GANSpace: Discovering Interpretable GAN Controls

Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, Sylvain Paris

Presented by Charumathi Badrinath, Eric Shen, and Skyler Wu

Motivation + Example

- GANs are models trained to generate realistic-looking images well—and they do!
- They have **not** been designed, however, to make the image-generation process itself interpretable, or to support edits to generated images
- E.g.: This GAN generates faces well and can take in high-level styles or criteria (gender, age, race), but is not designed to facilitate post-hoc *continuous* and *meaningful* edits, such as head angle, hair length, lighting…

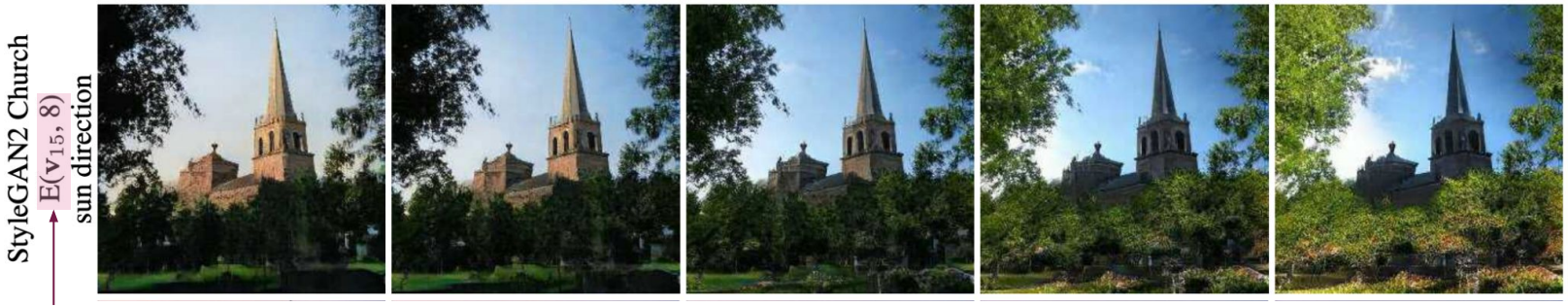
Main Contributions and Key Ideas

1. Applying a simple technique (**PCA**) in the **latent space** of GANs gives semantically significant directions.
2. Perturbing inputs in these directions in certain layers of the GANs → interpretable edits in properties of generated images.
3. Such edits are high-level and nuanced (e.g. object shape to background landscape)
4. A user need only label each “control direction” once to edit the image

Novel setting:
unsupervised identification of **interpretable** directions in **existing** GANs

In other words, the authors posit that exploring the “Ei**GAN**space” gives us new controls to edit and possibly better understand GANs at low cost

Main Contributions and Key Ideas



Perturbing the 15th principal component direction in the latent space, in the 8th layer.

Combining edits in various principal component directions to edit a generated photo.

GANs: Generative Adversarial Networks

Seminal Paper: Goodfellow et al. (2014)

