Controlling Memorability of Face Images

CNAI Exploratory Summer Report

Jason Cala

This report contains a summary of the exploratory work done in the summer of 2023 by Jason Cala on the topic of Memorability in Face Images to be paired with the presentation slides. Supplementary material and papers can be found in the links below.

Predicting Face Memorability: https://psyarxiv.com/2a7hy/
Controlling Memorability of Face Images: https://arxiv.org/abs/2202.11896
Mohammad Younesi's Files: https://openreview.net/forum?id=tm9-r3-O2lt
Jason Cala's Github Repo: https://github.com/jasoncala/face-memorability

This report is just a summary of everything, a further explanation on how to use the code is given in the Github README file. Code snippets are shown at the end of this file.

Research Questions:

- What controls a facial image's ability to be remembered?
- How can we manipulate to increase the memorability of a face image?
- Can we quantifiably prove which traits are most important?

Related Works:

This section includes assurances that give us confidence that our questions can be answered along with previous research work that is helpful to read for better understanding. These topics are important to understand prior.

Image Memorability:

- Memorability Is an intrinsic feature of an image.
- Consistent across different observers.
- Can be measured and manipulated.
- https://ieeexplore.ieee.org/document/6629991

General Adversarial Networks (GANs):

- GANs have been shown to be a good tool for creating synthetic images.
- Images can be modified through their latent vectors.
- https://arxiv.org/abs/1406.2661 (This paper not very necessary to read)

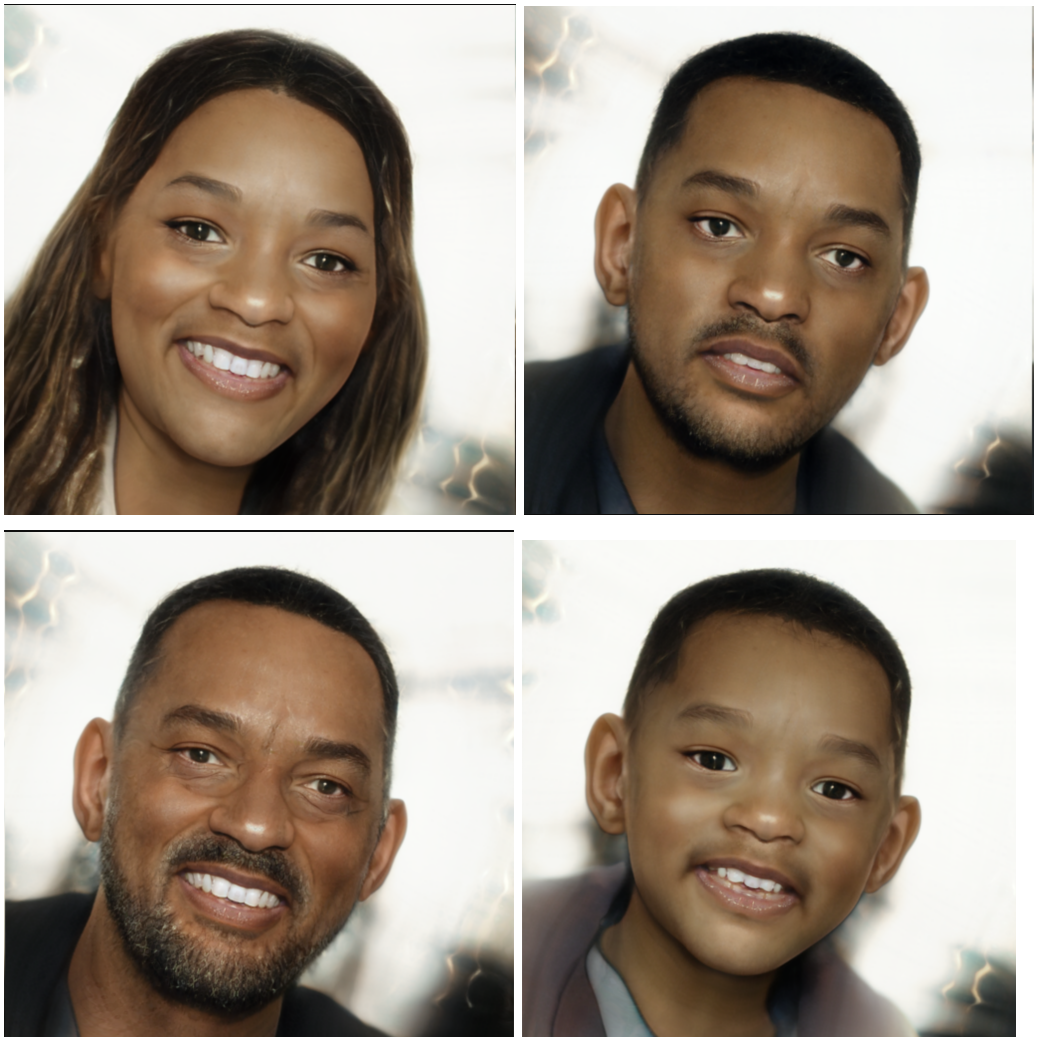
StyleGAN:

- StyleGAN was chosen for this use case.
- State-of-the-art model for generating and reconstructing real looking faces with high accuracy.
- Provides an extended latent space that can be leveraged to derive accurate hyperplanes.

- Less entangled than other GANs which can make it easier to specify traits.
- https://arxiv.org/abs/1812.04948
- https://arxiv.org/abs/1912.04958
- https://github.com/NVlabs/stylegan2

InterFaceGAN:

- Proposes approach for semantic face editing of images using latent space.
- An example with StyleGAN image is shown below.
- https://github.com/genforce/interfacegan



DISCLAIMER: The following steps are not needed to be repeated as the hyperplane is provided in materials. Better explanations are provided in the Controlling Memorability of Face Images paper.

Creating the Dataset:

For the purpose of analyzing the latent vectors of the GAN and their relation to memorability, a large dataset of face images with their memorability score is needed. The largest dataset available was the 10k US Adult Faces Database, which consists of 2,222 images that are annotated with memorability scores. This is not a large enough dataset so StyleGAN was leveraged to make it larger. The model was trained upon the FFHQ dataset which consists of 70000 high quality face images with variations of age, gender and glasses and randomly generated 100k images.

The next step to finish the dataset was assigning their memorability values. Each image had its memorability predicted using computational memorability prediction models. SENet50, an assessor, was pre-trained with VGGFace2, a large scale face recognition dataset, and then fine-tuned on the 10k Face Database to correctly estimate face memorability scores.

Finding and Modifying the Hyperplane:
Now with a good size dataset, the aim was to find a hyperplane to separate the highly-memorable and low-memorable images. These images were organized by the mean memorability of the group. Next, logistic regression was used to find a hyperplane between the low and high memorable images. The extended latent w-space of StyleGAN was used for this rather than the z-space for better accuracy. This step was important because memorability of each image could now be seen as the distance from the image's extended latent vector from this hyperplane. The equation is also shown below.

$$mem_i \propto d(\boldsymbol{w}^*, \boldsymbol{w}_i) = \boldsymbol{w}^{*^T}.\boldsymbol{w}_i$$

Now, the memorability of each image could be modified by changing the distance of its extended latent vector from the extended latent vector as shown in the equations.

