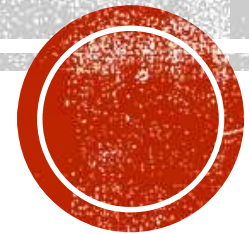


LOW COMPLEXITY DATA-EFFICIENT GENERATION IN DISENTANGLED SETTING

Project by Jacob Sela and Shavit Borisov

Supervised by Elad Richardson





Recently, much work has been done on the topic of generation from a disentangled space, 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.

INTRODUCTION



OBSTACLES

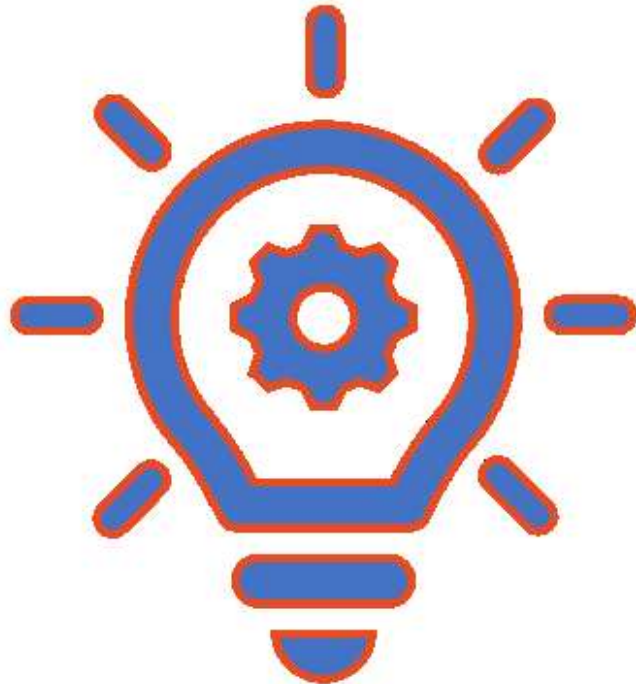


However, training an autoencoder for the disentanglement of a feature is expensive in time (both computationally and that put in by the engineer of the system) and resources (a highly varied and expansive data set is required).



This sets a barrier, allowing only those highly trained in such systems and equipped with a lot of data to effectively evoke these models for generation.





IDEA

- One would expect, however, that a sufficiently disentangled space would isolate many features to a reasonably linear degree.
- This would allow 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.



PROPOSAL



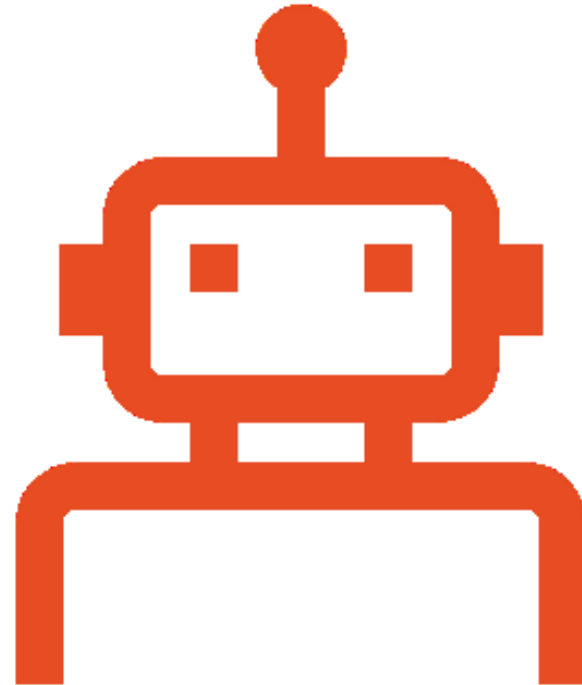
IN THIS WORK

- 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.
- 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.

WHY AGING?

Aging was picked for two reasons:

1. Aging data is readily and freely available online by looking up “A Photo a Day” on YouTube.
2. Aging is useful to demonstrate the limited data needed to learn the distribution. Many previous works on aging relied on a much more expansive data set, while we mainly use 3 videos to achieve our results.

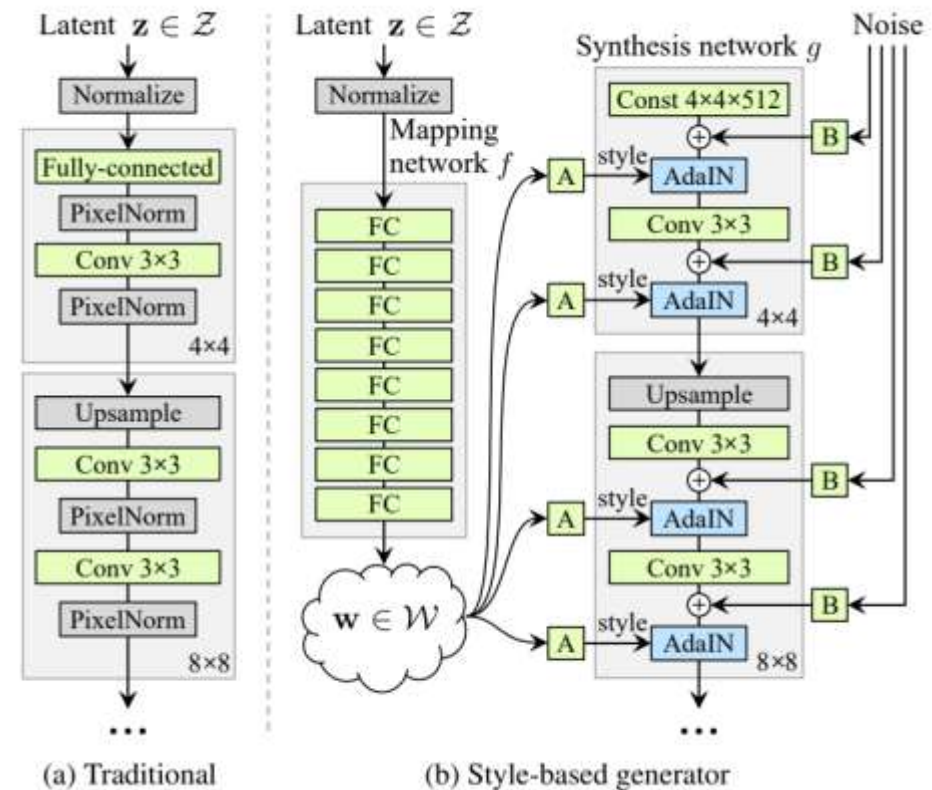


STYLEGAN

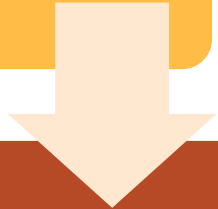
StyleGAN is a “style based” generative adversarial network.

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.



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 TO ENCODE OUR IMAGES AND RECEIVE OUR LATENT REPRESENTATIONS.

STYLEGAN





Puzer's encoder is a 3rd party deep-learned tool to encode photos into StyleGAN's latent space.



We use this tool to encode our images.



Additionally, Puzer learned a direction for aging in StyleGAN's latent space using a 2-layer hidden network; this was the inspiration for the project.

PUZER'S ENCODER





“A Photo a Day” videos from YouTube.



Download using YouTube-DL.



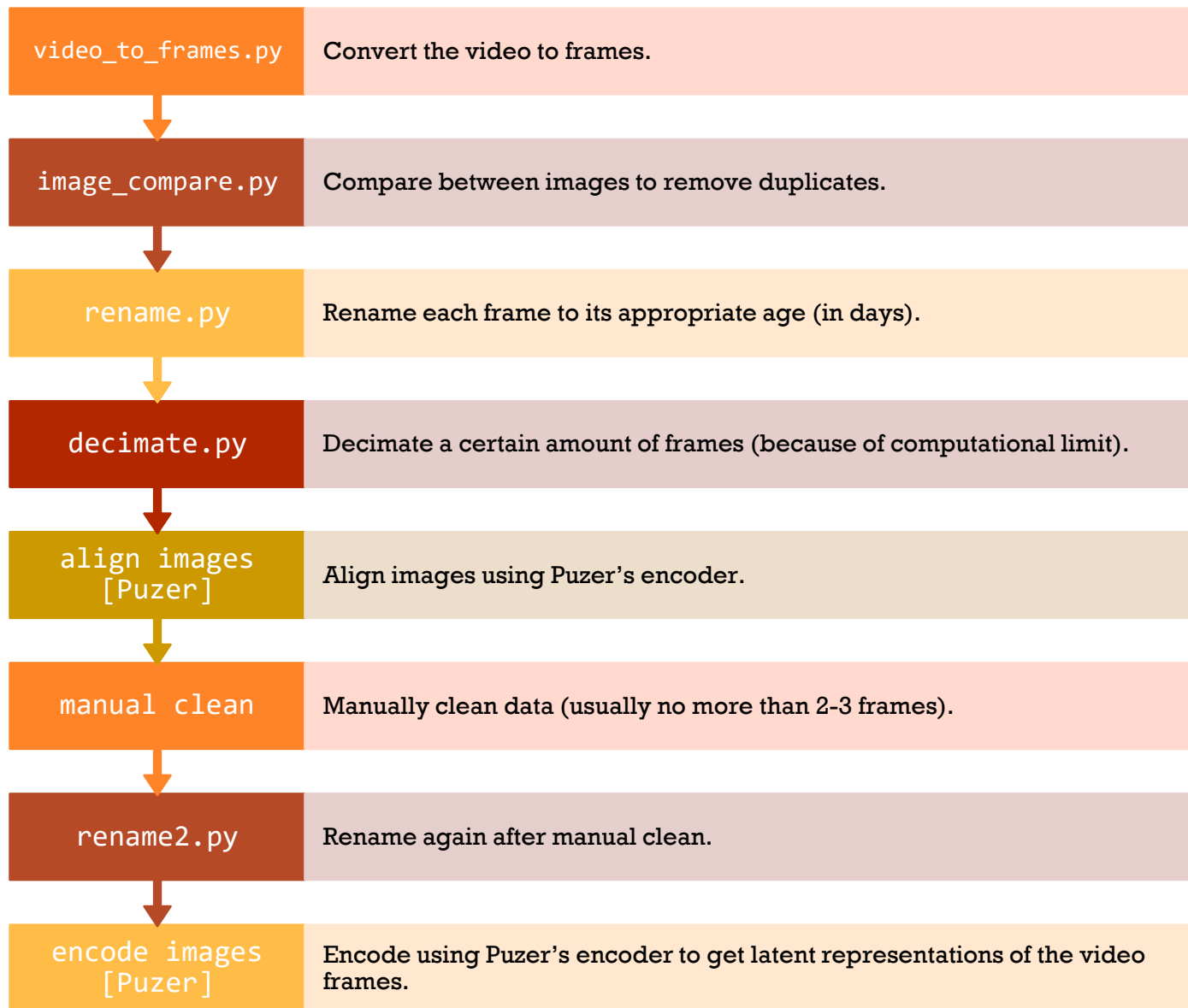
Cutting videos to frames using OpenCV.