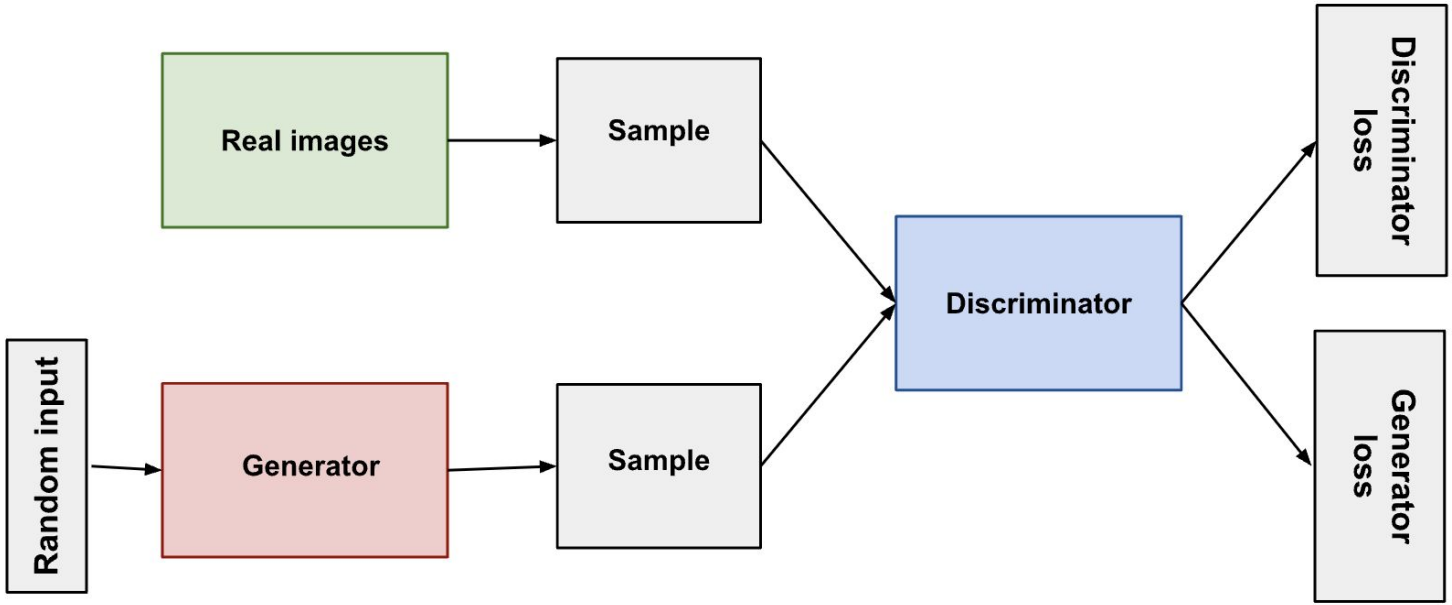
- Objective: **Generative Image Modeling**—create realistic-looking, artificial images that resemble the training data.
- GAN = Generator (G) + Discriminator (D) networks, both CNNs in image context.
    1. Generator: generate artificial images starting with noise input.

$$\underbrace{\mathbf{z} \sim p(\mathbf{z})}_{\text{Random Latent Vector}} \longrightarrow \underbrace{I = G(\mathbf{z})}_{\text{Artificial Image Generation (RGB)}} = \overbrace{\left(G_L \circ G_{L-1} \cdots \circ \underbrace{G_1}_{\text{Feature Tensor, i.e. a Set of Feature Maps}}\right)}^{\text{Iterative Feature Outputs}} (\mathbf{z})$$

    2. Discriminator: try to distinguish generator's fake images from real training data.

GANs: Generative Adversarial Networks

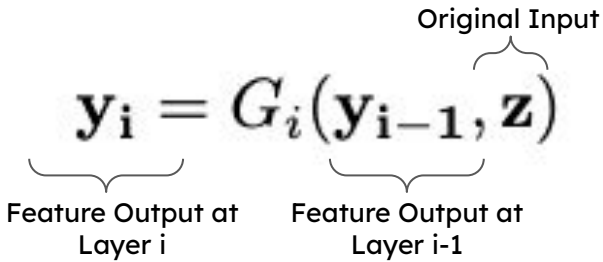
Key Idea: "Back-and-Forth Between Generator and Discriminator"



Source: https://developers.google.com/machine-learning/gan

BigGAN: Brock et al. (ICLR 2019)

1. Class-Conditional Image Generation
   - Given **z** from latent space and some class information (e.g., “dog”), generate the synthetic image.
   - BigGAN takes in class information as input → model *conditional distribution!*
2. Architecture designed for higher-resolution images and more parameters
3. “Skip-Z Inputs” in intermediate layers: **z** is imputed as a shared embedding into each layer



BigGAN: Brock et al. (ICLR 2019)

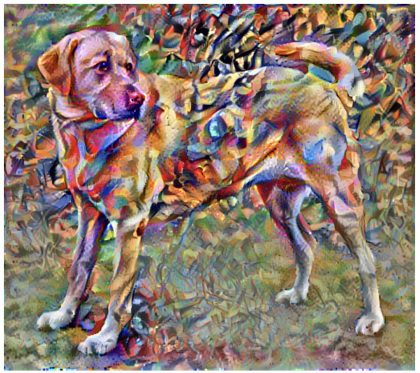
BigGAN → High Quality, High Resolution Images → Difficult to Interpret.

