

# Low Complexity Data-Efficient Generation in Disentangled Setting

Jacob Sela<sup>co</sup>

[jacob.sela@campus.technion.ac.il](mailto:jacob.sela@campus.technion.ac.il)

Shavit Borisov<sup>co</sup>

[shavit@campus.technion.ac.il](mailto:shavit@campus.technion.ac.il)

Elad Richardson

[elad.richardson@gmail.com](mailto:elad.richardson@gmail.com)

## Abstract

*Recent advancements in the fields of generation are promising better results, with more control over generation. Unfortunately, these results are achieved with massive data sets used to train highly complex models by the leading experts of machine learning, making them inaccessible to “the average Joe”. In this paper, we propose that the disentangled latent spaces created as a by-product of these tools can be repurposed for generation with a specific factor of variation in mind, using simple tools and little data. We demonstrate these claims by generating aging videos using NVidia StyleGAN’s latent space from a single source image.*



Code for this project can be found on [GitHub<sup>\[1\]</sup>](#). To view GIFs, please go to the official [Google Doc<sup>\[1\]</sup>](#).

**Table of Contents**

<b><u>1 Introduction</u></b>	<b><u>2</u></b>
<b><u>2 Methods</u></b>	<b><u>3</u></b>
<u>2.1 StyleGAN</u>	<u>3</u>
<u>2.2 Puzer's Encoder</u>	<u>3</u>
<b><u>3 Models</u></b>	<b><u>3</u></b>
<u>3.1 Simple Linear Interpolation</u>	<u>3</u>
<u>3.2 Linear Regression, Direct</u>	<u>4</u>
<u>3.3 Linear Regression, Offset</u>	<u>4</u>
<u>3.4 MLP</u>	<u>4</u>
<b><u>4 Tools</u></b>	<b><u>6</u></b>
<u>4.1 Early Tools</u>	<u>6</u>
<u>4.2 Data Collection</u>	<u>6</u>
<u>4.3 Automation and GIF Generation</u>	<u>6</u>
<b><u>5 Results</u></b>	<b><u>7</u></b>
<u>5.1 Motivation</u>	<u>7</u>
<u>5.1 Simple Linear Interpolation - Results</u>	<u>8</u>
<u>5.2 Linear Regression Direct - Results</u>	<u>8</u>
<u>5.3 Linear Regression, Offset - Results</u>	<u>9</u>
<u>5.4 MLP - Results</u>	<u>10</u>
<u>5.5 Puzer's Vector - Results</u>	<u>11</u>
<b><u>6 Results Analysis</u></b>	<b><u>12</u></b>
<b><u>7 Conclusions</u></b>	<b><u>13</u></b>
<b><u>8 Appendix - Project Diary</u></b>	<b><u>14</u></b>
<u>8.1 Beginning</u>	<u>14</u>
<u>8.2 Finding Our Feet</u>	<u>14</u>
<u>8.3 Linear Interpolation</u>	<u>16</u>
<u>8.4 Linear Regression, Direct</u>	<u>19</u>
<u>8.5 Linear Regression, Offset</u>	<u>21</u>
<u>8.6 MLP</u>	<u>29</u>
<u>8.7 Puzer's Original Vector</u>	<u>31</u>

## **References** **32**

### **1 Introduction**

Recently, much work has been done on the topic of generation from a disentangled space<sup>[2][12][13][14]</sup>, usually by first using some form of encoding to disentangle the space. This has made generation problems simpler to tackle by allowing to directly and precisely change a single factor of variation. However, training models to disentangle high dimensional spaces is expensive in time (both computationally and in human work) and resources (a highly varied

and expansive data set is required)<sup>[1][2][3]</sup>. This sets a barrier, allowing only those highly trained in such systems of deep learning and equipped with a lot of data to effectively evoke these models for generation.

One would expect, however, that a sufficiently disentangled space would isolate many features to a reasonably linear degree, allowing for a simpler model to be applied onto the disentangled latent space in order to isolate the needed features. We suggest that this model of work can be automated and repurposed for many features and could thus offer a framework by which much more generation work can be done. We demonstrate this using NVidia StyleGAN’s latent space and the age feature, showing that we can age an individual using only a linear regression model with comparatively very little data<sup>[9][14][15][16]</sup>. We benchmark our results against a deep learned direction for aging created by Puzer as well as our own shallow (2-hidden layer) neural network.

Aging was picked for two reasons. First, aging data is readily and freely available online by looking up “A Photo a Day” on YouTube (this is also the namesake of the project). Though these videos are certainly biased (the vast majority being of white males), and we discuss this later in the paper, their granularity allows for a lot of flexibility in the dataset that is used to train the system. Aging is also useful to demonstrate the limited data needed to learn the distribution. Many previous works on aging relied on a much more expansive data set<sup>[17][18]</sup>, while we mainly use three videos (610 photos in total, after cutting down to a photo every 10 days) to achieve our results.

We demonstrate these claims throughout the paper, step-by-step as they were implemented.

## 2 Methods

### 2.1 StyleGAN

StyleGAN is a “style based” generative adversarial network. We use it to generate our images, as well as its latent space for extrapolation. It was picked for its (advertised) excellent interpolatability in the latent space, as well its excellent generated photo quality<sup>[4]</sup>. NVidia released the decoder for StyleGAN and we use this to decode our latent representations into actual images. However, NVidia had not released the encoder for StyleGAN, which is why we use Puzer’s encoder ([section 2.2](#)) to encode our images and receive our latent representations.

### 2.2 Puzer’s Encoder

Puzer’s encoder<sup>[4]</sup> is a 3<sup>rd</sup> party deep-learned tool to encode photos into StyleGAN’s latent space. We use this tool to encode our images. Puzer additionally learned a direction for aging in StyleGAN’s latent space using a 2-layer hidden network; this was the inspiration for the project.

## 3 Models

### 3.1 Simple Linear Interpolation

In order to first check whether StyleGAN’s latent space was indeed linear and continuous to a degree sufficient for more complex models, we attempted simple linear

interpolation on it. Given two points  $p_1, p_2$  in the latent space, each corresponding to a photo of the same individual at ages  $t_1, t_2$ , we define the following interpolator:

$$\text{interpolator}(t) = p_1 + (t - t_1)(p_2 - p_1) \frac{t_2 - t_1}{t_2 - t_1}$$

We tested this model on start and end points at different distances apart to experiment with different parts of the latent space. We expected that as we increase the gap in age between the two points the interpolation would become weaker and that there would be a correlation between the distance from  $t$  and  $(t_1, t_2)$  and the error of the interpolation (based on the implicit bias that the unfolded manifold would be relatively inflection free). In addition, we tested the interpolation on a subset of the latent space components, as well averaging of multiple photos to find the initial  $p_1$  and  $p_2$ . The error was measured in MSE between the interpolated photo and the original photo at that age in the latent space (unless otherwise stated this is the case for all models).

### 3.2 Linear Regression, Direct

Once continuity and linearity were established (to a degree), we attempted to fit a more complex model to the problem. We tested a linear regression model fitted on linear least squares. That is, given a set of  $n$  data points  $\{(startPhoto_i, startAge_i, targetAge_i, targetPhoto_i)\}_{i=1}^n$  we found the function  $f(startPhoto, startAge, targetAge) = targetPhoto$  such that we minimize:

$$\text{Error}(f) = \sum_{i=1}^n (f(startPhoto_i, startAge_i, targetAge_i) - targetPhoto_i)^2$$

### 3.3 Linear Regression, Offset

In order to find a better fit, we trained for offset instead of direct interpolation. Additionally, we added a few more weighting options for the different components based on a linearity test we devised. Given a video of individual aging  $\{p_i\}_{i=1}^n$ , we define  $\{d_i\}_{i=1}^{n-1}$  where

$$d_i = p_{i+1} - p_i$$

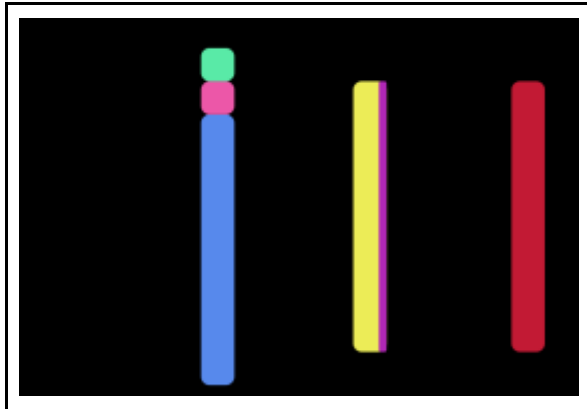
We find the variance of each component of  $d_i$  and sort the indices by increasing variance. Alternatively, we first normalize  $\{d_i\}_{i=1}^{n-1}$  by dividing each component by the greatest absolute value of any  $d_i$  in that component. We call the sorted indices list  $\{mostLinear\}_{i=1}^{151218}$  and the normalized sorted indices list  $\{normalizedMostLinear\}_{i=1}^{151218}$ .