Comparing frames based on the L0 norm.

EARLY TOOLS





DATA COLLECTION



AUTOMATION AND GIF GENERATION



Receive input for the test with relevant parameters and data.



Train the model according to the relevant parameters.



Test the model on internal and external data.



Create an "Error vs. Age" chart.



Predict latent representations of aging video frames for a given test subject.



Convert latent information to actual images using StyleGAN's decoder.



Create an aging GIF of real and fake data, side by side, using the FFmpeg tool.



SIMPLE LINEAR INTERPOLATION

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:

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

This model was tested on:

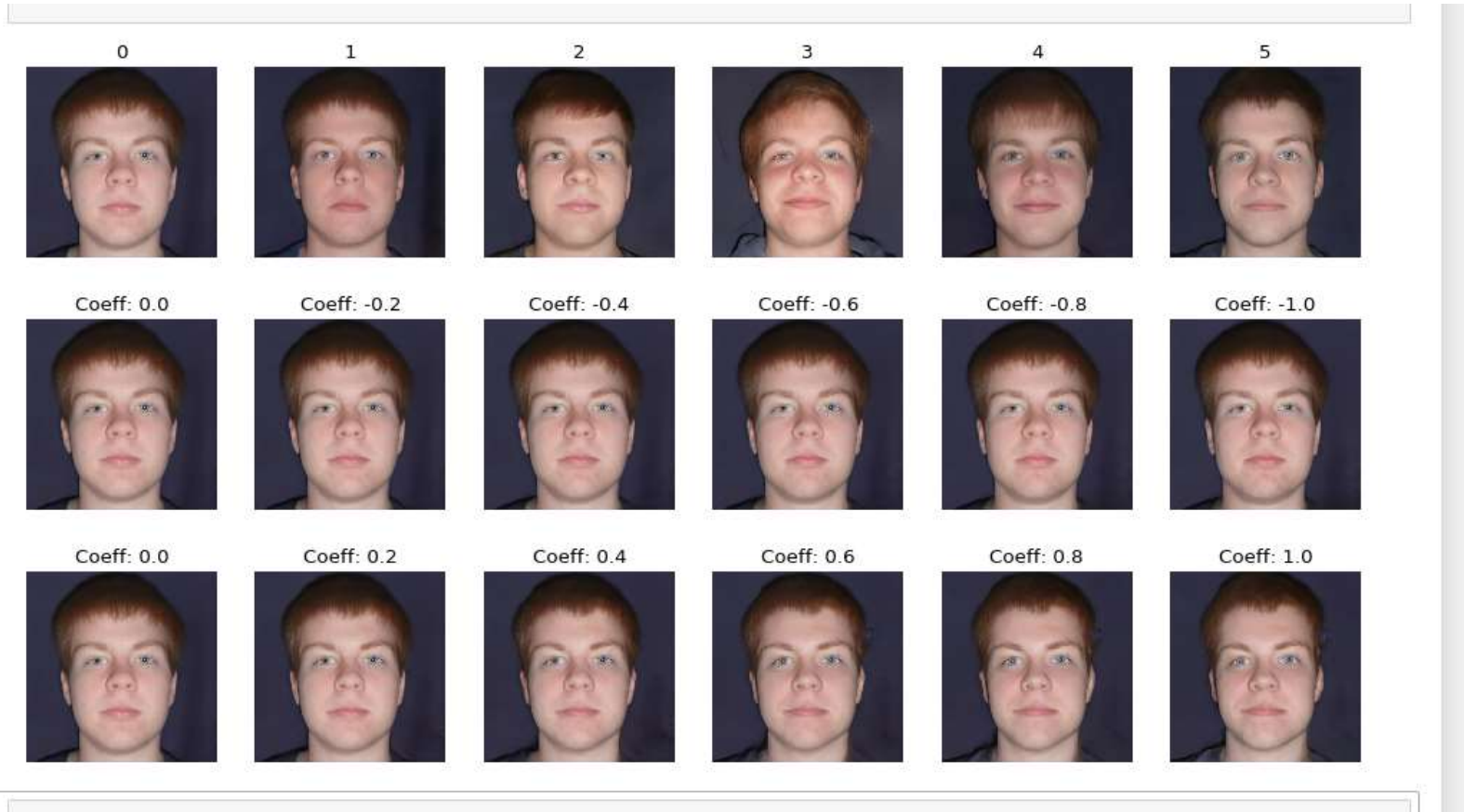
- Start and end points at different distances apart.
- A subset of the latent space components.
- Averaging of multiple photos to find the initial p_1 and p_2 .

Error was measured in MSE between the interpolated photo and the original photo at that age in the latent space.

This is also the case for all other models (unless stated otherwise).



RESULTS



Results were obscure and led to the decision to move on to a more complex model than linear interpolation (by Elad's advice).



SUM UP

LINEAR REGRESSION, DIRECT

We tested a linear regression model fitted on linear least squares.

Given a set of n data points:

$$\{((startPhoto_i, startAge_i, targetAge_i), targetPhoto_i)\}_{i=1}^n$$

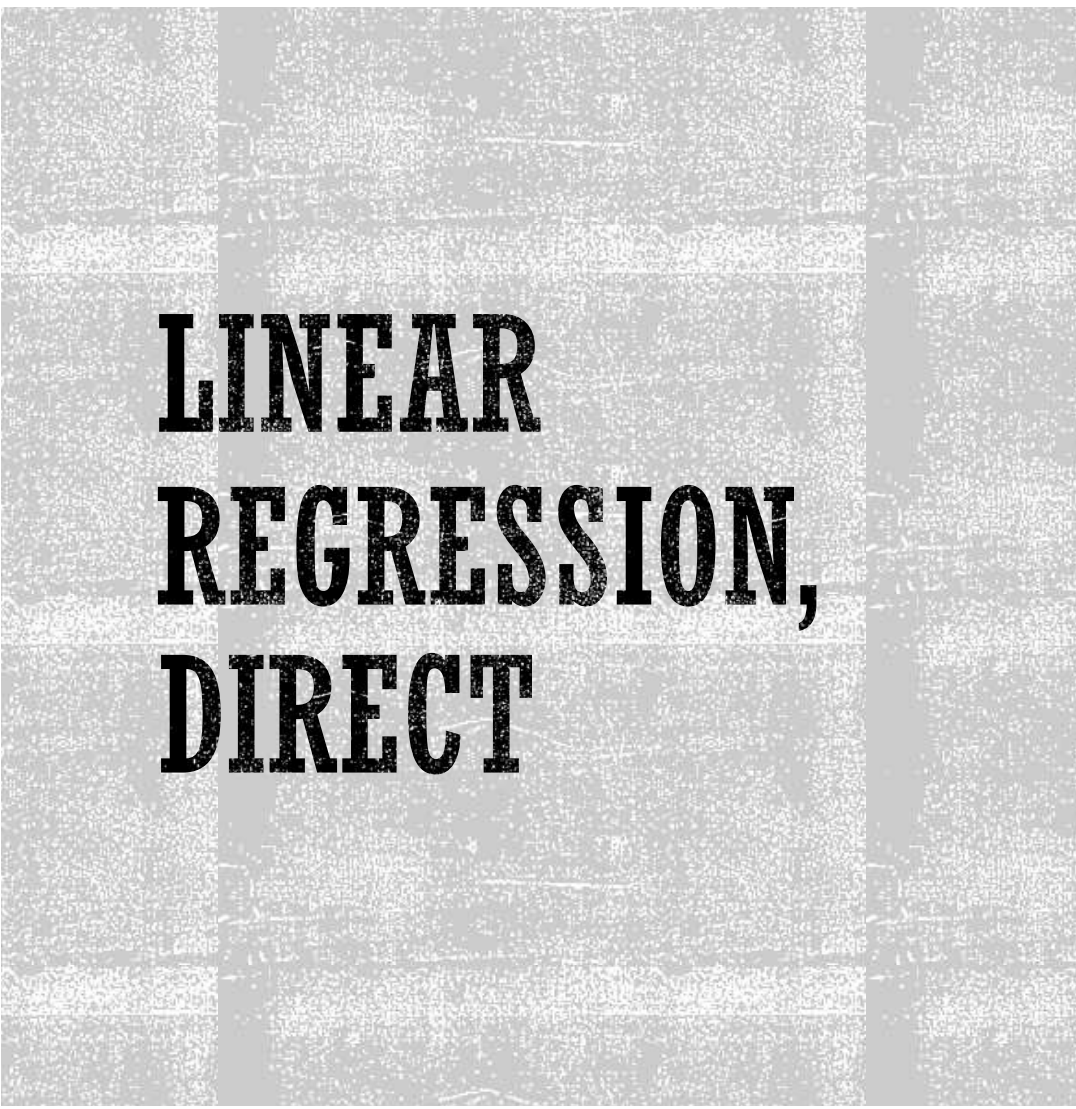
We found the function:

$$f(startPhoto, startAge, targetAge) = targetPhoto$$

such that we minimize:





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

- We trained the model to learn the result directly.
- Using our videos we produced the data set 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.
 - The process was done for three videos.
 - Use all previous tools to conduct multiple tests.



**LINEAR
REGRESSION,
DIRECT**

RESULTS

| | Original | Result |
|---------------|--|--|
| Internal Data |  |  |
| External Data |  |  |



- While results were good when using the model on one of the videos that we trained on (with different target ages than the ones trained on), the result on external data was far from optimal.
- We decided to try to learn the offset rather than the result.



LINEAR REGRESSION, OFFSET

- In order to find a better fit, we trained for offset instead of direct interpolation.
- Based on a linearity test we devised, we added a few more weighting options for the different components.
- 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 $\{mostLinear\}_{i=1}^{512 \cdot 18}$ and the normalized sorted indices list $\{normalizedMostLinear\}_{i=1}^{512 \cdot 18}$.