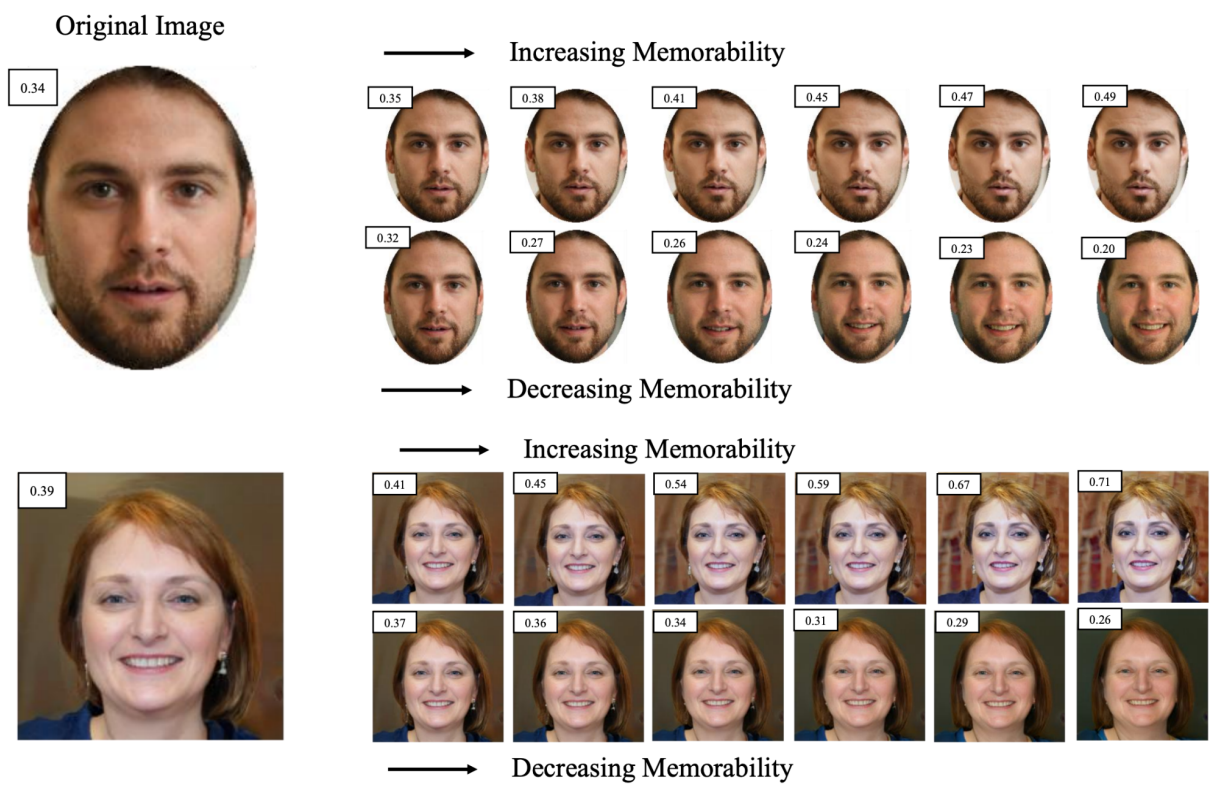
$$W_{edit} = W + \alpha W*$$

```
# new image vector = old image vector + magnitude * direction vector
x = np.ravel(lat1).copy() + 5 * w_sq
```

$$d(\boldsymbol{w}^*, \boldsymbol{w}_{edit}) = \boldsymbol{w}^{*^T}.\boldsymbol{w}_{edit} = \boldsymbol{w}^{*^T}.(\boldsymbol{w} + \alpha \boldsymbol{w}^*) = \boldsymbol{w}^{*^T}.\boldsymbol{w} + \alpha = d(\boldsymbol{w}^*, \boldsymbol{w}) + \alpha$$

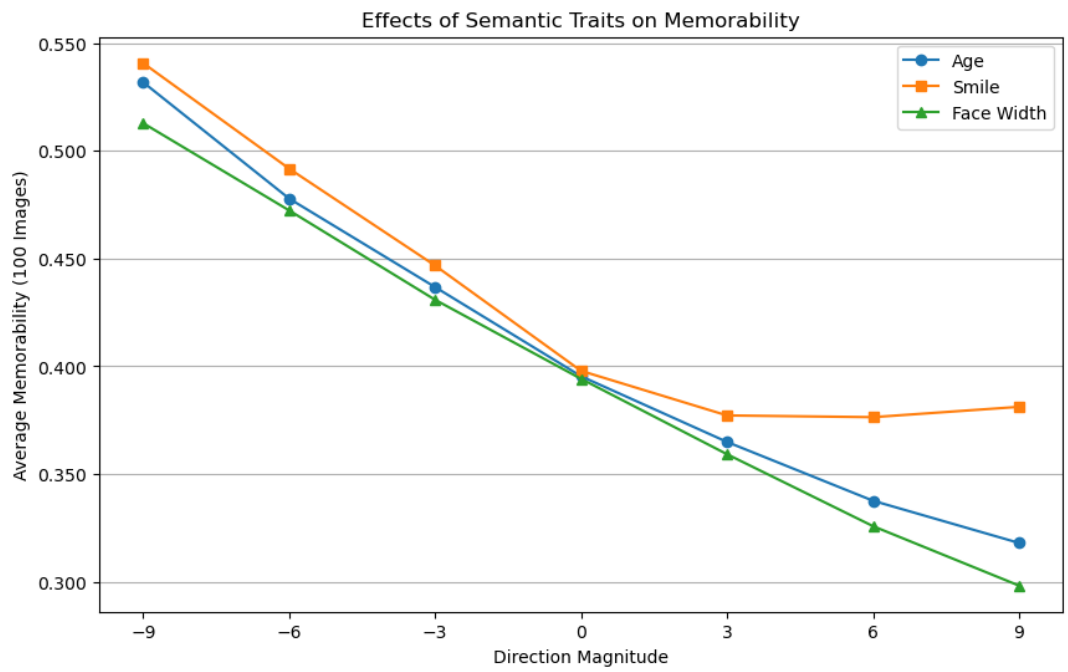
Modifying Face Memorability:

From the below experiments, we can see that certain traits are commonly being modified during memorability changes. These include age (you can see them getting younger), they become more serious, and a decrease in facial weight. Just these observations aren't enough so further examination had to be done.



Tests of Semantic Traits:

To ensure the previous trait observations were consistent across a larger amount of images, average memorability scores of different trait magnitudes were calculated to ensure the trends. The results showed that age, smile and face weight all on average decreases memorability.



Conditional Modifications:

Conditional manipulation of images were found to be successful if both direction vectors are available. As seen when modifying memorability, multiple traits are changed at once, so it would be good to disclude certain ones. This is done through subspace projection where essentially the vector directions make N^T N a diagonal matrix where semantics are independent of each other.

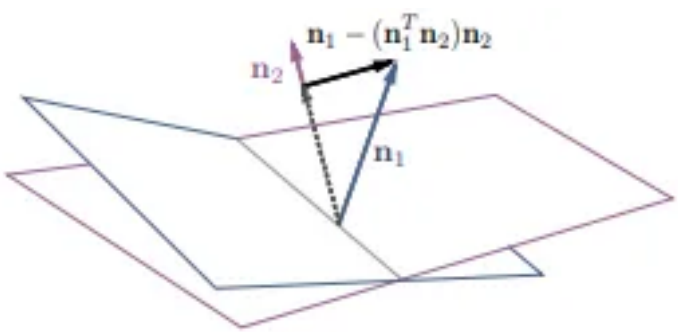
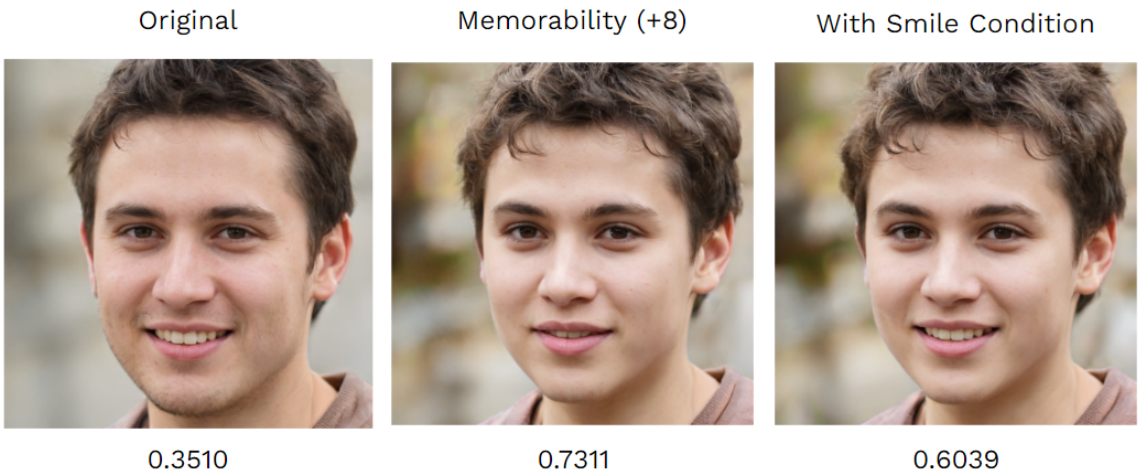