This, along with a limitation on the number of dimensions taken for the regression, gives us three new models:

1. Offset Linear Regression on first  $n$  components (referred to from now on as the *first* method).
2. Offset Linear Regression on most linear  $n$  components (referred to from now on as the *linear* method).
3. Offset Linear Regression on normalized most linear  $n$  components (referred to from now on as the *norm* method).

### 3.4 MLP

To benchmark the quality of our results, we compare them to a shallow MLP. Specifically, we tested the following 2-hidden-layer network:



With our hyperparameters being:

1. The data couplings (limitations on the distance between start and target, quantity).
2. Size of hidden layer (10, 100, 1000, 51218).
3. Input features (All 51218 components, 180 most linear components, 180 normalized most linear components).
4. Activation function on hidden-layer (sigmoid, relu, tanh).
5. Learning rate (0.1, 0.01, 0.001).
6. Regularization (none, 11 - 0.001, 11 - 0.0001, 12 - 0.001, 12 - 0.0001).

In order to tune our hyperparameters, we performed a grid test for the size, activation, learning rate, regularization, and input features.

We found the optimal data set through random search. We defined our space of possible data points as  $i=13(P_i[age][age])P_i$  where  $P_i$  is the set of photos we had for person  $i$ . In each test, we randomly sampled a subset of the data and used it for training.

The model we have found to work best (and subsequently chose as our benchmark) is the following:

1. Data - 6100 samples picked randomly from the above set.
2. Size of hidden layer - 1000.
3. Input features - All 51218.
4. Activation function - *tanh*.
5. Learning rate - 0.01.
6. Regularization - none.

## 4 Tools

### 4.1 Early Tools

To find our feet, we developed a method of downloading and cutting videos; Downloading was based on the YouTube-DL [\[4\]](#) tool, and cutting the video into frames was done using OpenCV [\[5\]](#). We have also built a tool for image comparison, based on the L0 norm, that was later used in the data collection phase ([section 4.2](#)) to remove duplicate frames.

### 4.2 Data Collection

For our datasets, we used “A Photo a Day” videos from YouTube, namely three different videos<sup>[1][2][3]</sup>. We have created a pipeline with the intention to make the process of data gathering quicker. This tool has made it significantly easier to try new experiments, as most of the manual work that was done beforehand to get usable data has been automated away. The pipeline works in the following way:

1. Download the relevant videos, using a premade data structure and the tools discussed previously ([section 4.1](#)).
2. For each video, insert starting age and do the following:
  - a. Cut the video into frames.
  - b. Remove duplicate frames using the image comparison tool we have previously built ([section 4.1](#)).
  - c. Catalog each frame according to age.
  - d. Decimate a certain amount of frames (due to computational limits).
  - e. Align each frame using Puzer’s tool.
  - f. Manually clean faulty frames (usually, no more than 2-3 frames).
  - g. Re-catalog frames.
  - h. For each frame:
    - i. Use Puzer’s encoder<sup>[4]</sup> to receive corresponding latent representation in the form of a `.npy` file.

## 4.3 Automation and GIF Generation

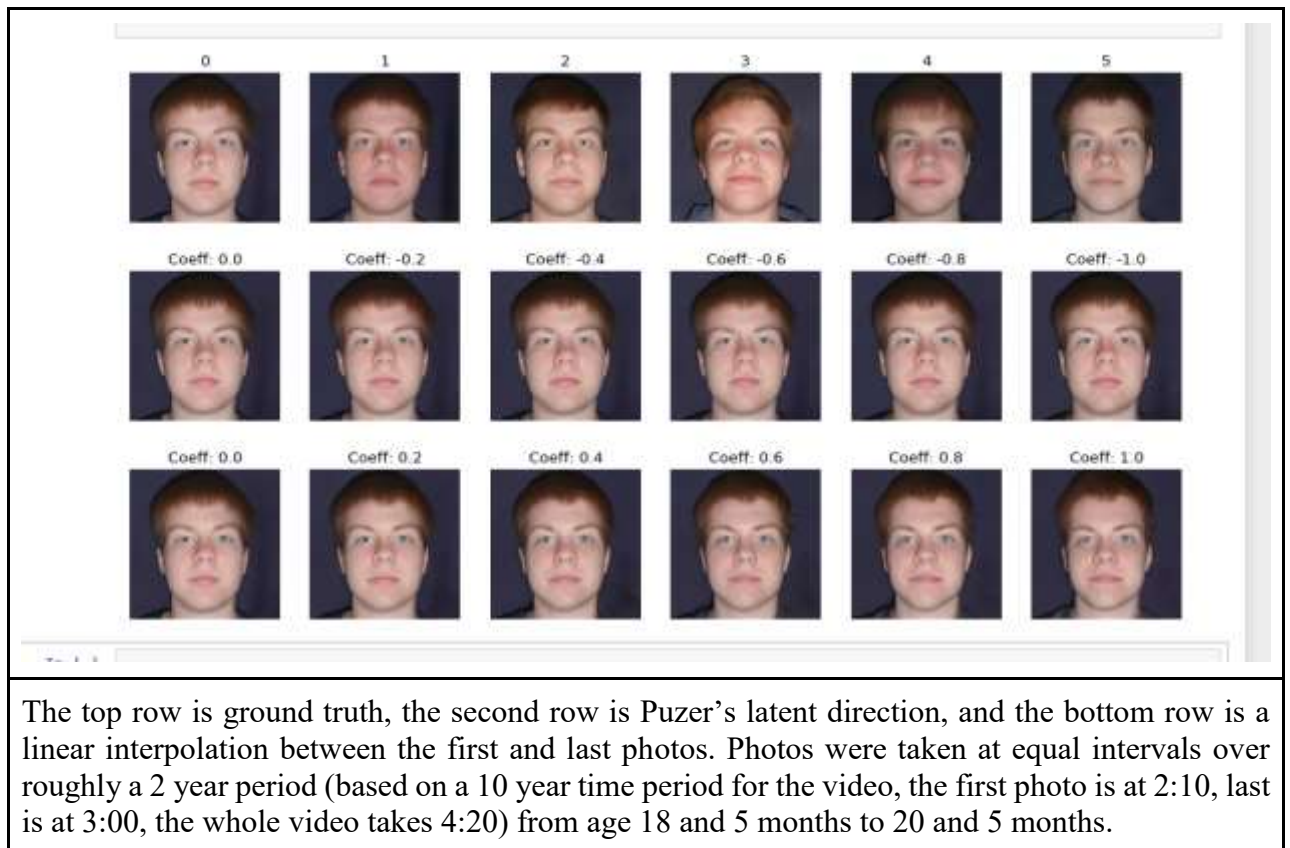
Lastly, we have built a tool to automate tests. This combined all of our previous tools and allowed us to perform multiple tests fairly easily, with the only limit being computational. The pipeline looks as such:

1. Receive input for the test. This consists of all relevant parameters and data.
2. Train the model according to the relevant parameters.
3. Test the model on both internal and external data and create an error vs. age chart for the given model and parameters.
4. Predict latent representations of aging video frames for a given test subject.
5. Convert latent information to actual images, using StyleGAN’s decoder.
6. Create an aging GIF of real and fake data, side by side, using FFmpeg<sup>[5]</sup> tool.

## 5 Results



### 5.1 Motivation

To gain intuition, we first conducted a simple test on video 1<sup>[1]</sup> using Puzer’s latent direction<sup>[4]</sup>. Here is the result:



## 5.1 Simple Linear Interpolation - Results

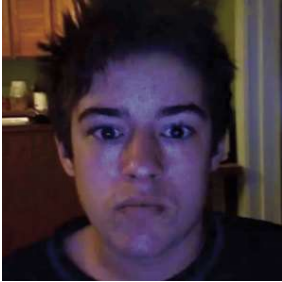



Below is a demonstration of the simple linear interpolation model on internal data:

Original Video	Result
	

## 5.2 Linear Regression Direct - Results




Below are the results for our first linear regression model (using the “direct” method):

	Original	Result

<b>Internal Data</b>		
<b>External Data</b>		

### 5.3 Linear Regression, Offset - Results

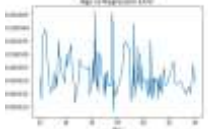
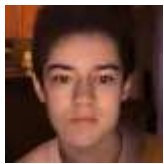



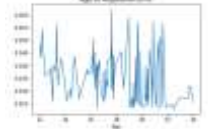




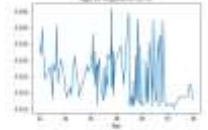
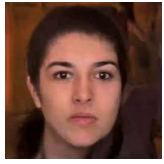

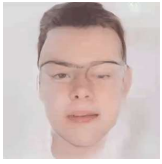

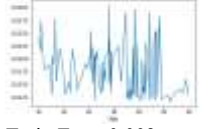
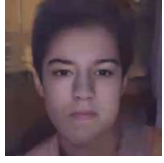