LINEAR REGRESSION, OFFSET



512 · 18 components



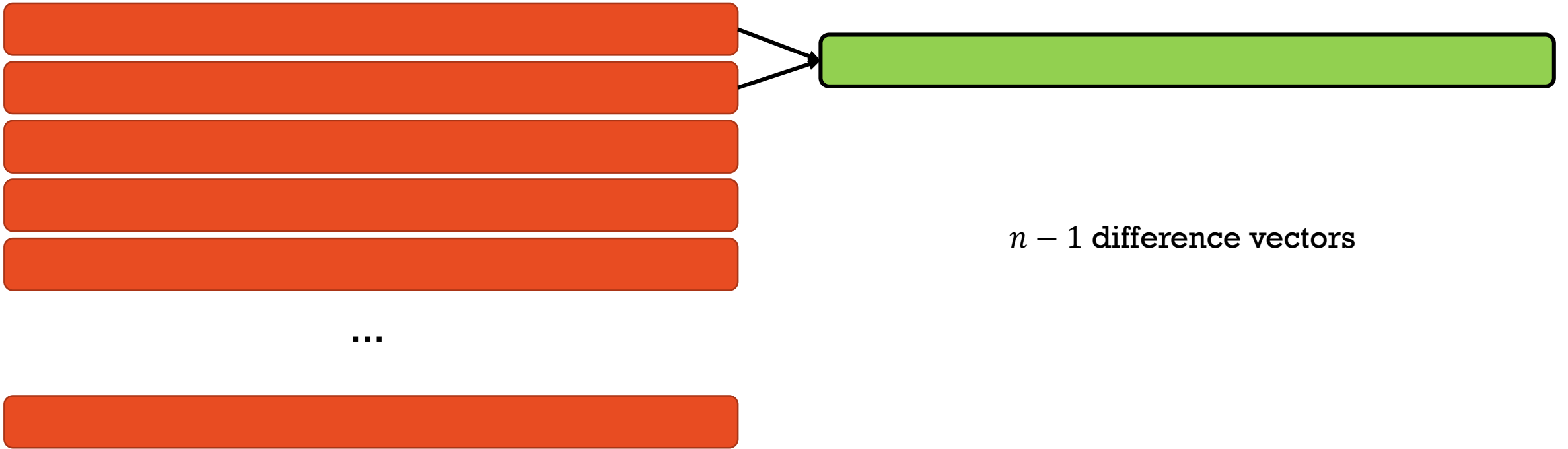


k components



n vectors

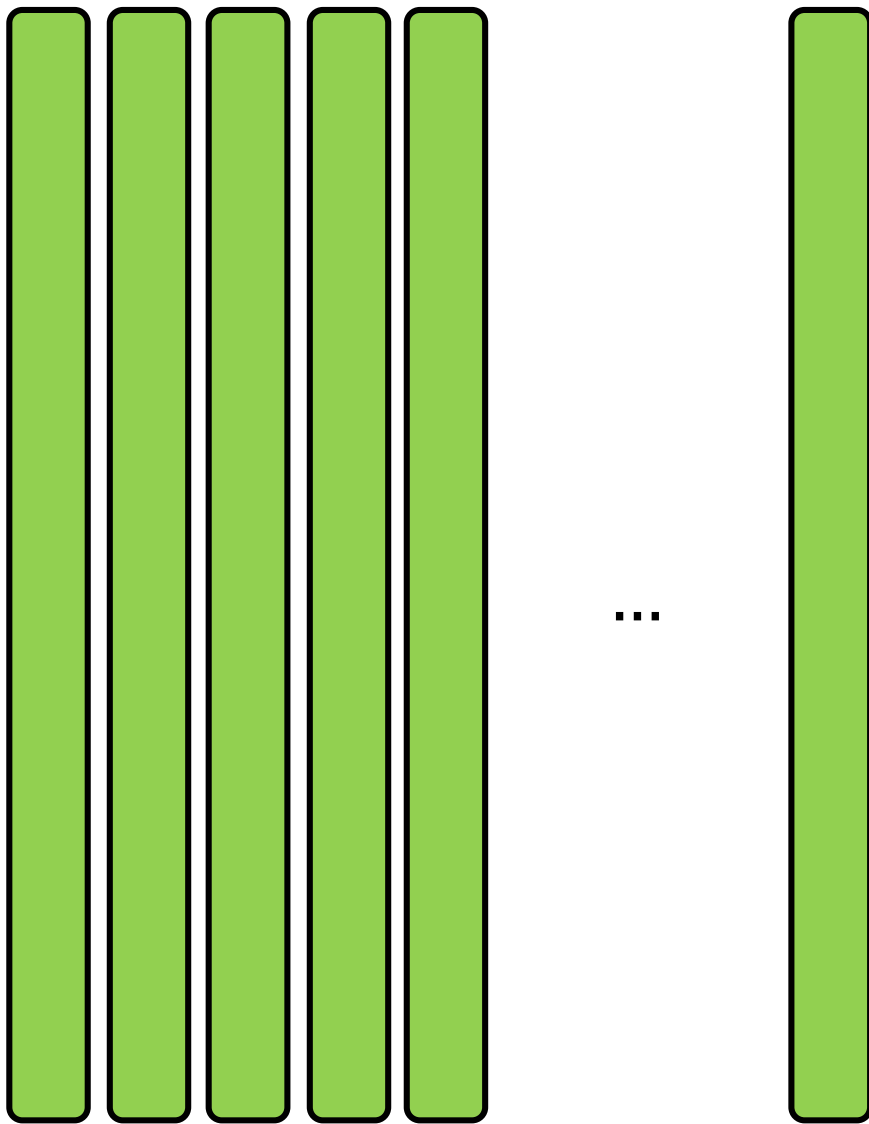


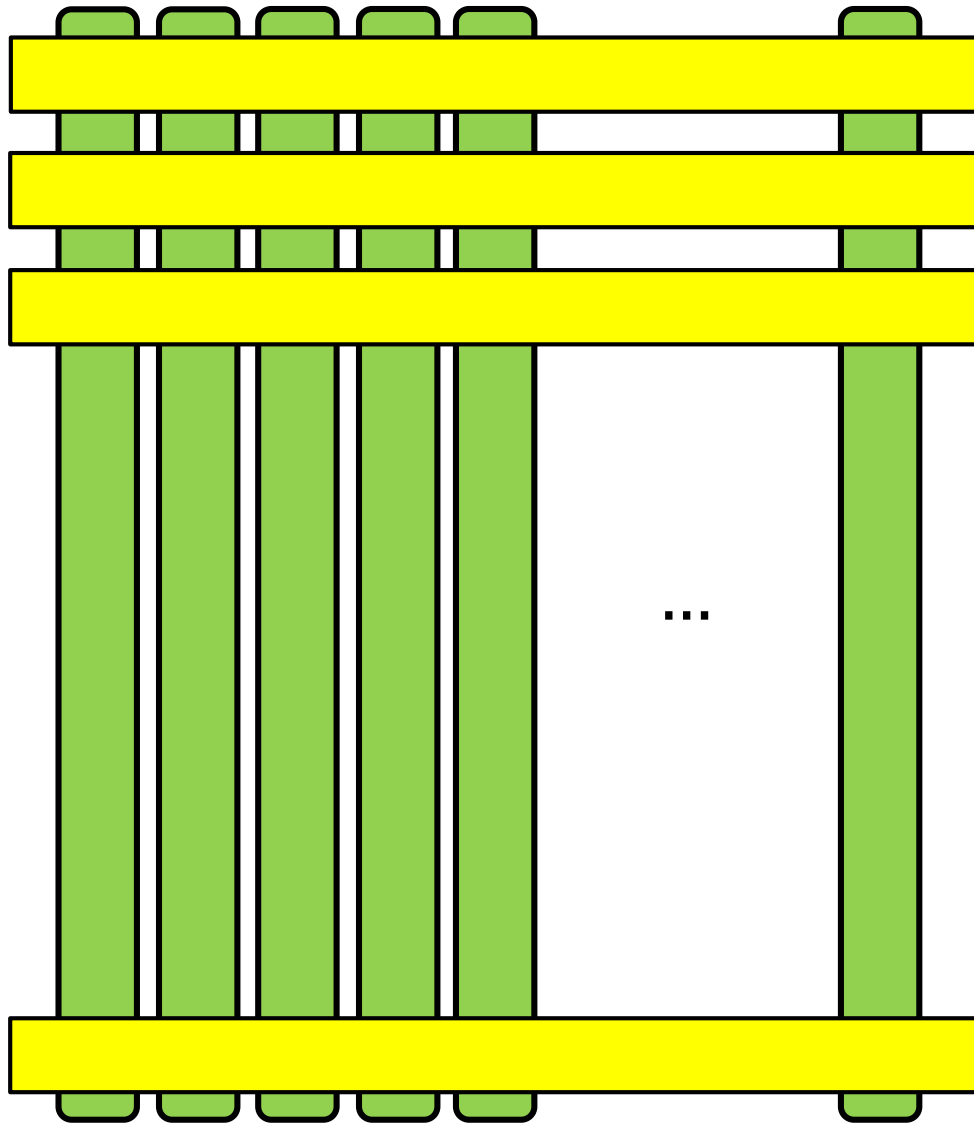




...







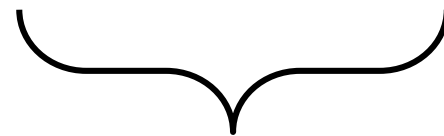
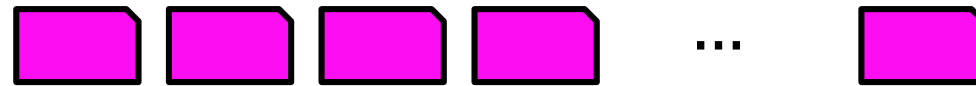
512 · 18 sequences of $d_i, \forall i \in [n - 1]$





$n - 1$ variances of
members





k most linear
components





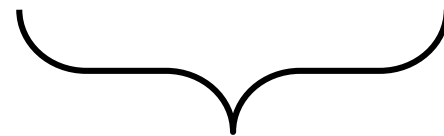
↑
Maximal
component





$n - 1$ variances of
members





k normalized
most linear
components






This, along with a limitation on the number of dimensions taken for the regression, gives us 3 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).

LINEAR REGRESSION, OFFSET

RESULTS

| | | |
|---|--|---|
|  |  |  |
| Jacob | Shavit | Earl |



RESULTS

Relevant GIFs can be found in the [report](#).



- We can certainly see aging.
- We now need to benchmark our results.



MLP

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

- The data couplings (limitations on the distance between start and target, quantity).
- Size of hidden layer (10, 100, 1000, 512·18).
- Input features (All 512·18 components, 180 most linear components, 180 normalized most linear components).
- Activation function on hidden-layer (*sigmoid*, *relu*, *tanh*).
- Learning rate (0.1, 0.01, 0.001).
- Regularization (none, 11 - 0.001, 11 - 0.0001, 12 - 0.001, 12 - 0.0001).