Class-Conditional Samples from BigGAN

StyleGAN: Karras et al. (CVPR 2019)

1. Inspiration: “**style transfer**” (content image + reference style reference image → blend into output image that keeps main aspects of content image, but appears to be “painted” in the “style” of the style reference image.
2. Motivation: want to better “control the image synthesis process”
    - Learned constant input + adjust “style” at each convolutional layer (e.g., pose, lighting, viewpoint)


+

=


StyleGAN: Karras et al. (CVPR 2019)

Relevant Mathematical Details:

1. No random input **z**, rather—a fixed, constant learned input $\mathbf{y_0}$.
2. Slightly different intermediate layers and connections:

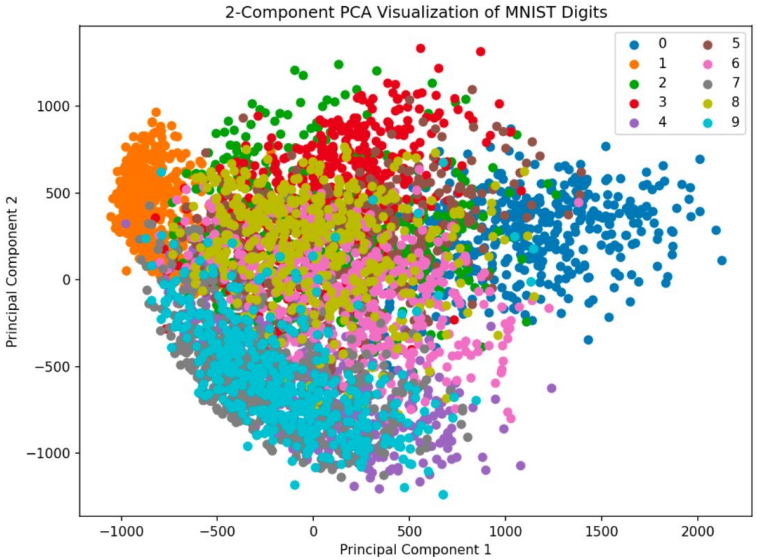
$$\mathbf{y_i} = G_i(\underbrace{\mathbf{y_{i-1}}, \mathbf{w}}), \text{ with } \mathbf{w} = \underbrace{M(\mathbf{z})}.$$

The **w** is a non-linear transformation of the original input **z.**

M is a 8-Layer Feed-Forward NN. Potentially different at each layer of Generator → “style mixing”

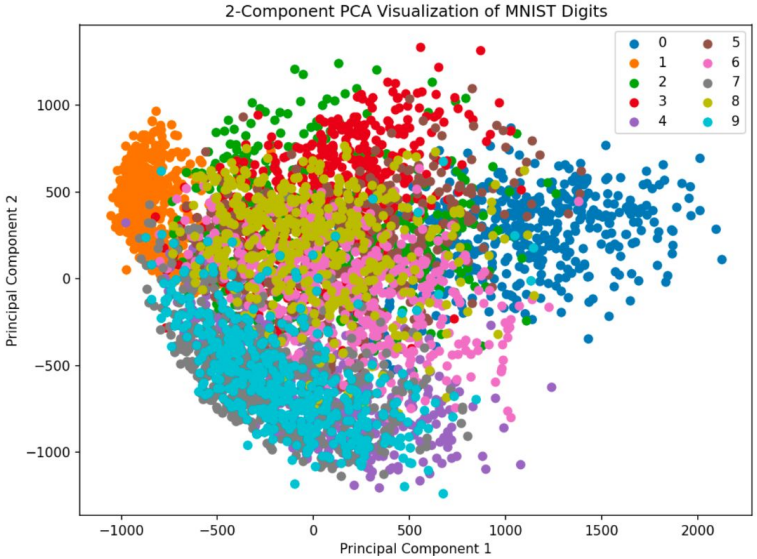
PCA: Principal Component Analysis

1. What is it?
   - An unsupervised learning technique for finding a lower-dimensional representation of a dataset
   - Find axes that explain the most variance in the data
   - **Principal Components** are these axes: they equal the eigenvectors corresponding to the largest eigenvalues of the covariance matrix