Fig. 2. Illustration of the **conditional manipulation via subspace projection**. The projection of $\mathbf{n}_1$ onto $\mathbf{n}_2$ is subtracted from $\mathbf{n}_1$, resulting in a new direction $\mathbf{n}_1 - (\mathbf{n}_1^T \mathbf{n}_2)\mathbf{n}_2$.



Investigating Vector Structure:
Image and direction vectors are all found to be in the structure of (1, 18, 512). This meant that each vector consisted of 18 subvectors made up of 512 values each. In all direction vectors, it was found that all 18 subvectors are equal to each other (contain the same 512 values) which causes it to be difficult to find the importance of a particular subvector. This led to further investigation of the effects of the 18.

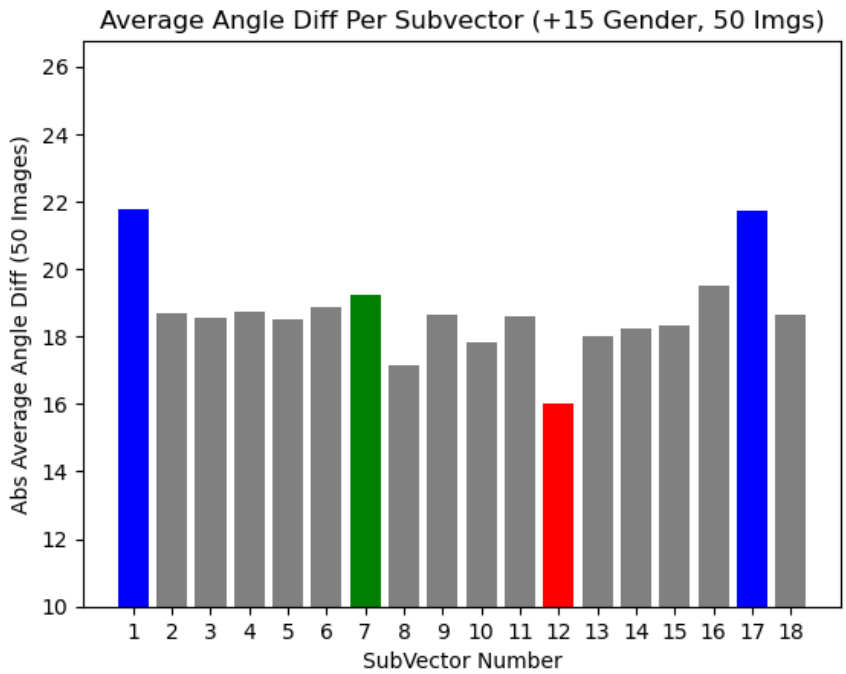
Experiments were conducted to find the main semantic traits controlled by each subvector with the below results.