MLP

- 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:

$$\bigcup_{i=1}^3 (P_i \cup [age] \cup [age]) \times P_i$$

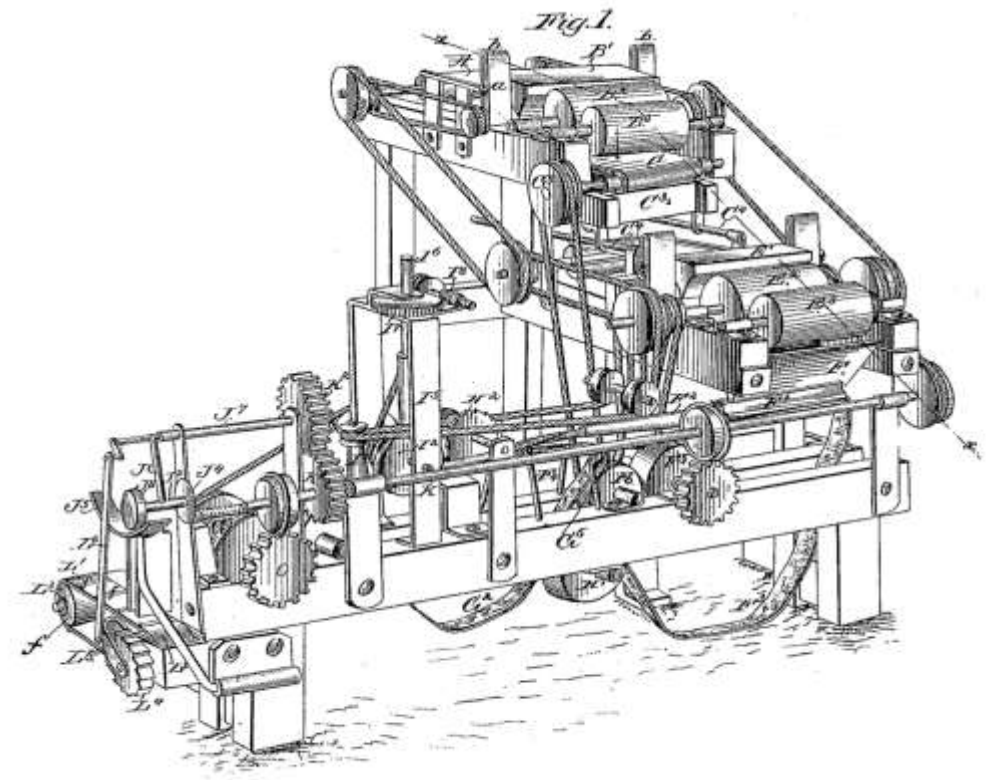
Where P_i is the set of photos for person i .

In each test, we randomly sampled a subset of the data and used it for training.

MLP

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

- Data - 6100 samples picked randomly from the above set.
- Size of hidden layer - 1000.
- Input features - All 512·18.
- Activation function - *tanh*.
- Learning rate - 0.01.
- Regularization - none.



RESULTS

Relevant GIFs can be found in the [report](#).



PUZER'S VECTOR

Finally, we wanted to test Puzer's original vector to again benchmark our results, this time against a known, working, deep-learned direction.



RESULTS



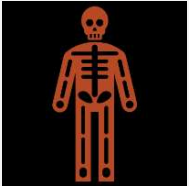
- As we can see, our results beat a naively put together and trained MLP.
- Unsurprisingly, it can't beat Puzer's 2-hidden-layer learned direction (an example of a well put together and trained MLP).
- Although the results weren't perfect, they can be significantly improved with minor changes (will be discussed later).



SUM UP



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 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.


RESULTS ANALYSIS



We achieved this goal with arguably *simpler* methods than those of other latent interpolation techniques.



We rely only on a linear function to interpolate rather than the more common piecewise linear or slerp and using the comfortably familiar L2 as our metric.



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.

PROS



- 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.
- It 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.



BEST RESULT

RESULTS ANALYSIS

- When we expand the number of components taken 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.
- This model performs *better* than the naive MLP, indicating that we succeeded in our initial goal.



BENCHMARK



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.



CONCLUSION

- 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.



OVERVIEW



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.



We 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.

THESIS



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.
- 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.

NEXT STEPS



THANK YOU!



Special thanks to Elad Richardson for supervising this project.

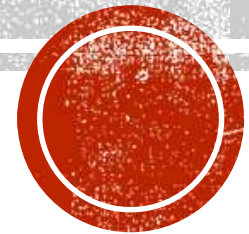


A PHOTO A DAY

MID-PRESENTATION

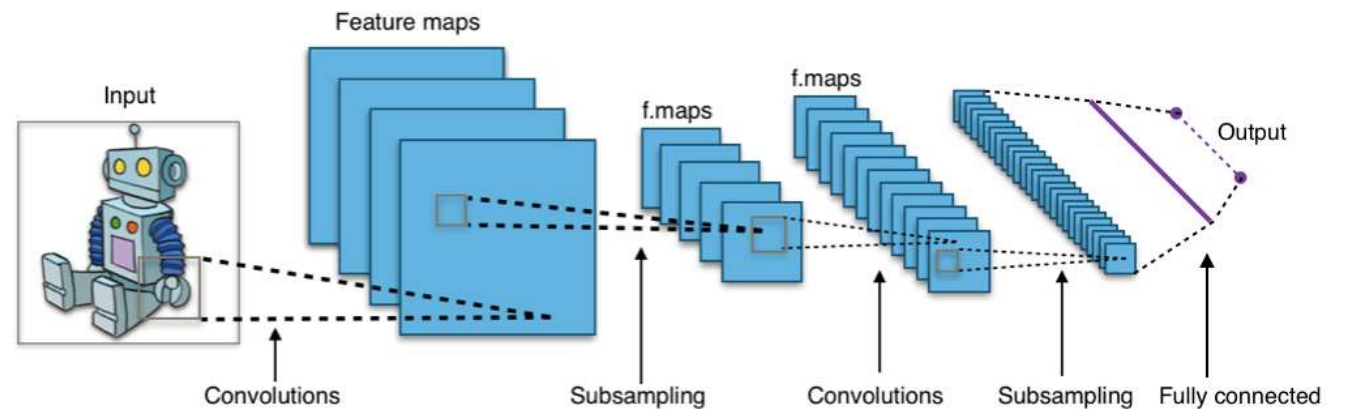
Project by Jacob Sela and Shavit Borisov

Supervised by Elad Richardson



CONVOLUTIONAL NEURAL NETWORKS

- Neural Network archetype useful for image analysis.
- Distinguishing features:
 - Convolutional layers.
 - Local Connectivity.
 - Pooling.
 - Shared Weights.
- Strengths:
 - Translational Invariance.
 - Smaller than fully-connected networks.
 - Weight sharing further minimizes weights to be learned.
- This makes them well suited for vision problems.

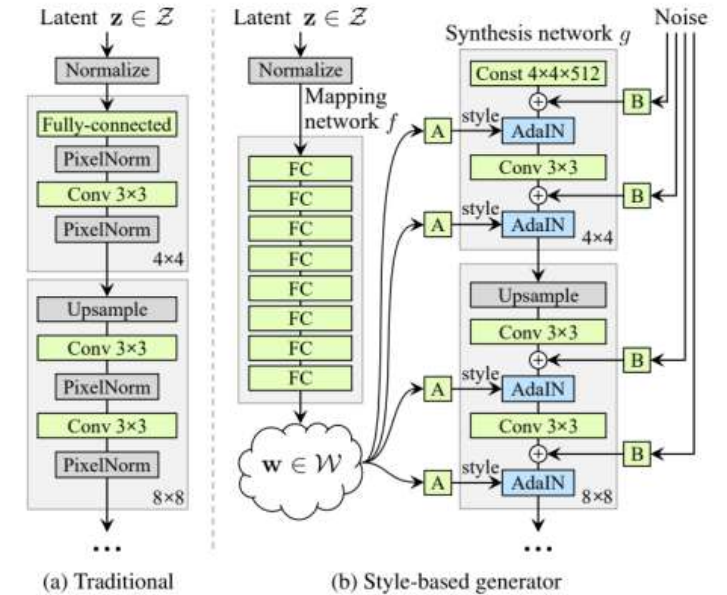


- Made up of:
 - Generative network – suggests candidates.
 - Discriminative network – decides if candidate is within distribution.
- Each learns to increase the other's loss in a 0-sum game.
- Unsupervised.
- Can be used in image synthesis.

GENERATIVE ADVERSARIAL NETWORKS

STYLEGAN

- Nvidia's GAN.
- Proposes new architecture for generator.
 - Intermediate latent space (independent of the data distribution).
 - Latent representation (style) is fed at every layer.
 - Stochastic noise at every layer.
- Latent space is disentangled.
- Style mixing allows generation of images with features from multiple sources.





Given a photo, generate an aging video/a-photo-a-day.



This could be possible using existing tools such as StyleGAN.



We would like to (eventually) do this through supervised learning.

PROJECT IDEA



FINDING OUR FEET

- We started by trying to think of ways to automate the data collection process.
- Downloading videos – youtube-dl
- Photo/video manipulation – OpenCV
- Image Comparison – using L0 norm