PCA: Principal Component Analysis

1. What is it?
   - An unsupervised learning technique for finding a lower-dimensional representation of a dataset
   - Find axes that explain the most variance in the data
   - **Principal Components** are these axes: they equal the eigenvectors corresponding to the largest eigenvalues of the covariance matrix
2. Why is it useful in our context?
   - Each PCA basis vector helps better separate our data, or, in this context, latent space representation vectors



Related Work and Their Limitations

1. Trying to Learn Latent Directions via **Supervised Learning** (Jahanian et al., Goetschalckx et al., etc.)
    - Helpful in the sense of giving users more control over generated images
    - However, this is very expensive—need manual supervision of learned controls. Requires labeling training images, very costly

Related Work and Their Limitations

1. Trying to Learn Latent Directions via **Supervised Learning** (Jahanian et al., Goetschalckx et al., etc.)
    - Helpful in the sense of giving users more control over generated images
    - However, this is very expensive—need manual supervision of learned controls. Requires labeling training images, very costly
2. Training GANs with **inherently disentangled representations** (Ramesh et al.)
    - “Entanglement” = one model attribute, but multiple possible effects, either simultaneously or selectively
    - If successful, would permit fine-tuned edits on generated images
    - But, extremely, extremely computationally expensive

Related Work and Their Limitations

1. Trying to Learn Latent Directions via **Supervised Learning** (Jahanian et al., Goetschalckx et al., etc.)
    - Helpful in the sense of giving users more control over generated images
    - However, this is very expensive—need manual supervision of learned controls. Requires labeling training images, very costly
2. Training GANs with **inherently disentangled representations** (Ramesh et al.)
    - “Entanglement” = one model attribute, but multiple possible effects, either simultaneously or selectively
    - If successful, would permit fine-tuned edits on generated images
    - But, extremely, extremely computationally expensive

So, why not just take an already-trained GAN and analyze its latent / feature space?

Finding Useful Directions in z-space

- Latent space controls the image → we want to find directions in latent space that correspond to interpolating over a certain feature in pixel-space
- For example, can we find a direction corresponding to hair color?

Idea: can use PCA to find most influential directions

Issue: What to run PCA on?

- **z**-space prior? too simple to be informative
- pixel space? too complex to be informative

Finding Useful Directions in StyleGAN

StyleGAN maps $\mathbf{z} \rightarrow \mathbf{w}$ which controls "styling" at each layer…

Idea: Finding principal axes of distribution p($\mathbf{w}$) could be useful!

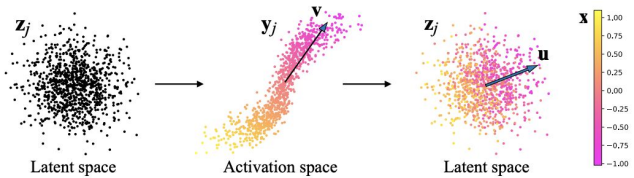
1. Sample *N* random vectors $\mathbf{z}_{1:N}$, compute $\mathbf{w}_i = M(\mathbf{z}_i)$
2. Apply PCA to $\mathbf{w}_{1:N}$ ⇒ yields new basis $\mathbf{V}$ for $\mathbf{w}$-space

Recall that there is a deterministic mapping between latent vector and image ⇒ given an image *defined by* $\mathbf{w}$ we can edit it by modifying w as follows:

$$\mathbf{w'} = \mathbf{w} + \mathbf{Vx}$$

controls magnitude of change in each direction of the basis

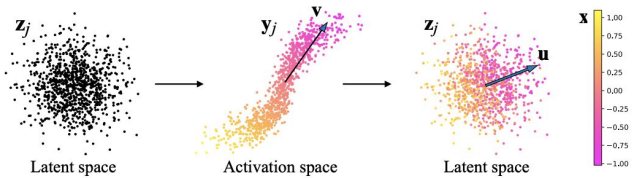
Finding Useful Directions in BigGAN



Issue: there is no styling parameter **w**; instead, **z** is passed as additional input

Idea: 2-step approach

Finding Useful Directions in BigGAN



Issue: there is no styling parameter **w**; instead, **z** is passed as additional input

