculous amount of RAM.

References

[1] Z. Zhang, Y. Song and H. Qi, *Age Progression/Regression by Conditional Adversarial Autoencoder*, 2018. [Online]. Available: https://arxiv.org/pdf/1702.08423.pdf.

[2] *Neural Network Learns to Synthetically Age Faces and Make them Look Younger*, 2018. [Online]. Available: https://www.technologyreview.com/s/603684/neural-network-learns-to-synthetically-age-faces-and-make-them-look-younger-too/

[3] *UTKFace*, 2018. [Online]. Available: https://www.kaggle.com/jangedoo/utkface-new.

[4] G. Antipov, M. Baccouche and J. Dugelay, *FACE AGING WITH CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS*, 2018. [Online]. Available: https://arxiv.org/pdf/1702.01983.pdf.