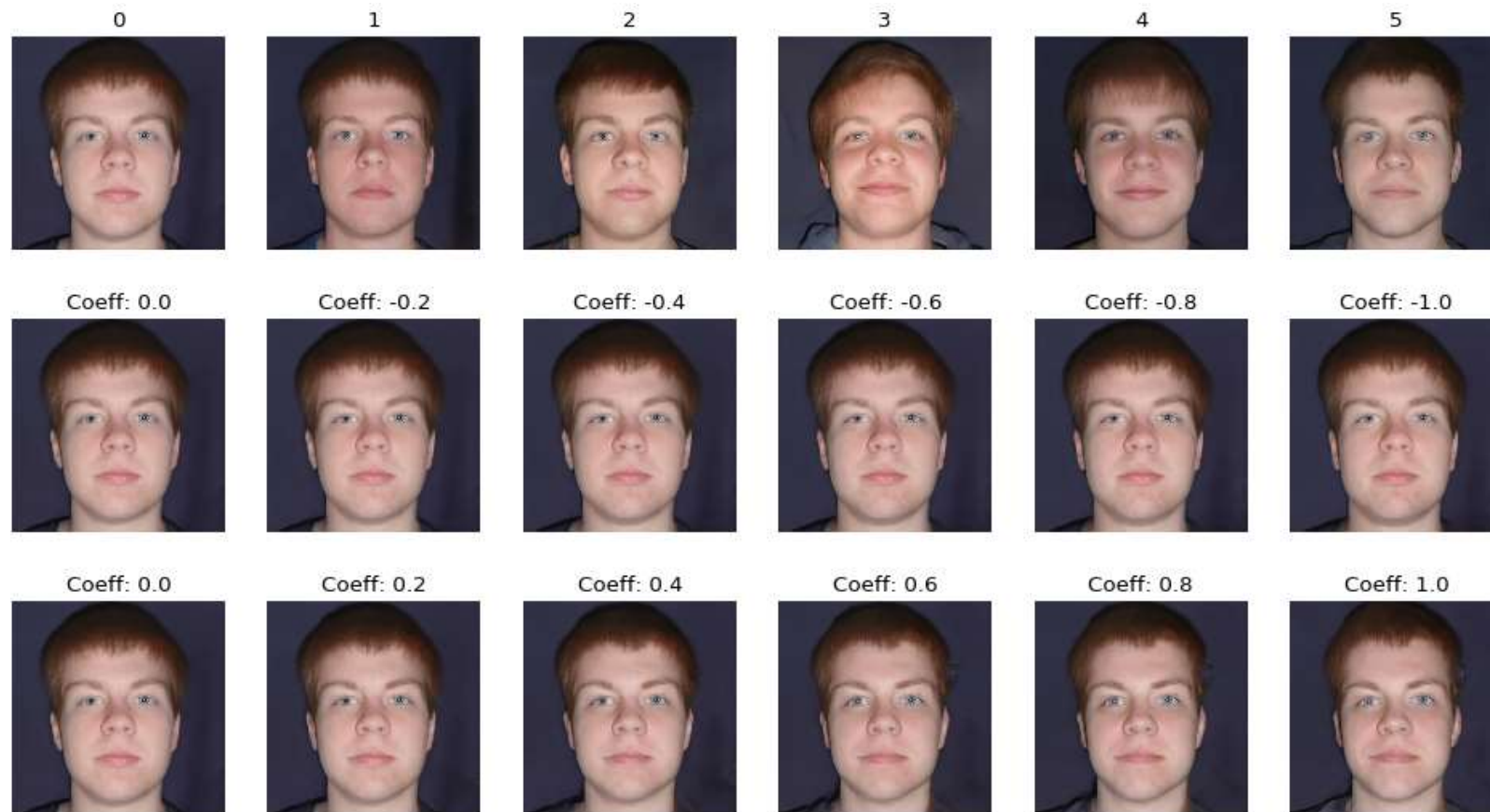
TESTING LINEARITY

- Before using neural networks and more advanced techniques we wanted to gain some intuition.
- Additionally, we wanted to practice abduction.
- Features should be linear in StyleGAN's latent space, at the very least continuous.
- We wanted to find a way to test this.

FIRST EXPERIMENT

- Given two data points, how well do they approximate the rest of the data between them?
- We take two real points, and evenly interpolate between them, compare that to real data, and Puzer's latent direction
- If the data is linear, the interpolation should be relatively accurate





EXP. 1 RESULTS

Top row is truth.

Mid row is Puzer's latent direction.

Bottom row is our linear interpolation.





Given two real data points, how well do they approximate the rest of the data between them?



Across different amounts of time?



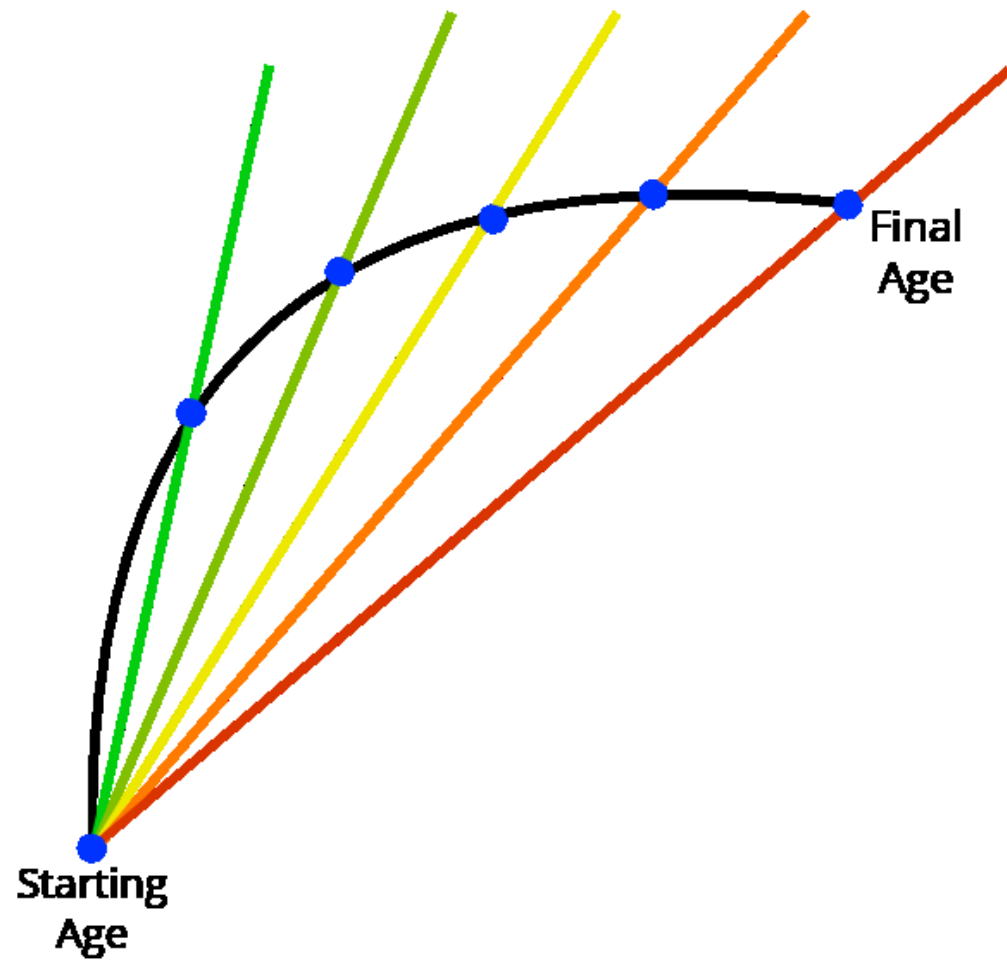
Across different people?



Etc...