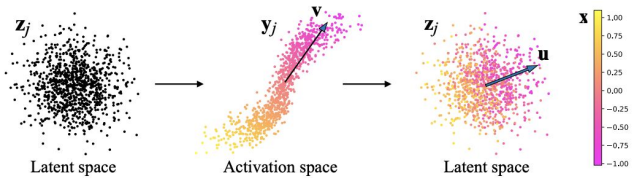
Idea: 2-step approach

done at the first linear layer

(Step 1) Perform PCA at intermediate network layer i

1. Sample *N* random vectors $\mathbf{z}_{1:N}$
2. Process through model to produce *N* feature tensors at ith layer $\mathbf{y}_{1:N}$
3. Apply PCA to $\mathbf{y}_{1:N}$ ⇒ yields new basis **V** for **y**-space, data mean **μ**
4. Project $\mathbf{y}_{1:N}$ into new basis yielding vectors $\mathbf{x}_{1:N} = \mathbf{V}^T(\mathbf{y}_{1:N} - \boldsymbol{\mu})$

Finding Useful Directions in BigGAN



Issue: there is no styling parameter **w**; instead, **z** is passed as additional input

Idea: 2-step approach

done at the first linear layer

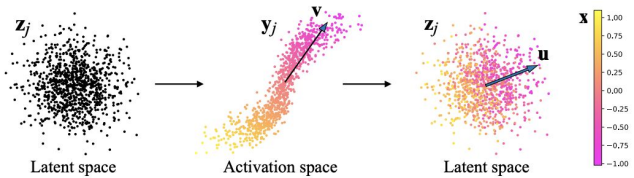
(Step 1) Perform PCA at intermediate network layer i

1. Sample *N* random vectors $\mathbf{z}_{1:N}$
2. Process through model to produce *N* feature tensors at ith layer $\mathbf{y}_{1:N}$
3. Apply PCA to $\mathbf{y}_{1:N}$ ⇒ yields new basis **V** for **y**-space, data mean **μ**
4. Project $\mathbf{y}_{1:N}$ into new basis yielding vectors $\mathbf{x}_{1:N} = \mathbf{V}^T(\mathbf{y}_{1:N} - \boldsymbol{\mu})$

(Step 2) Find latent directions corresponding to principal components

1. $\mathbf{U} = \text{argmin } \Sigma_j \|\mathbf{U}\mathbf{x}_j - \mathbf{z}_j\|^2$ → “principal direction” matrix

Finding Useful Directions in BigGAN



Issue: there is no styling parameter **w**; instead, **z** is passed as additional input

Idea: 2-step approach

done at the first linear layer

(Step 1) Perform PCA at intermediate network layer i

1. Sample *N* random vectors $\mathbf{z}_{1:N}$
2. Process through model to produce *N* feature tensors at ith layer $\mathbf{y}_{1:N}$
3. Apply PCA to $\mathbf{y}_{1:N}$ → yields new basis **V** for **y**-space, data mean **μ**
4. Project $\mathbf{y}_{1:N}$ into new basis yielding vectors $\mathbf{x}_{1:N} = \mathbf{V}^T(\mathbf{y}_{1:N} - \boldsymbol{\mu})$

(Step 2) Find latent directions corresponding to principal components

1. $\mathbf{U} = \text{argmin } \Sigma_j \|\mathbf{U}\mathbf{x}_j - \mathbf{z}_j\|^2$ → “principal direction” matrix

We can edit an image defined by **z** by modifying **z** as follows

$$\mathbf{z'} = \mathbf{z} + \mathbf{U}\mathbf{x}$$

Granularity of Edits

Idea: Edits can be applied to all layers or just a few for more localized effects

- StyleGAN: pass the modified **w**’ to some layers, pass the original **w** to others
- BigGAN: pass the modified **z**’ to some layers and the original **z** to others + keep the **z** at the beginning of the network the same

Idea: Edits can be *composed* by moving in several principal directions simultaneously