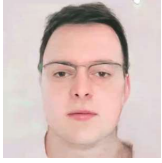

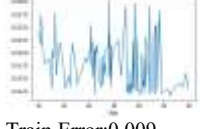
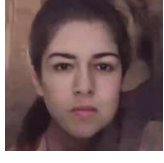
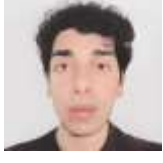
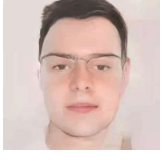
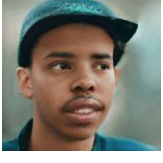
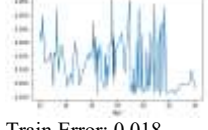
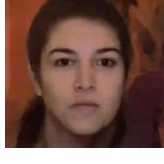
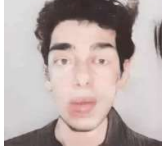

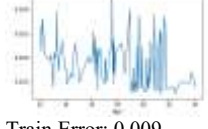
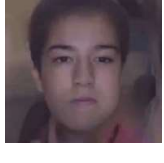
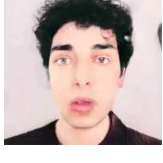


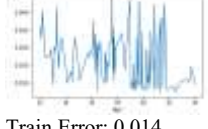
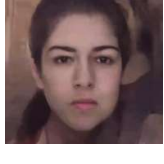
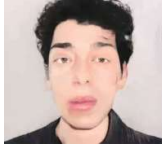


Below are the results for our linear-regression tests, using the offset method. Original photos:

		
Jacob	Shavit	Earl

Results:



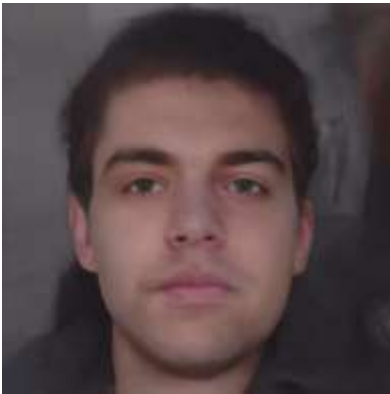
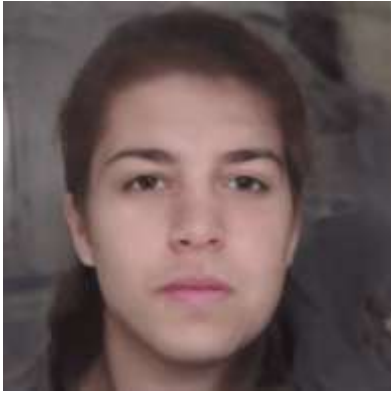

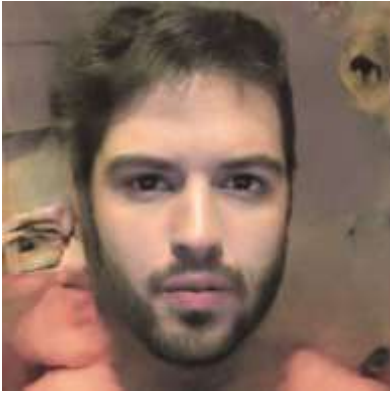
Latent Dims	Mode 1	Test Error	Internal GIF	External GIF - Jacob	External GIF - Shavit	External GIF - Earl
-------------	--------	------------	--------------	----------------------	-----------------------	---------------------



A few (10)	First	 Train Error: $4.35710 \cdot 10^{-6}$ Test Error: $2.20110 \cdot 10^{-5}$				
All (9216)	N/A	 Train Error: 0.019 Test Error: 0.036				
Half (4608)	First	 Train Error: 0.014 Test Error: 0.025				
	Linear	 Train Error: 0.003 Test Error: 0.008				
	Norm	 Train Error: 0.009 Test Error: 0.018				
Three quarters (6912)	First	 Train Error: 0.018 Test Error: 0.033				-
	Linear	 Train Error: 0.009 Test Error: 0.019				
	Norm	 Train Error: 0.014 Test Error: 0.027				

## 5.4 MLP - Results

Below are the results for different MLP methods we tested. External data used here is Jacob's original photo (same one presented in [section 5.3](#)).

		
10 components, <i>sigmoid</i> , external data.	100 components, <i>relu</i> , external data.	1000 components, <i>tanh</i> , external data.
		
100 forced distance components, external data.	100 "most linear" forced distance components, external data.	100 overfitted components, internal data.

## 5.5 Puzer's Vector - Results

Using Puzer's aging vector on Jacob's original photo (same one presented in [section 5.3](#)) led to the following result:



## 6 Results Analysis

As can be seen in the [Results](#) section above, we have achieved our goal of generating convincing aging videos. Certain features of aging such as beard growth, wrinkle formation, face restructuring, and receding hair are present in one model or another. This was done using a small data set<sup>[4][5]</sup> of just 610 original photos, a few orders of magnitude smaller than needed to train state of the art generation models which rely heavily on unsupervised learning<sup>[3][12][13]</sup>. Additionally, we achieved this goal with arguably simpler methods than those of other latent interpolation techniques<sup>[4][5]</sup>, relying only on a linear function to interpolate and using the comfortably familiar L2 as our metric, despite previous works suggesting that it is not optimal. We also demonstrate that it is not trivially simple to replace our models with a superior MLP, as it is difficult to tune an MLP to such a high dimensional space with such a small amount of data.

The most realistic looking result is from the regression on normalized most linear components model with half the dimensions taken as input. It kept a relatively low error rate on the train data that translated well into the test data, and is the most generalizable (in terms of its performance on external individuals). It also avoids many of the deformations and artifacting that results from the other models, namely the facial deformities common in the regular regression, and the lighting issues in the regression on the most linear components without normalization. Indeed, when we expand the number of components taken into account to three quarters, we again see some of this artifacting and deformities, meaning that they were likely features encoded in the components dropped when only half the components are taken. We can also see that this model performs better than the naive MLP, indicating that we succeeded in our initial goal. Though we do not match Puzer’s learned direction, we did not expect to, there is still no doubt that a well trained network will outperform any simple model, given it is trained correctly. However, Puzer used FFHQ as data and a more complex model<sup>[6]</sup>.

We take from this that a simple feature extraction method applied before we use a learning model, more akin to classical computer vision techniques, is still easier and simpler to implement than many of the modern models (which would learn such a feature “on their own”). Specifically, features based on time or other metric that gives order or sequence are notoriously hard to learn<sup>[14]</sup>.

That said, we were not able to generate the convincing, smooth, realistic transformations advertised in the StyleGAN informational video<sup>[20]</sup>. We believe that this can be explained by the overfitting taking place in our models (every regression model had a lower train error than test error). This happens due to the small variance in the data set, with only three individuals, of similar ethnic background (and thus appearance and aging features), aging around the same timeframe. This leads to weak generalizability, as can be seen in the external test. We also inherit other biases present in the data, such as the lighting problems (in one of the videos<sup>[4]</sup> the photos were on average brighter towards the later end), an emphasis on beard growth (all three videos were of men and two of them grew out beards once they were able to past adolescence).

## 7 Conclusions

We trained a simple model with a small data set to generate aging videos by utilizing StyleGAN’s latent space. These methods can be used on other latent spaces for generation with a single (or a few) factors of variation in mind. This leads us to believe that a promising avenue of generation would be simple, portable models for latent extrapolation and the creation of

general latent spaces as that created by NVidia, rather than the end-to-end, complex models that are more common today<sup>[1][2]</sup>.

We can expand on the work done here by comparing more classical learning techniques on this latent space, finding more easily extractable and relevant features in a latent space, expanding tests to other latent spaces, as well as testing applicability for more features and factors of variation. Another interesting direction for future work can be on the effectiveness of a data set in capturing the signal for a certain factor of variation based on its distribution (i.e. how many different individuals should we have trained on here, at what ages, of what backgrounds, etc.).

Thanks to Elad Richardson for supervising this project and to you, dear reader, for making it here.

## 8 Appendix - Project Diary

### 8.1 Beginning

As the new Spring semester started, we have spent most of the first 2 months on studying the material we would later use in this project. Specifically the Stanford convolutional neural network course<sup>[3]</sup> and the original paper<sup>[4]</sup>.