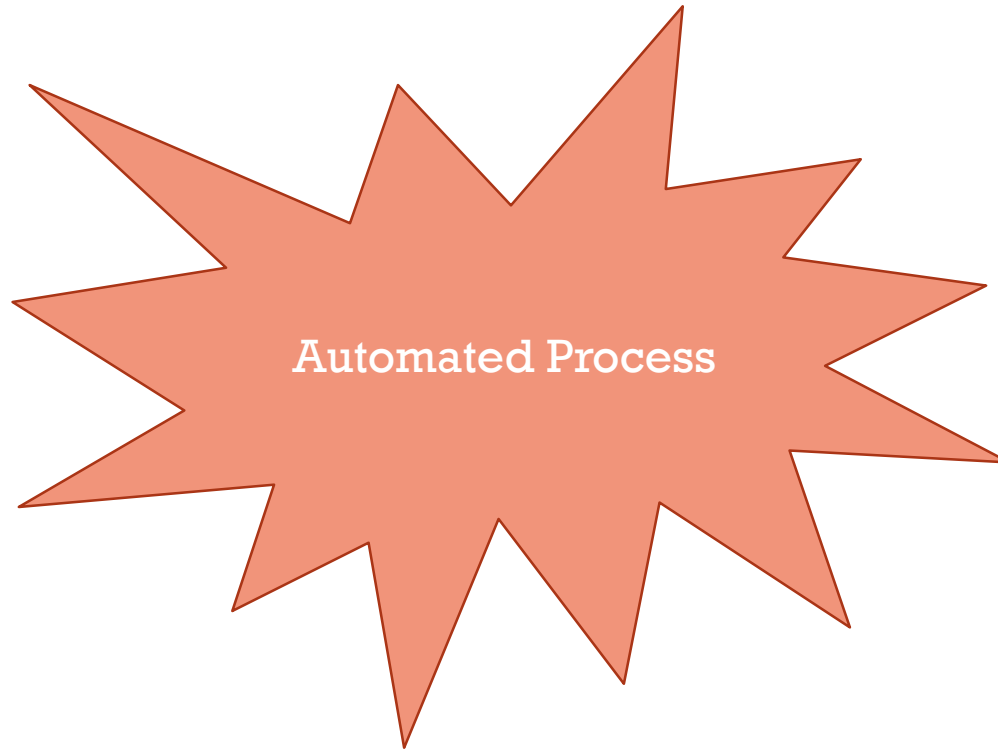
LINEAR INTERPOLATION

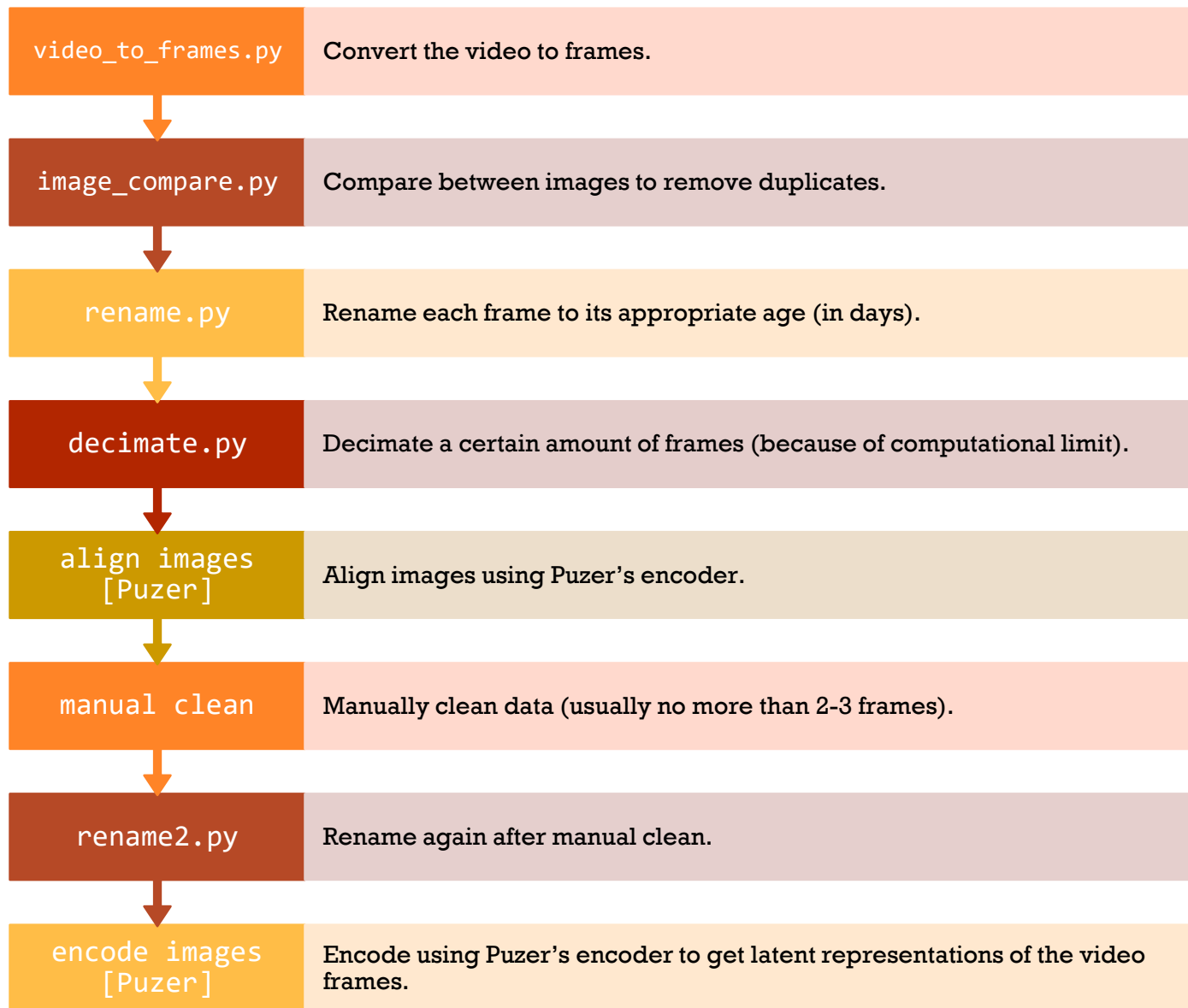




DATA PIPELINE

We wanted to automate the data processing pipeline:





DATA PIPELINE TOOLS



LINEAR INTERPOLATION

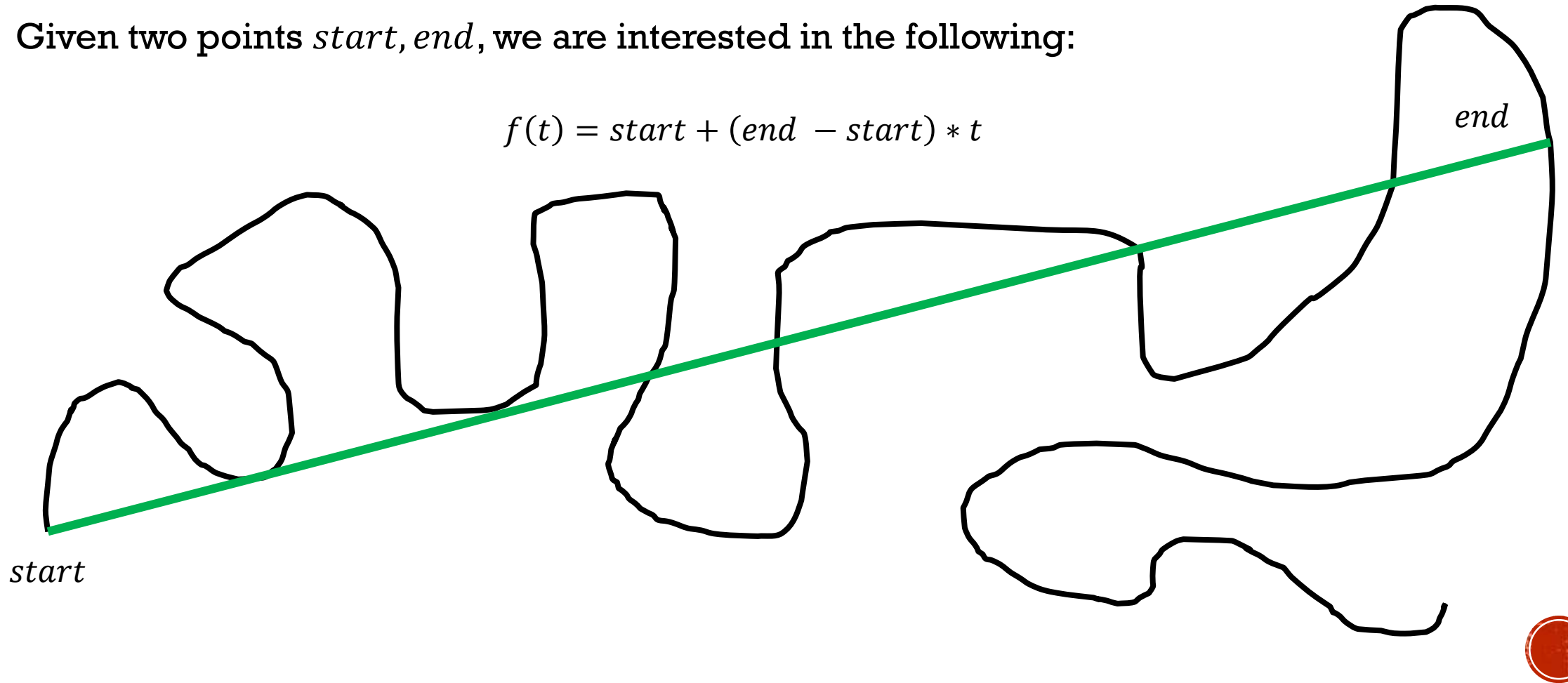
- Given a data set, we need a method to easily carry out experiments.
- Most experiments' independent variable is the interpolation method used.
- Since we are currently only testing linear interpolators, we built a function that finds the error of a linear interpolator, given the expected value.



LINEAR INTERPOLATION

Given two points *start*, *end*, we are interested in the following:

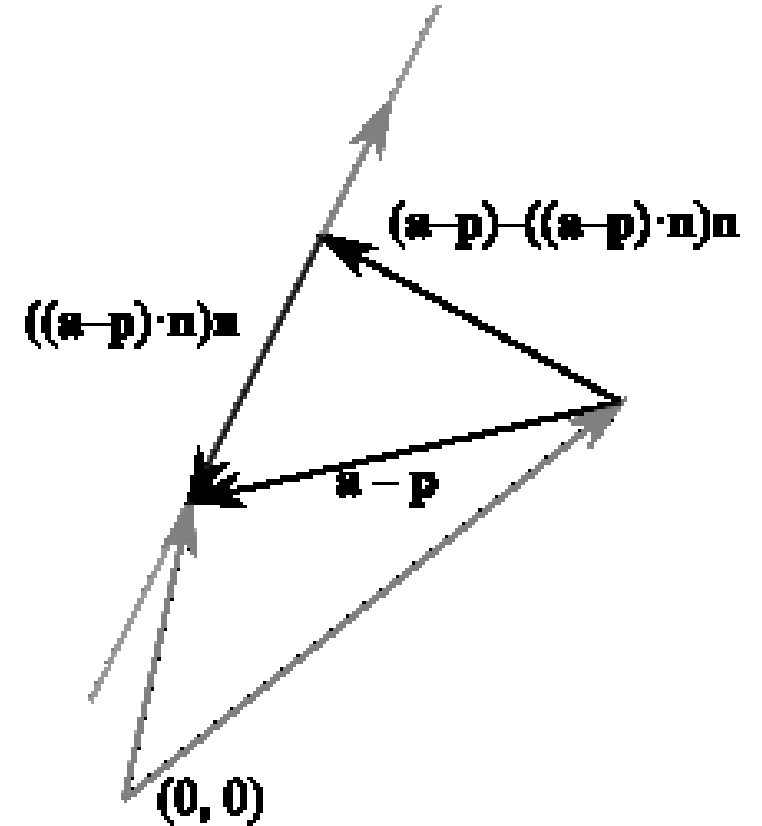
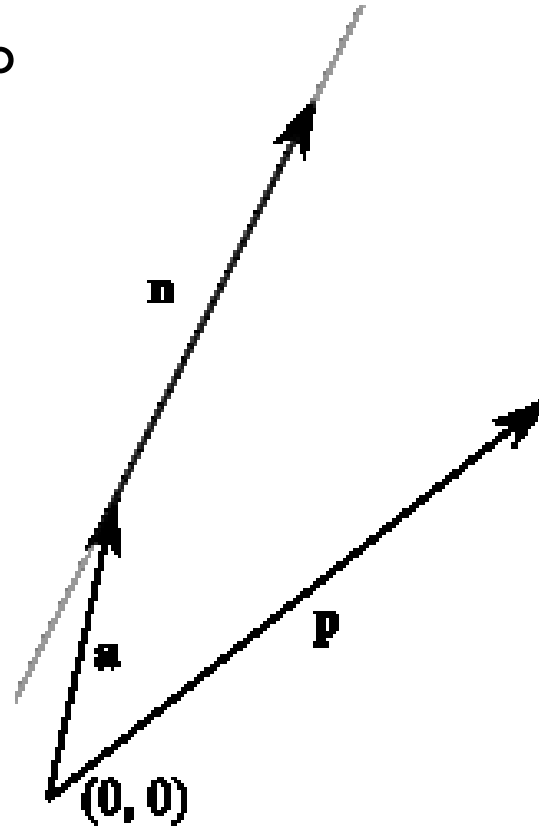
$$f(t) = start + (end - start) * t$$



LINEAR INTERPOLATION

We find how close our line gets to
the real value:

$$d = (a - p) - ((a - p) \cdot n)n$$



IMPLEMENTATION

```
def findError(real, interpolator):  
    line = interpolator(1) - interpolator(0)  
    normalizedLine = line * (1/np.linalg.norm(line))  
    distanceLine = interpolator(0) - real - normalizedLine *  
                    np.dot((interpolator(0) - real), normalizedLine)  
    return real + distanceLine, np.linalg.norm(distanceLine)
```

- Do the above for all points in the data set.
- Compare over different start and end points and different videos.