Edit directions can be labelled as corresponding to a concept by users

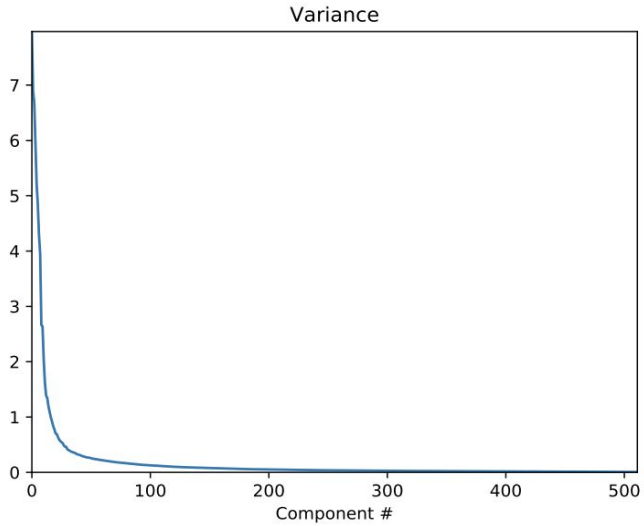
Findings and Results

(a) Fix first 8 PCA coord., randomize remaining 504 (Pose and camera fixed, appearance changes)

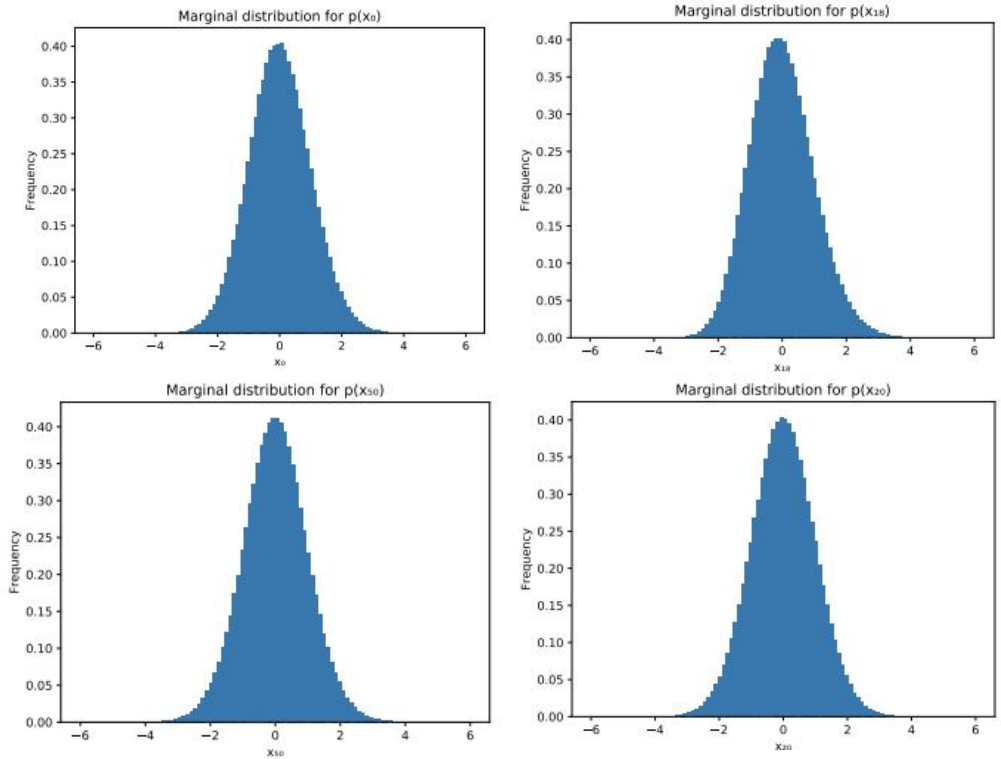
(b) Randomize first 8 PCA coord., fix remaining 504 (Appearance fixed, pose changes)

Changing the **first 20** principal components (PCs) in latent space correspond to changes in image **layout**, **configuration**, and **perspective**; later PCs dictate object appearance, background, and smaller details.