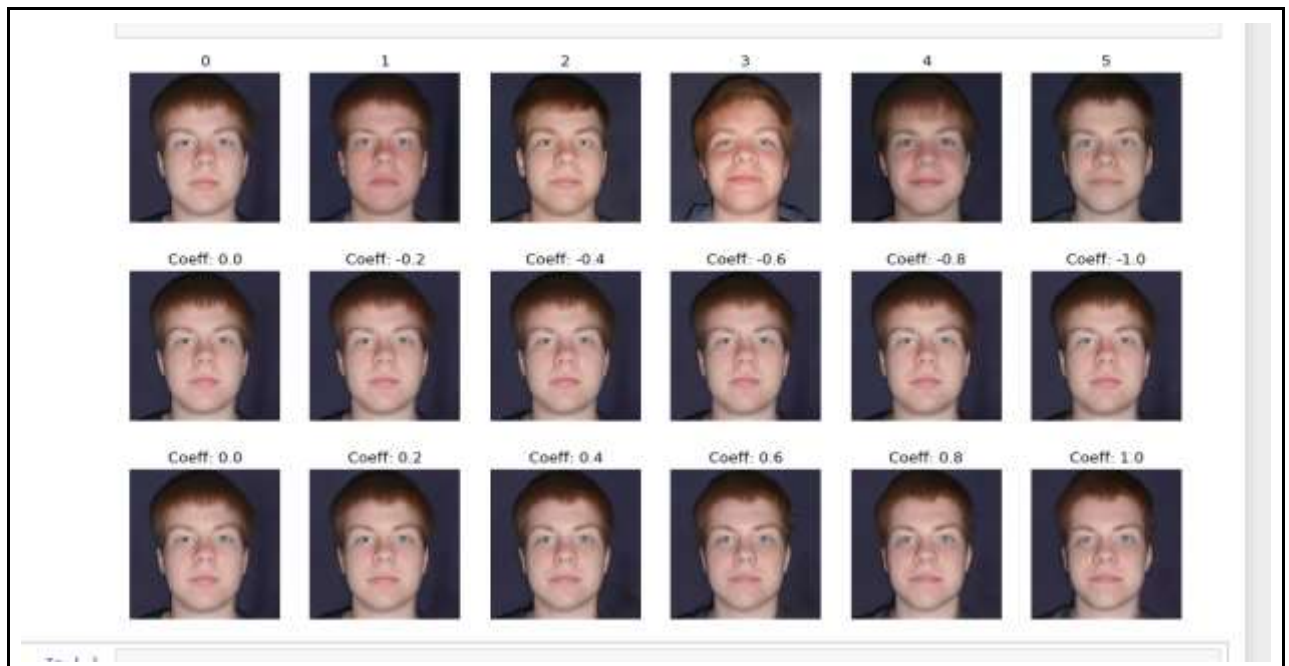
Next, we started tool-building and data collecting. We developed a method of downloading and cutting videos; Downloading was based on the YouTube-DL<sup>[5]</sup> tool, and cutting the video into frames was done using OpenCV<sup>[6]</sup>.

We have built a tool for image comparison using the L0 norm that will later be used to remove duplicates.

Troubleshooting has been a major issue during this time since we have run into technical difficulties quite a few times. These problems mostly had to do with setting up the environment in the lab for the tools we were building.

### 8.2 Finding Our Feet

The next objective is to find a more robust and accurate path for aging in StyleGAN's latent space. In order to gain some basic intuition, we start by comparing ground truth to an already learned latent direction by Puzer<sup>[7]</sup> and a simple linear interpolation.



**Figure 1** [\[a\]](#)

Top row is ground truth, the second row is Puzer's latent direction, and the bottom row is linear interpolation between the first and last photo. Photos were taken at equal intervals over roughly a 2 year period (based on a 10 year time period for the video, first photo is at 2:10, last is at 3:00, the whole video takes 4:20) from age 18 and 5 months to 20 and 5 months.

This shows that on some scale, aging can be approximated linearly, though this proof is a bit contrived, more results are needed, specifically over a larger timescale.

In order to continue experimenting we need more data in the latent space. We downloaded the full video that was used yesterday and cut it into frames. Due to time and computational power limitations, we are only encoding every tenth frame (about every 5 days); even under this limitation the runtime for encoding is currently projected to be 31 hours (after 3 hours of runtime), as of 4:25 pm. The final result will be 780 evenly spaced photos over a 10 year period.

As the photos start trickling in an issue is arising. It seems the encoder has a difficult time handling child and adolescent photos. A few of the results were a bit unsettling. This will narrow down the usable data to that of adults.



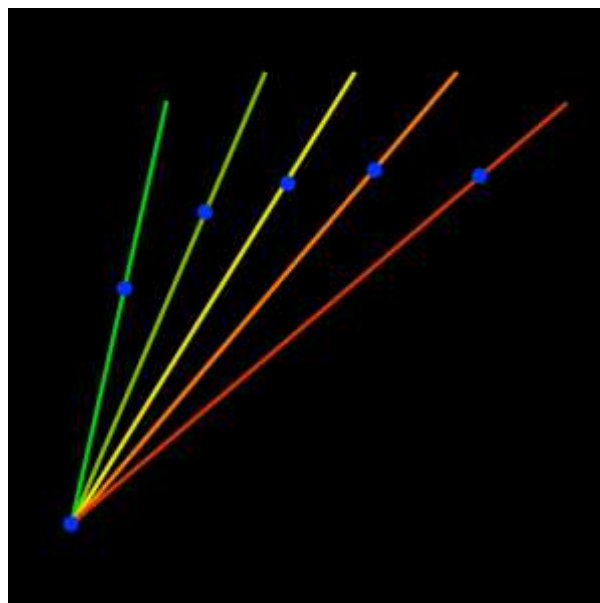


**Fig 2.1** Original image, frame 1290, age 15



**Fig 2.2** Generated image

The next step once all of the latent representations are done is to plot the path of the vectors and to check if and where linearity crumbles on a longer timeframe.



**Fig 3** Proposed experiments for linearity of aging. The black path is ground truth, the blue dots would be actual frames encoded into the latent space. The colored paths are possible linear models. A ratio has to be found between the distance of the original points used for interpolation and the accuracy of it.

In order to do this we need to find an empirical tool that decides how good the linear approximation is.

Before beginning developing a tool to carry out the experiments, we first built an automated pipeline for data collection, cleaning, and processing.

Our pipeline [\[4\]](#) downloads a video, chops it into frames, catalogues them, then throws them into the encoder. This tool has made it significantly easier to try new experiments, as most of the manual work that was done beforehand to get usable data has been automated away.

### 8.3 Linear Interpolation

Once this was done, we built a tool to actually carry out the experiments, and to present them nicely. Given a data set, a start age, and an end age, we create a linear interpolation between the two photos in the latent space. In order to test the fidelity of the interpolation, we see how close it gets to various real data points using the formula  $d = (a - p) - ((a - p) \cdot n)n$ , where  $a$  is the starting point in the interpolation,  $p$  is the point being approximated, and  $n$  is the normalized vector between the start and end of the interpolation. Under this formula,  $p + d$  is the point on the interpolation line closest to  $p$  and is considered to be the interpolation's approximation for the real value at that age.  $d$ 's magnitude (in L2) is the error in the interpolation. Once this is done for all of the data points in the set, we can maybe start to gain some intuition for how reliable of an interpolation we have.

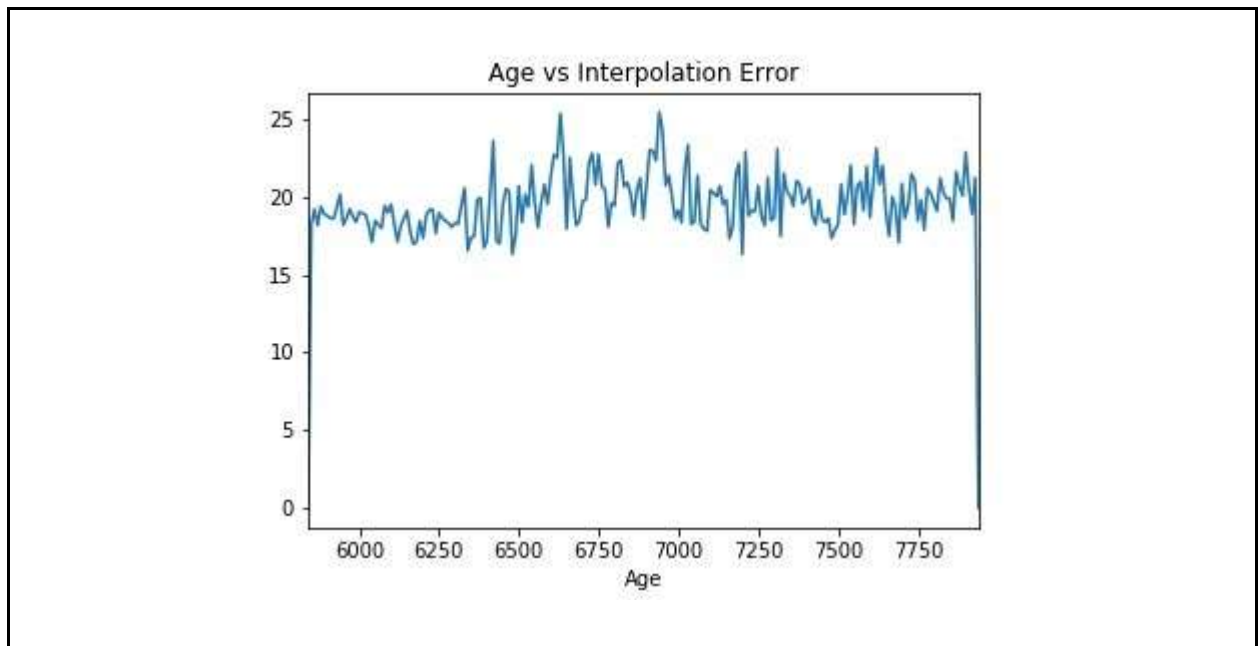
When we carried out experiments, we got strange results that were consistent, but not very precise:



**Fig 3.1** The real aging video, subject ages from 16 to almost 22.



**Figure 3.2** The generated video

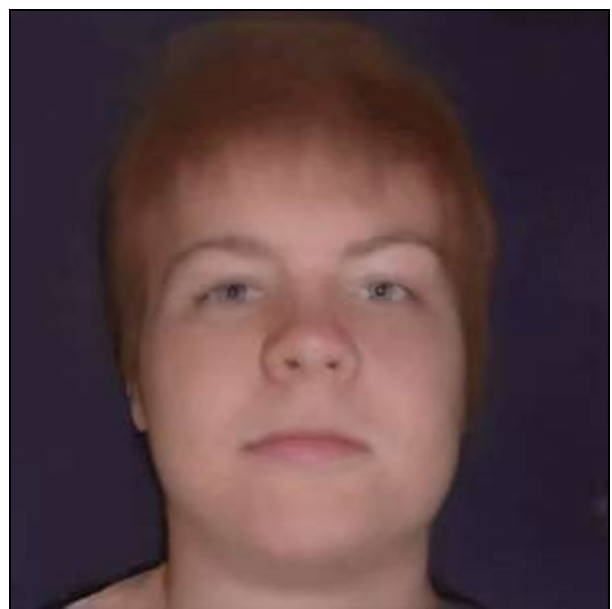


**Fig 3.3** The error for every age in the interpolation range. Interpolation started at 5840 days old, and ends at 7940 days old. The error is 0 in the edge cases because the interpolation is based on those points.

The constant nature of the interpolation threw us off, so we conducted another experiment to verify the results, this time with fewer samples as per Elad's advice.

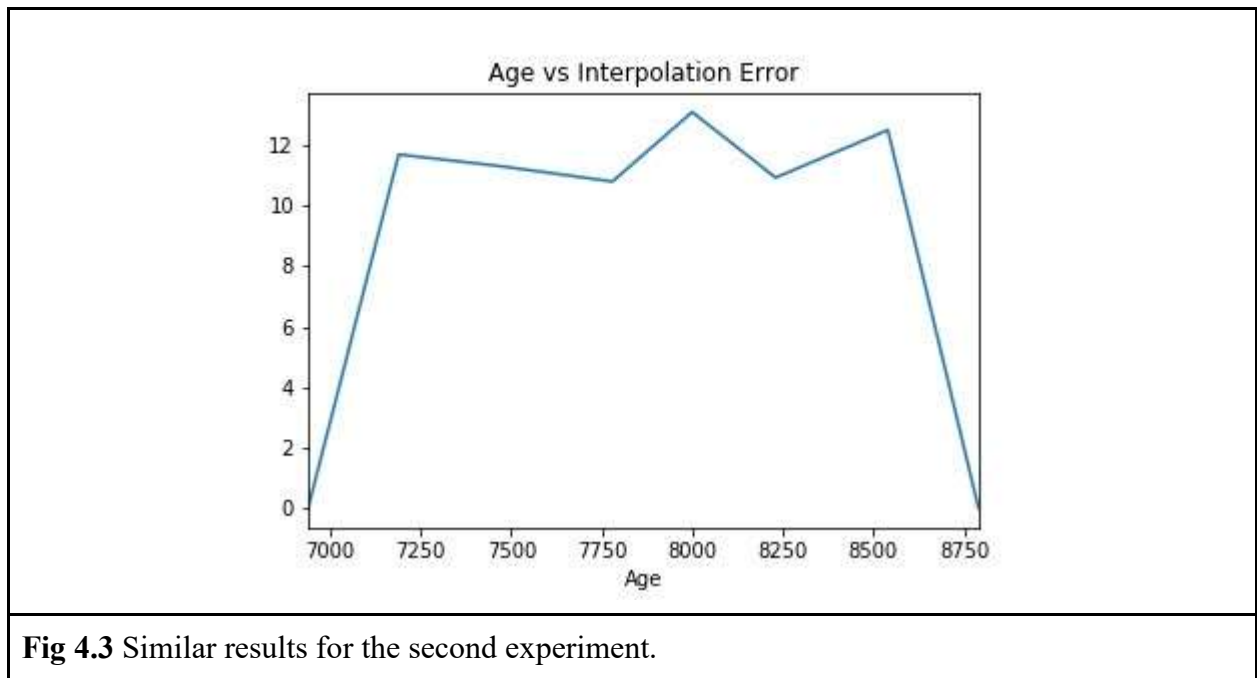


**Fig 4.1** Real aging video. Starts at 15 and ends at 23.



**Fig 4.2** Generated video





After this we considered that maybe we didn't detect a bug in the code, but after thorough review, no bug has been found.

These results are unituitive because it doesn't really make sense that the same point is the closest to all the data points. Additionally, it implies that aging is not at all linear in the latent space.

Further experiments were needed to verify this conjecture and the validity of the tools used. We have added an option to average frames, meaning that given an average ratio, we would use one frame per each set of frames of the average ratio's size. We got similar results here as well (for different ratios).

Next, we checked Puzer's age direction to see its behaviour compared to our interpolation. We used his direction as our interpolator function and got similar results to the previous ones.

This kept repeating over all possible combinations of data, granularity, start age, end age, averaging ratio and interpolation method.

Our conclusion is that there simply is too much noise and/or the feature is still entangled, therefore our next step will be to use a neural network.

## 8.4 Linear Regression, Direct

We attempted to fit a linear regression model to the problem. We used the regression library from scikit learn. The input/output pairs were of the form:

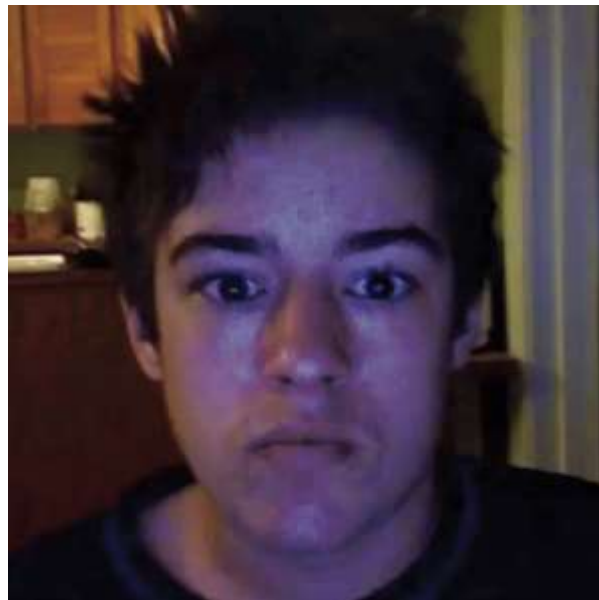
```
(start age, latent representation of start image, target age)/latent
representation of target image
```

Giving equal weight to each of the components (i.e. each of the 51218+2 components from the latent vector and the ages). This is most likely not optimal, and greater weight should be given to the age components, but has not yet been.

We decided to train the model as follows; pick the youngest photo for each person and create pairs with that photo as the start photo and any later photo as the target photo (at the end we need to be able to interpolate continuously onto every age so we thought it would be more helpful to show lots of different target results from a single start photo).

Again, likely not the optimal set of data to train on but other options have not been tested. Maybe a different subset of the combinations is superior (or maybe even training on all combinations).