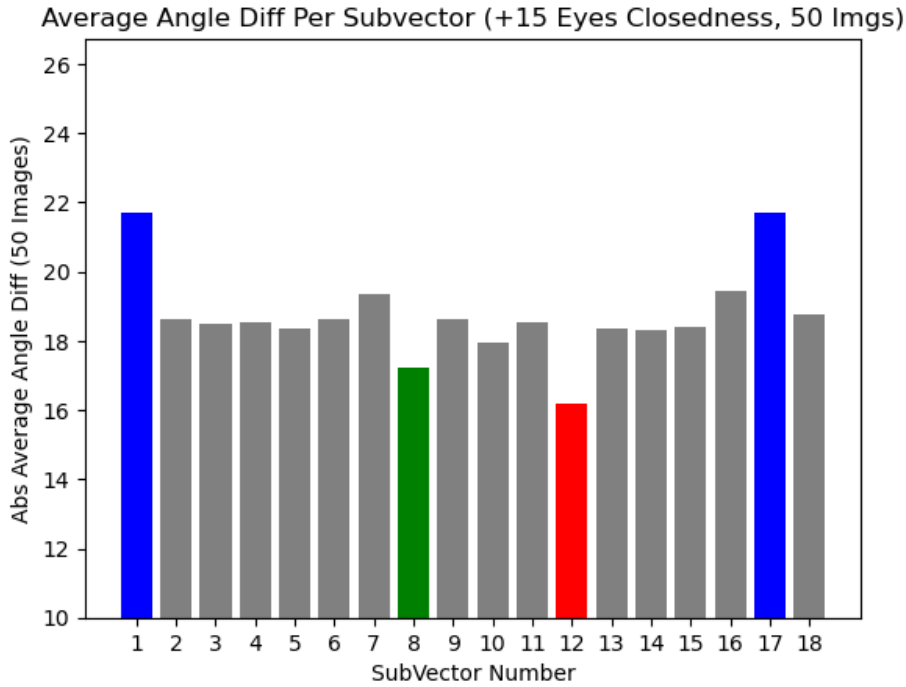
- 1/18 -> Face shape, orientation
- 2/18 -> Face orientation, body
- 3/18 -> Eye distance, hair
- 4/18 -> Hair length, smile
- 5/18 -> Smile, beard
- 6/18 -> Nose size, nose tilt
- 7/18 -> Gender, eye bags
- 8/18 -> Eye openness, eye colour
- 9/18 -> Beard, eyebrows
- 10/18 -> Eye colour
- 11-18/18 -> Colouring and texture

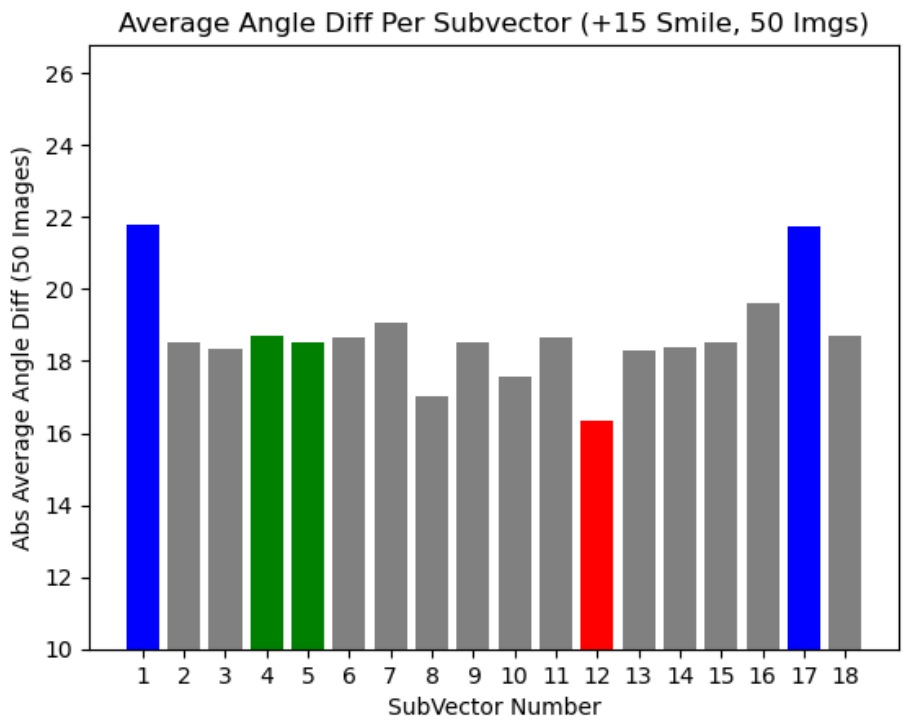
Below is an example of the experiments done to find these results.

Now knowing that each subvector represents unique traits, it would be very useful to know which subvector of the memorability direction vector has the greatest impact on the changes we see when we increase memorability. This would allow us to be more sure of which semantic traits are paired with memorability. However, a problem was ran into where changes to any singular subvector in the memorability vector did not cause any noticeable changes at all. Only when subvectors were grouped into groups of 4 were any changes noticeable. This could suggest that these traits or subvectors are not as disentangled as we thought.