EXPERIMENT PIPELINE - INPUT



VIDEO



START AGE



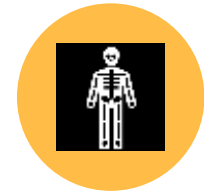
END AGE



AVERAGING RATIO



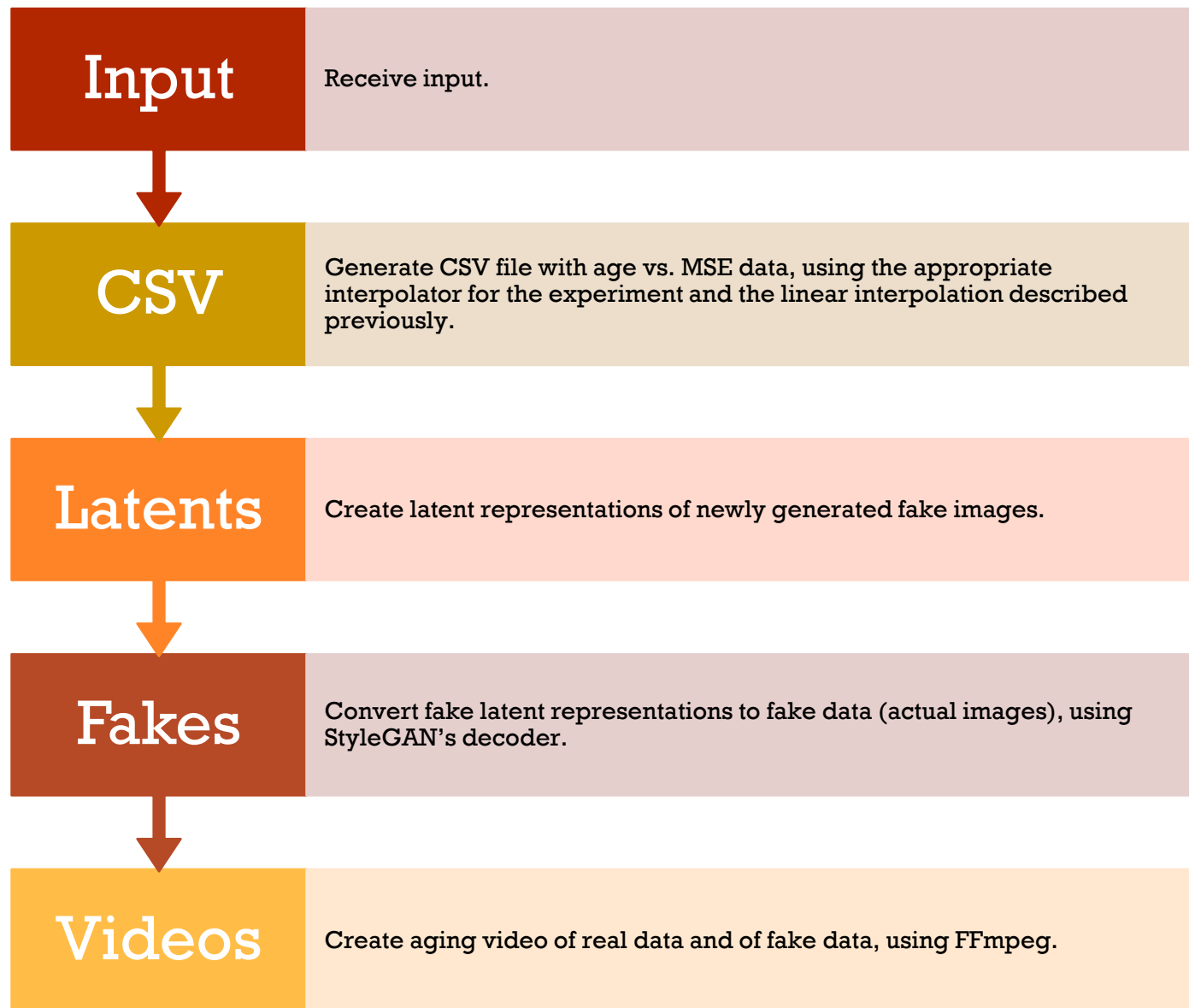
DIMENSIONS IN
LATENT SPACE TO BE
TESTED



INTERPOLATION
METHOD (STANDARD
/ PUZER)

With this information we are able to produce the plot of age vs. MSE and real vs. fake videos of aging.