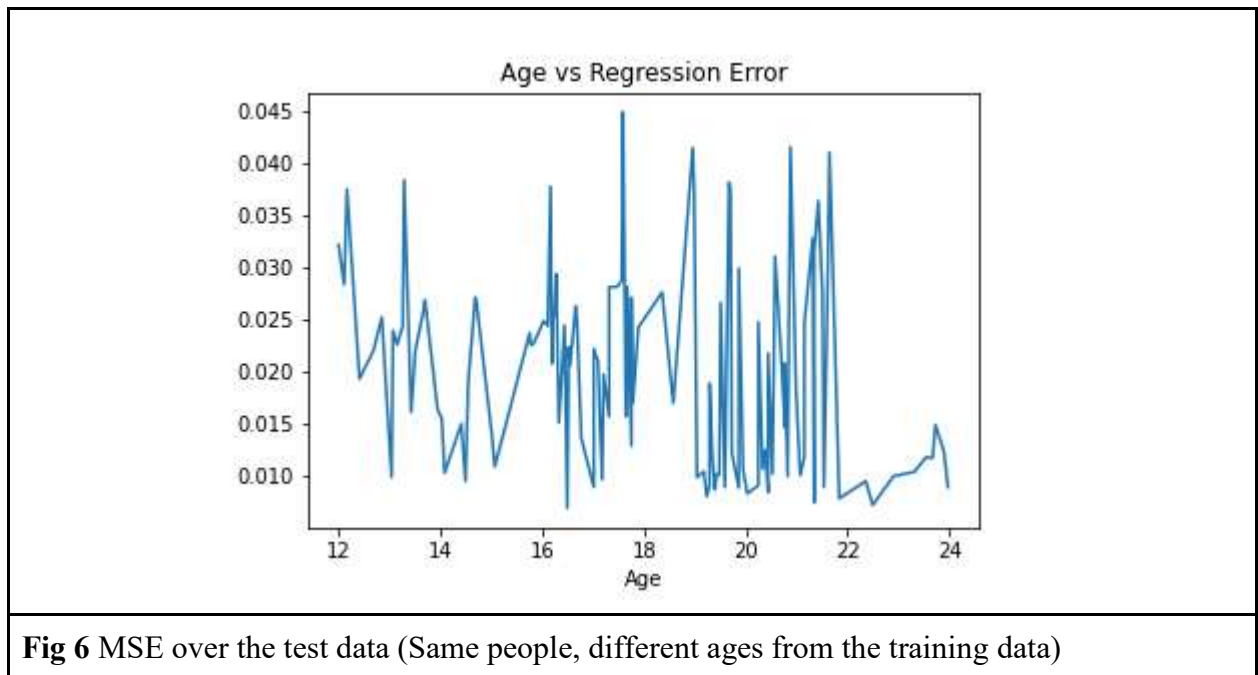
We did the process above for three videos, and then tested the model on one of the videos that we trained on (but different target ages then the ones trained on). The results can be seen below:



**Fig 5.1** Real video



**Fig 5.2** Generated video (aged from 12 to 24)



The results do look lifelike and are relatively close to what happened in reality but the interpolation fails in aging past what was shown in the video and just continues to grow out the man's beard.

We then tested the model on brand new cases (namely Jacob's face). The results can be seen here:



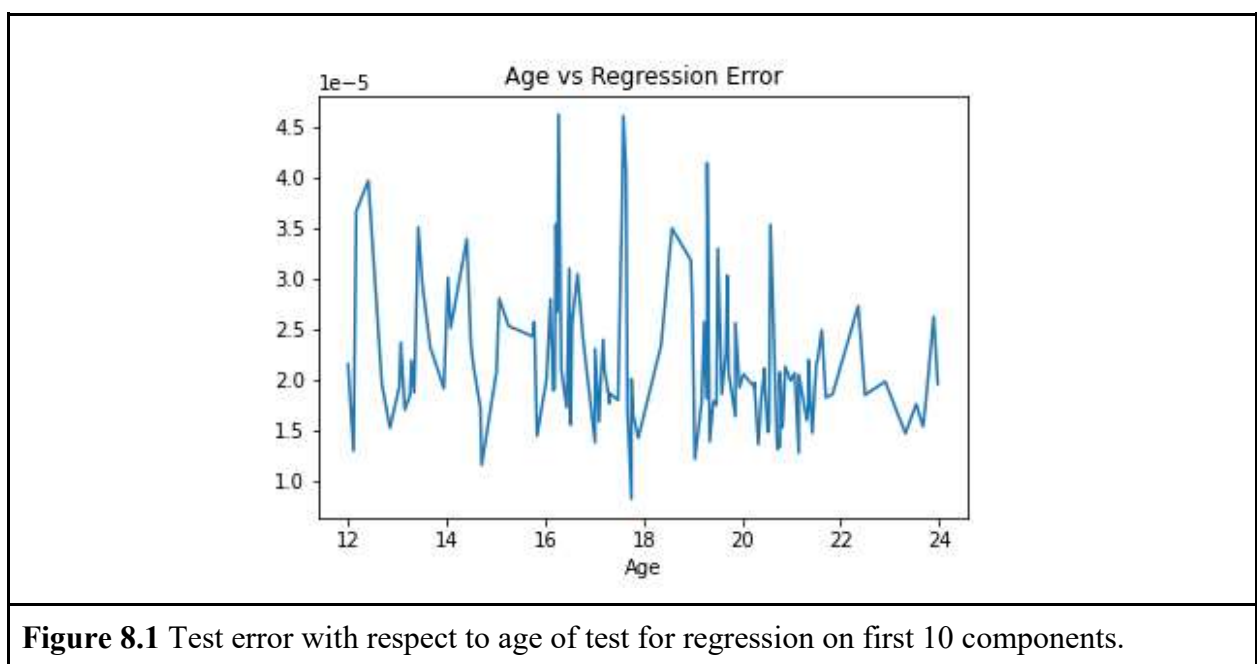
The model can't seem to be able to interpolate on faces it hasn't seen before, instead trying to match them to something learned. We think this is because of a lack of examples and that the solution would be to train the model on many different types of faces (ranging in age, ethnicity, gender, etc.).

## 8.5 Linear Regression, Offset

In order to purify the direction found (to avoid any noise or biases forced into the model by our choice of videos), we decided to only take some of the dimensions of the latent representation, as well as learning the offset rather than the end value. We test out these following assumptions:

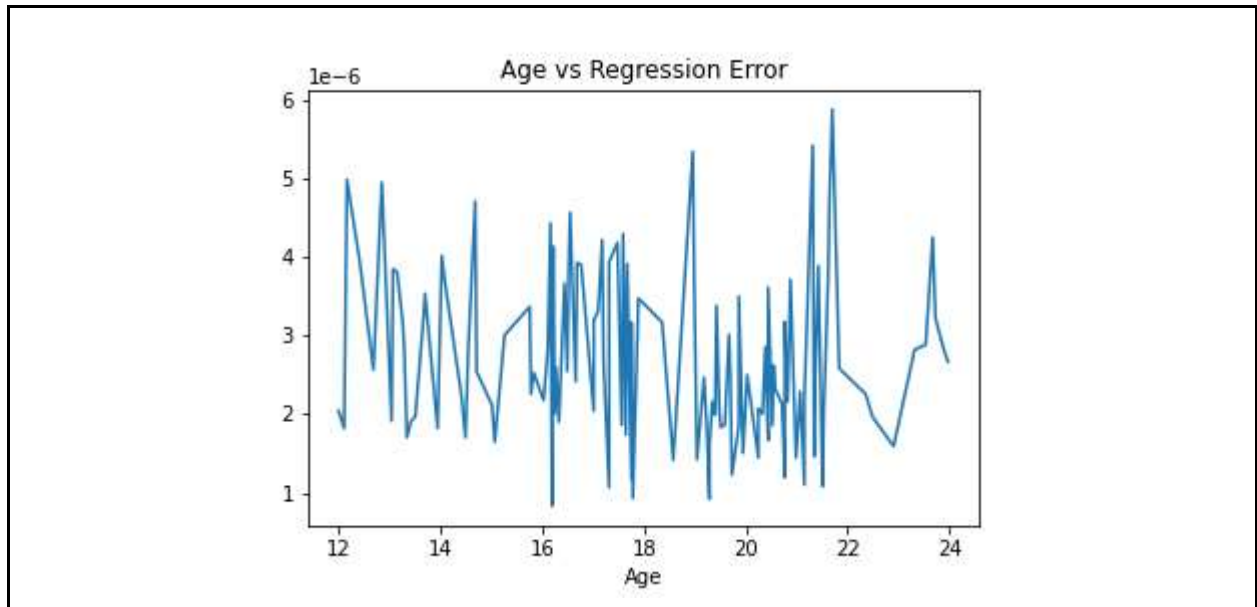
1. The first few components are most likely more significant to the final image (and thus age, a significant feature, should appear there)
2. The components that change more linearly along a sequence of photos of a person aging should be more highly correlated with age (assuming age is indeed linear in stylegan's latent space)

In order to test out the first assumption, we used the same regression model limited to the first 10 components (10 is an arbitrarily decided number and should be tested). The test error can be seen here:



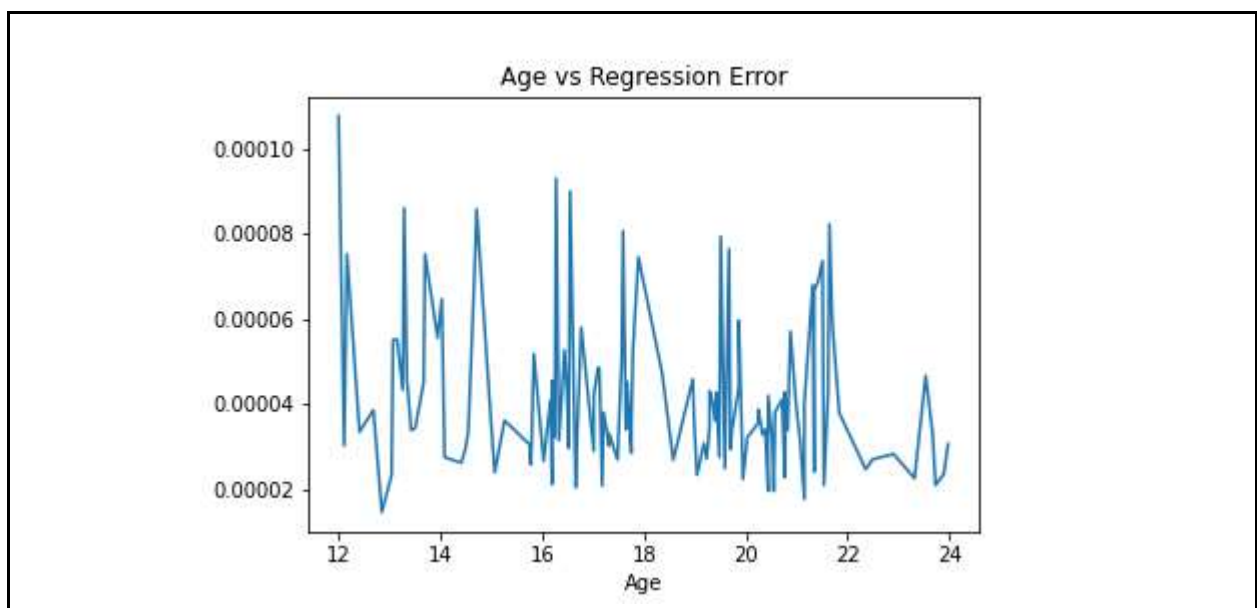
In order to test out the second assumption, we started with a few sequences of photos (evenly spaced out with respect to time). For each such sequence, we found the step sequence (i.e. the sequence of elements such that the sum of an element of a certain index with the element of the same index in the original sequence will yield the the element of the next index in the original sequence), and then broke it down to it's components. For each such sequence (of steps of a single component), we found the variance, and picked the 10 components whose sequences had the least variance.

Surprisingly, the “most linear” components are not in the earlier components but rather they seem to be sporadically spread across the vector. This may be because with nearly 10,000 dimensions some might act linearly by chance, though no statistical significance testing has been done to verify this claim (though it should be, probably as a next step). These were the results when the regression was done over the 10 “most linear” components:



**Figure 8.1** Test error with respect to age of test for regression on 10 “most linear” components.

We additionally tried to normalize the differences (by naively dividing the data set by the largest member), and then doing the variance analysis. Taking the 10 “most linear” dimensions using this method gives us the following results:



**Figure 8.3** Test error with respect to age of test for regression on 10 normalized “most linear” components.

The lowest error is achieved by taking the 10 components with the least variance in step size.

When we actually ran the model to predict aging, the resulting video was almost entirely constant. The issue here, as we later found out, was that the number of components taken into account within the model was too small.



**Figure 8.4** Output of tests for regression on 10 components, using all three methods.

When we saw this result we wanted to verify that we were indeed learning something, so we ran the models on all 51218=9216 components. Since the models all take the first so and so dimensions that have some property, taking all dimensions makes the models equivalent, and so we got the following result for all three models:

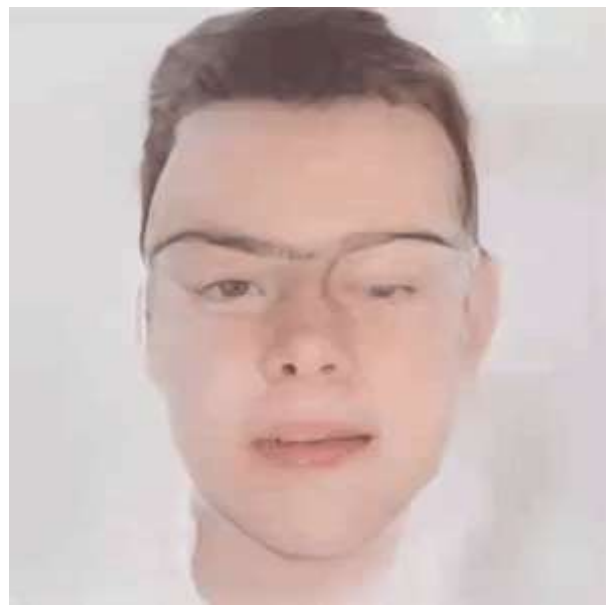


**Figure 9** Output of tests for regression on all components, using all three (equivalent in this case) methods.

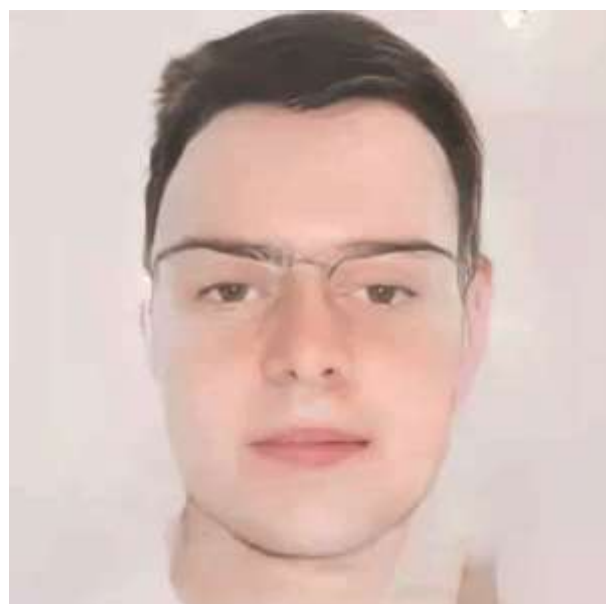
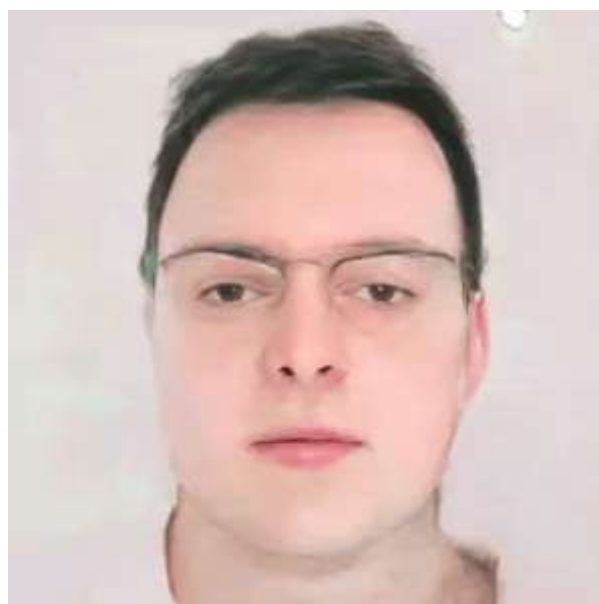
In order to differentiate between the models, we ran another test, this time with half of the components.



**Fig 10.1** Original photo taken.



**Figure 10.2** Output of test for regression on half of the components.





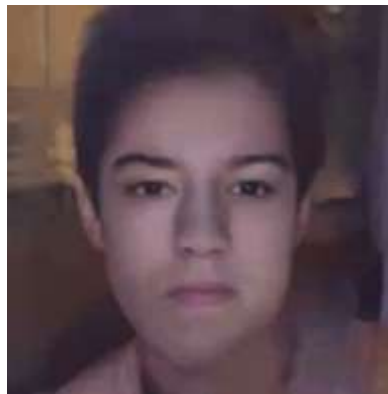
**Figure 10.3** Output of test for regression on “most linear” half of the components.

**Figure 10.4** Output of test for regression on normalized “most linear” half of the components.

The results here definitely differ based on the method used. To better understand each of the models here, we ran the same test on test data (that is, data we used to train the regressions with).



**Figure 10.1** Output of test for regression on half of the components, on internal data.

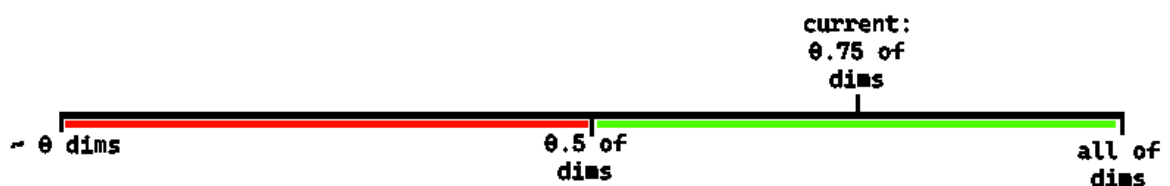


**Figure 10.2** Output of test for regression on “most linear” half of the components, on internal data.



**Figure 10.3** Output of test for regression on normalized “most linear” half of the components, on internal data.






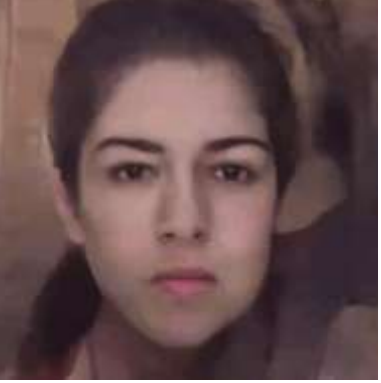
Now it was time for fine-tuning. In search of the *ideal* number of dimensions to use, and the ideal method to use, we’ve decided to conduct some sort of binary-search. This time, we ran a test of 34 of the available dimensions - we figured that the “second half” of dimensions led to better results so far. A demonstration of this can be seen in figure 11.