The next experiment done to attempt to find the importance of subvectors was to calculate the angles between each subvector of the face image and each subvector of the direction vector. We could do this while changing the memorability of the image to see how this angle changes and whether there is a consistent pattern. However, the results showed that the same subvectors had the greatest/lowest changes no matter what trait was being modified. The below graphs show these results where blue represents the greatest change in angle, red represents the least change and green represents the subvector that affects the semantic trait the most.


Average Angle Diff Per Subvector (+15 Gender, 50 Imgs)
Abs Average Angle Diff (50 Images)
SubVector Number
10
12
14
16
18
20
22
24
26
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18


Average Angle Diff Per Subvector (+15 Eyes Closedness, 50 Imgs)
Abs Average Angle Diff (50 Images)
SubVector Number
10
12
14
16
18
20
22
24
26
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18



Conclusion and Next Steps

Three methods of finding subvector/trait importance were attempted and found unsuccessful:

- Can’t look directly at value changes (they’re all equal)
- Can’t look at subvectors one at a time (no image change)
- Can’t look at angle differences (no trait specific patterns)

Next steps would be to search for a new method or conclude that the subvectors are not of importance for finding the traits of most importance in memorability.

Code Snippets

Modifying Memorability

This below code snippet shows how an image vector (lat1) is modified by memorability by a factor of 5 (positive direction).

$$W_{edit} = W + \alpha W*$$

```
# new image vector = old image vector + magnitude * direction vector
x = np.ravel(lat1).copy() + 5 * w_sq
```

$$d(\boldsymbol{w}^*, \boldsymbol{w}_{edit}) = \boldsymbol{w}^{*^T}.\boldsymbol{w}_{edit} = \boldsymbol{w}^{*^T}.(\boldsymbol{w} + \alpha \boldsymbol{w}^*) = \boldsymbol{w}^{*^T}.\boldsymbol{w} + \alpha = d(\boldsymbol{w}^*, \boldsymbol{w}) + \alpha$$

Conditionally Modifying Memorability

This below code snippet shows how a conditional direction vector is created. This direction vector applies memorability while conditionally not affecting the smile trait/vector.

```
# n1 - (n1.t * n2) * n2
cond_smile = w_sq - np.ravel((np.dot(w_sq.reshape(-1,1).T,smile.reshape(-1,1))[0][0])*smile)
```

Example of Subvector Investigations

This below example shows how a chosen subvector of an image (7) was modified. In this particular example it was dropped (all set to 0) however, using the same method the 512 values of the subvector could be set to any other value or unique random values.

```
# Example dimension dropping
dimension_to_drop = 6

dropped_code = np.copy(lat1)
dropped_code[dimension_to_drop, :] = 0

# Generate image from dropped code
x = np.ravel(dropped_code).copy()
x = x.reshape((1, 18, 512))
out = generator.synthesis_network(x)
img = convert_images_to_uint8(out, nchw_to_nhwc=True, uint8_cast=True)
img = img.numpy()[0]
sample = load_image(img)
sample = sample.reshape((1,224,224,3))
x = assessor_2.predict(sample)[0]
print(calc_mem(x))

# Display the dropped image
plt.imshow(img)
# plt.title("Dimension Dropping")
plt.axis('off')
plt.show()
```

This next example below was the start of how the subvector angles were compared between a direction vector and an image for a single image (prior to modifications). This format was then later further applied to find the differences in angles caused by changing the memorability of the image (lat1). This further work can be found in the jupyter's notebook.

```python
# single image

direction_vector = w_sq.copy().reshape((18,512))
face_vector = np.ravel(lat1).copy().reshape((18,512))

# calculate angles between corresponding subvectors
angles = []
for i in range(18):
    subvector1 = direction_vector[i]
    subvector2 = face_vector[i]
    angle = calculate_angle(subvector1, subvector2)
    angles.append(angle)

# Print the angles
for i, angle in enumerate(angles):
    print(f"Angle between subvector {i+1}: {angle} degrees")
```

Overall, these are some basic examples of what started the work on this topic. Further code can be found in the Jupyter's Notebook in Github. In the future a new notebook will be created containing only the important code including descriptions of each step.