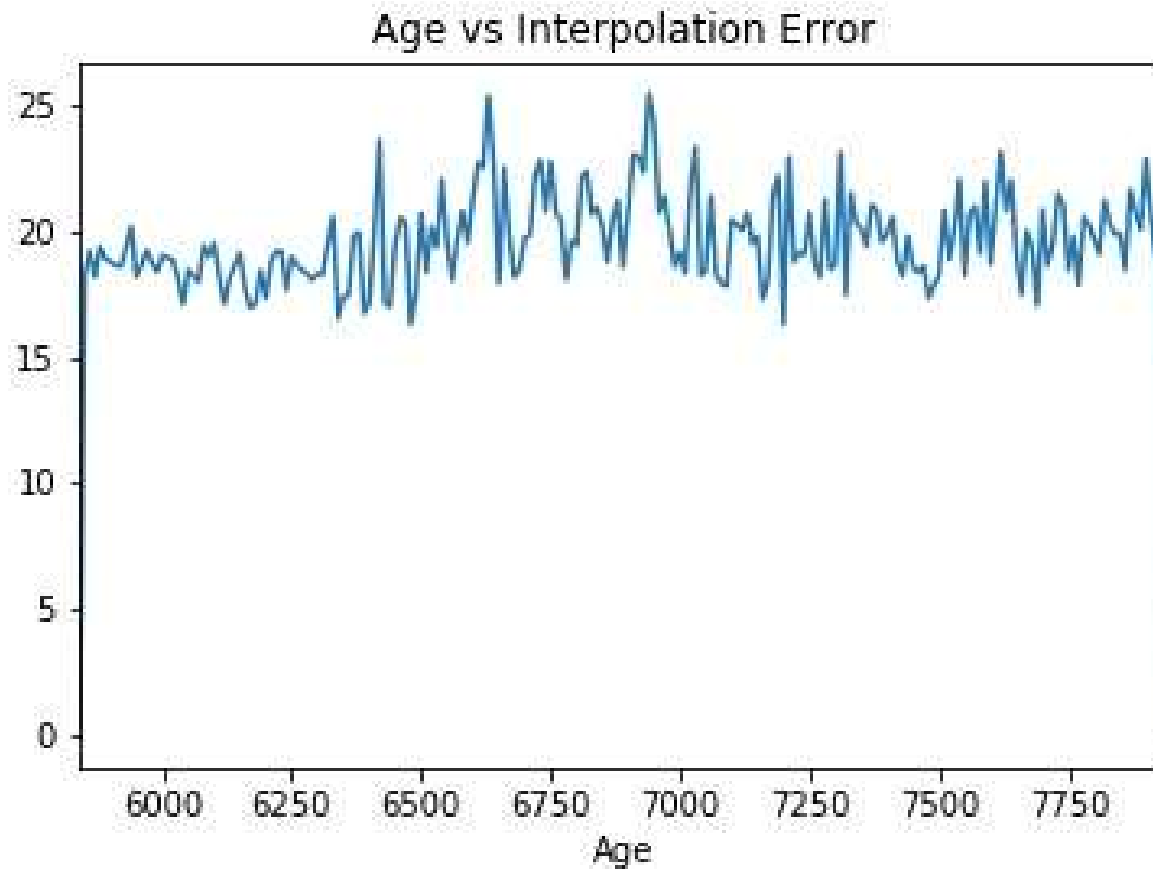
EXPERIMENT PIPELINE



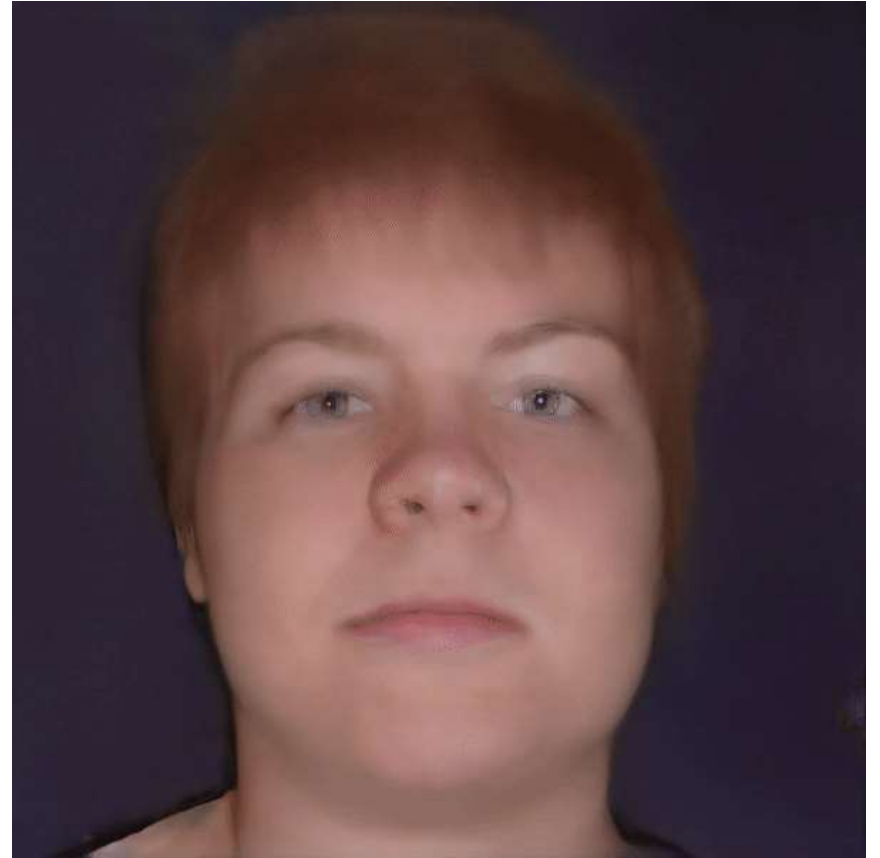
RESULTS - GIF



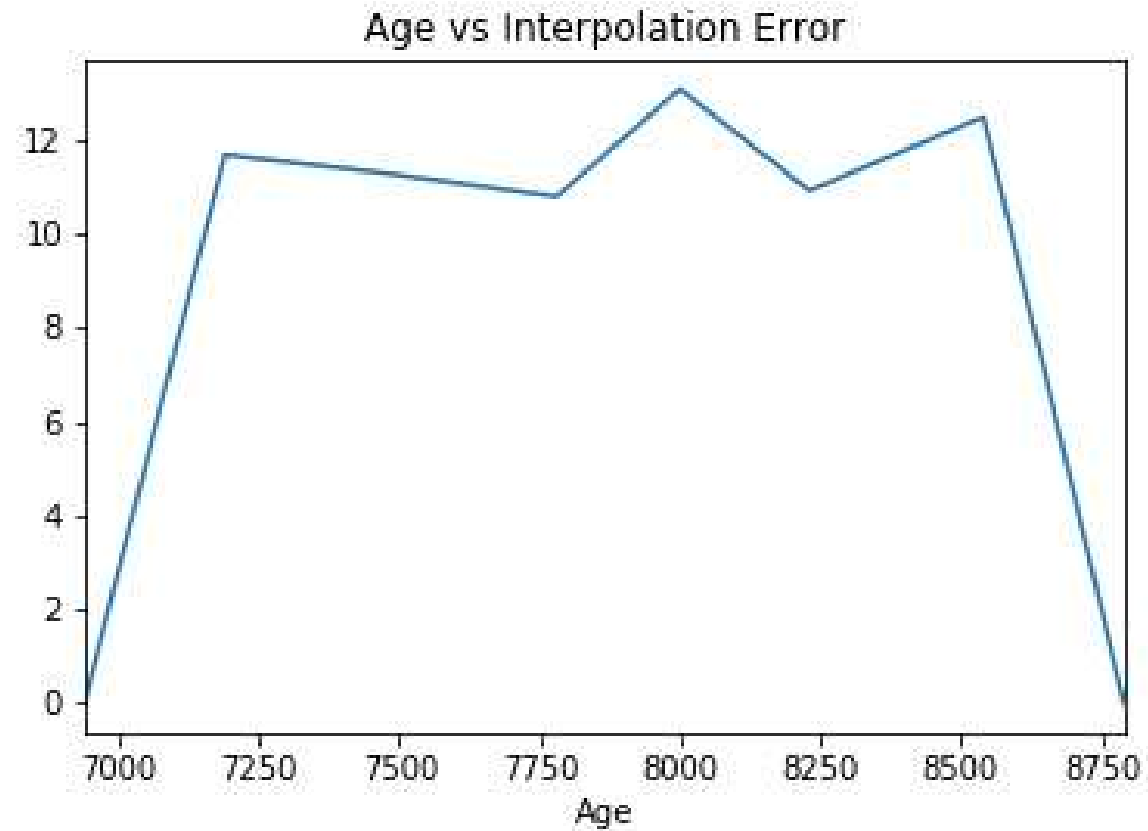
RESULTS - GRAPH



RESULTS - GIF



RESULTS - GRAPH

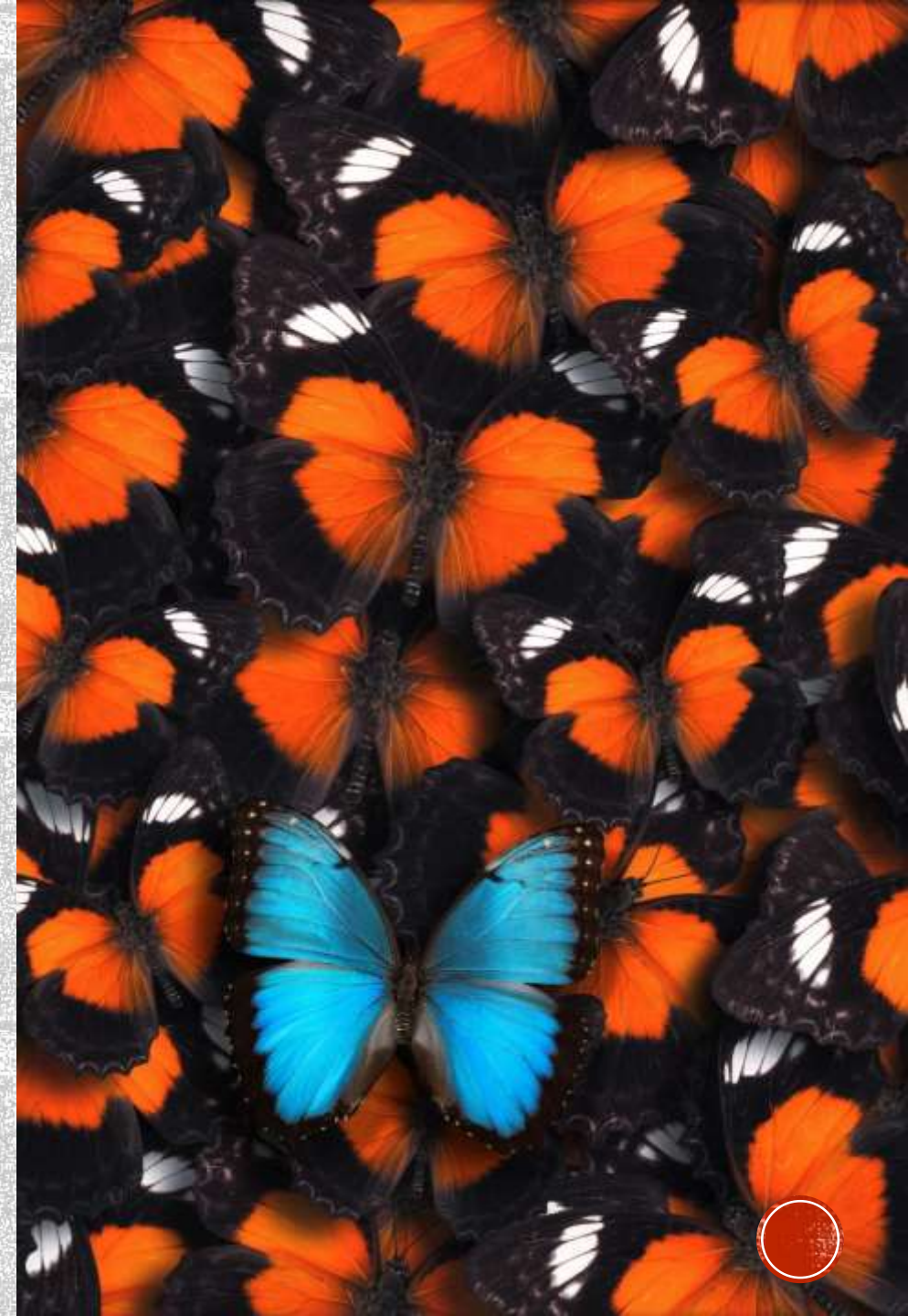


EXPERIMENT DEMO

STRANGE

This strange phenomenon persisted over:

- Different sizes of age gap for interpolation (different values for $|start - end|$).
- Different videos.
- Brute force search for closest match (instead of the analytic option shown earlier).
- Averaging batches of data.
- Using Puzer's latent direction for aging.

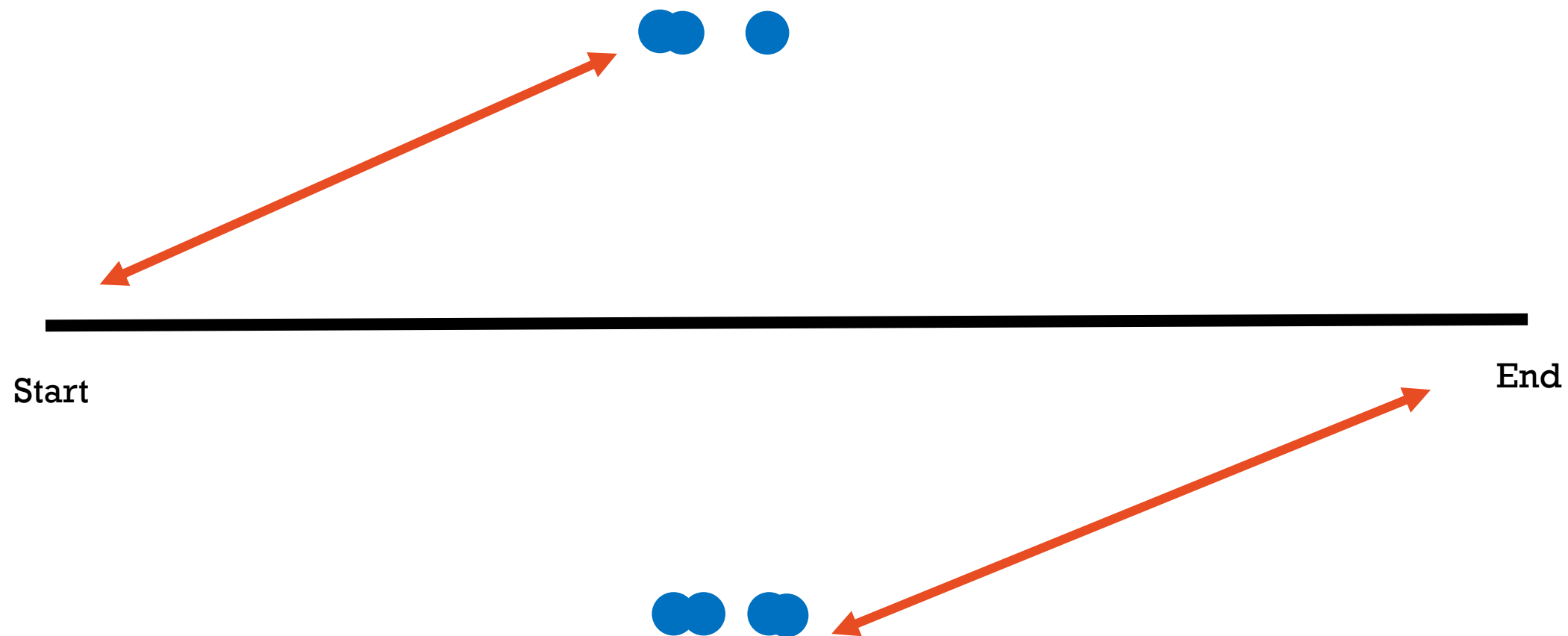






Start End







Account for more noise

Check error only on certain dimensions
(earlier dimensions are weightier in the
look of the image)

Average results over multiple
experiments/people



Use a more complex model

Linear regression

Neural Networks (maybe LSTM)

HOW DO WE CONTINUE FROM HERE



B. Truncation trick in \mathcal{W}

If we consider the distribution of training data, it is clear that areas of low density are poorly represented and thus likely to be difficult for the generator to learn. This is a significant open problem in all generative modeling techniques. However, it is known that drawing latent vectors from a truncated [42, 5] or otherwise shrunk [34] sampling space tends to improve average image quality, although some amount of variation is lost.

We can follow a similar strategy. To begin, we compute the center of mass of \mathcal{W} as $\bar{\mathbf{w}} = \mathbb{E}_{\mathbf{z} \sim P(\mathbf{z})}[f(\mathbf{z})]$. In case of FFHQ this point represents a sort of an average face (Figure 8, $\psi = 0$). We can then scale the deviation of a given \mathbf{w} from the center as $\mathbf{w}' = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}})$, where $\psi < 1$. While Brock et al. [5] observe that only a subset of networks is amenable to such truncation even when orthogonal regularization is used, truncation in \mathcal{W} space seems to work reliably even without changes to the loss function.

HOW DO WE CONTINUE FROM HERE





Until next time...