**Figure 11** Tests conducted so far, and abstract location of current test. Red represents worse results than green. Using binary-search tactics, we chose the green side (referred to above as the “second half”).

We switched back to Jacob’s face for this one, and then immediately conducted the same test on internal data. The results can be seen in figures 12.1-12.6.



		
<p><b>Figure 12.1</b> Output of test for regression on 34of the components.</p>	<p><b>Figure 12.2</b> Output of test for regression on “most linear” 34of the components.</p>	<p><b>Figure 12.3</b> Output of test for regression on normalized “most linear” 34of the components.</p>
		
<p><b>Figure 12.4</b> Output of test for regression on34of the components, on internal data.</p>	<p><b>Figure 12.5</b> Output of test for regression on “most linear” 34of the components, on internal data.</p>	<p><b>Figure 12.6</b> Output of test for regression on normalized “most linear” 34of the components, on internal data.</p>

Next, we wanted to see how the model will behave on a person with darker skin (figure 13.1). This turned out worse than the previous results (As expected), but the aging process can still be easily spotted. This can be seen in figures 13.1-13.3.

		
<p><b>Figure 13.1</b> Original photo taken.</p>	<p><b>Figure 13.2</b> Output of test for regression on 34 of the components.</p>	<p><b>Figure 13.3</b> Output of test for regression on normalized “most linear” half of the components.</p>

## 8.6 MLP

After conducting a few more tests with the previous linear regression model, we have come to the conclusion the method is fairly good. We decided to focus on the simplicity aspect of this method - given a comparatively small data set like ours and a powerful tool like StyleGAN, our linear regression model outputs quick results (it takes about 10 minutes to complete a test pipeline) and these results, as can be seen above, are fairly good.

To strengthen this claim, we decided to train a neural network and benchmark our results according to its outputs.

we tested the following 2-hidden-layer network:



With our hyperparameters being:

1. The data couplings (limitations on the distance between start and target, quantity).
2. Size of hidden layer (10, 100, 1000, 51218).
3. Input features (All 51218 components, 180 most linear components, 180 normalized most linear components).
4. Activation function on hidden-layer (sigmoid, relu, tanh).
5. Learning rate (0.1, 0.01, 0.001).
6. Regularization (none, 11 - 0.001, 11 - 0.0001, 12 - 0.001, 12 - 0.0001).



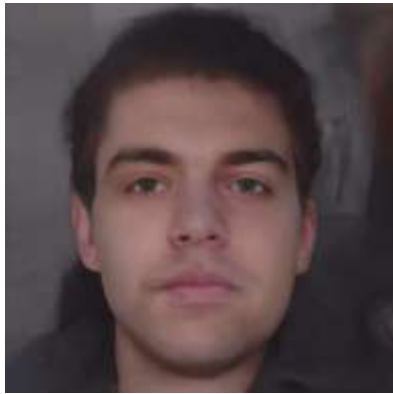
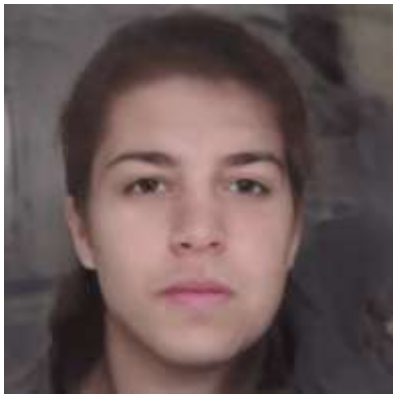
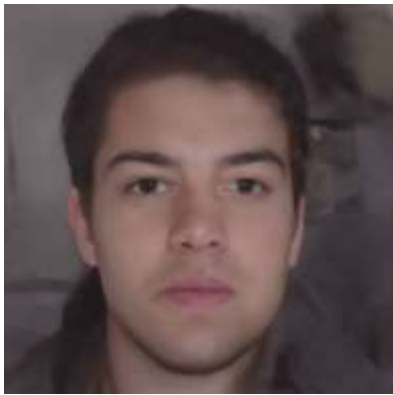
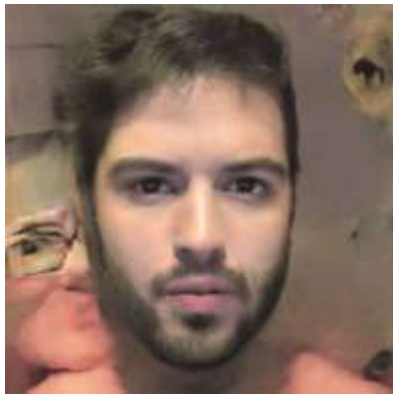
In order to tune our hyperparameters, we performed a grid test for the size, activation, learning rate, regularization, and input features.

We found the optimal data set through random search. We defined our space of possible data points as  $i=13(P_i[\text{age}][\text{age}])P_i$  where  $P_i$  is the set of photos we had for person  $i$ . In each test, we randomly sampled a subset of the data and used it for training.

The model we have found to work best (and subsequently chose as our benchmark) is the following:

1. Data - 6100 samples picked randomly from the above set.
2. Size of hidden layer - 1000.
3. Input features - All 51218.
4. Activation function - *tanh*.
5. Learning rate - 0.01.
6. Regularization - none.

The results we got were very overfitted, proving our initial thesis. Here are a few of the results we got (External tests were conducted on Jacob's original photo):

		
<b>Figure 14.1</b> 10 components, <i>sigmoid</i> , external.	<b>Figure 14.2</b> 100 components, <i>relu</i> , external.	<b>Figure 14.3</b> 1000 components, <i>tanh</i> , external.
		
<b>Figure 14.4</b> 100 forced distance components, external.	<b>Figure 14.5</b> 100 "most linear" forced distance components, external.	<b>Figure 14.6</b> 100 overfitted components, internal.

## 8.7 Puzer's Original Vector

Lastly, we wanted to compare Puzer's original vector to our vector. The result for Jacob can be seen in figure 15. This shows that better outputs are definitely achievable, but not as easily as with our model, since Puzer used a very large data set and complex methods to find his aging vector.



**Figure 15** Puzer's original vector on external data.

These results mark the end of our project, as we have shown our idea is indeed working and useful.

## References

1. "A Photo a Day" - GitHub repository: <https://github.com/shavitborisov/APhotoADay>.
2. Tang, Xu, et al. "Face Aging with Identity-Preserved Conditional Generative Adversarial Networks." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, doi:10.1109/cvpr.2018.00828.
3. "Video 1": [https://www.youtube.com/watch?v=zuRd\\_Eneuk8](https://www.youtube.com/watch?v=zuRd_Eneuk8).
4. "Video 2": <https://www.youtube.com/watch?v=LtpnjJlc4r8>.
5. "Video 3": <https://www.youtube.com/watch?v=65nfbW-27ps>.
6. Puzer's StyleGAN Encoder - GitHub Repository: <https://github.com/Puzer/stylegan-encoder>.
7. FFmpeg: <https://www.ffmpeg.org/>.
8. CS231n Convolutional Neural Networks for Visual Recognition: <https://cs231n.github.io/>.
9. YouTube-DL: <https://ytdl-org.github.io/youtube-dl/index.html>
10. OpenCV: <https://opencv.org/>.
11. Official Google Doc: <https://docs.google.com/document/d/1DC6SfjVyKHy0BKcRJRJV06X712L1GD00MMX9OSS-u2E/edit?usp=sharing>.
12. Dosovitskiy, Alexey, et al. "Learning to Generate Chairs, Tables and Cars with Convolutional Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016, pp. 1–1, arxiv.org/pdf/1411.5928.pdf, 10.1109/tpami.2016.2567384. Accessed 25 Apr. 2020.
13. Ma, Liqian, et al. *Disentangled Person Image Generation*. CVPR, 2018.

14. Zhu, Jun-Yan, et al. *Visual Object Networks: Image Generation with Disentangled 3D Representation*. 32nd Conference on Neural Information Processing Systems, 2018.
15. Ankita Shukla, Shagun Uppal, Sarthak Bhagat, Saket Anand, and Pavan Turaga. 2018. Geometry of Deep Generative Models for Disentangled Representations. In Proceedings of ICVGIP (ICVGIP'18). ACM, New York, NY, USA, 8 pages.
16. Bengio, Y., LeCun, Y., et al. Scaling learning algorithms towards ai. Large-scale kernel machines, 34(5):1–41, 2007.
17. Wang, Zongwei, et al. *Face Aging with Identity-Preserved Conditional Generative Adversarial Networks*. 2018.
18. Antipov, Grigory, et al. *Face Aging with Conditional Generative Adversarial Networks*. 30 May 2017.
19. Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166.
20. A Style-Based Generator Architecture for Generative Adversarial Networks: <https://www.youtube.com/watch?v=kSLJriaOumA&t=>.