Findings and Results



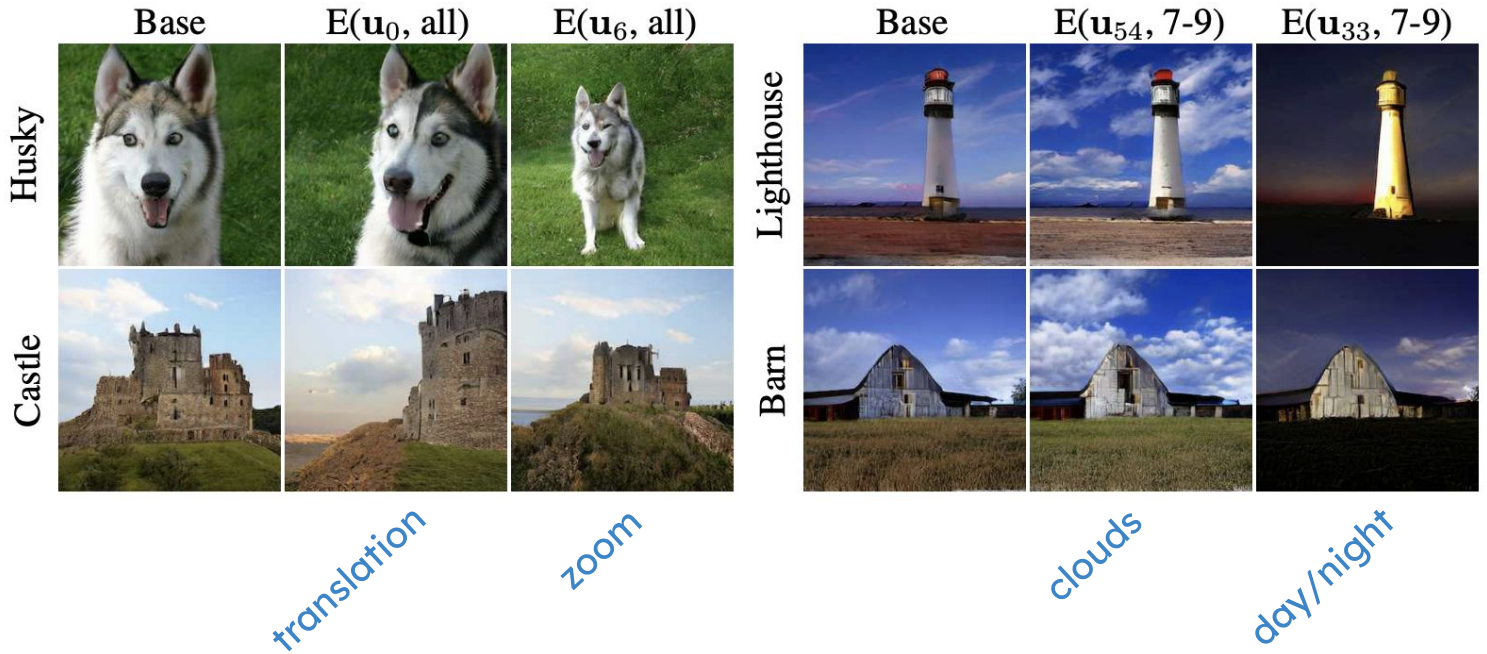
Variance in latent space is tied to magnitude of image change



Empirical PDFs of Principal Components are bell curves

StyleGAN's **first 100 PCs sufficiently explain overall image** appearance, and are distributed unimodally and almost independently

Findings and Results



BigGAN's PCs seem to be **class-independent**, i.e. PCA gives the same results for different images in different classes

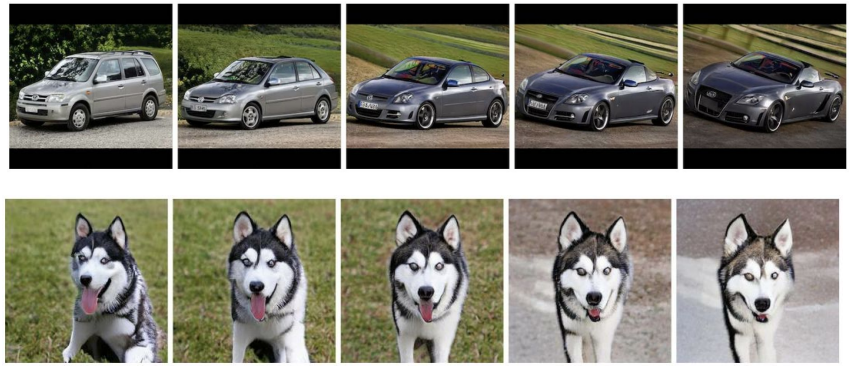
Analyzing GANs Through Principal Components

Limitations in edit expressivity through PCs seem to be inherited by dataset limitations:

- StyleGAN faces cannot be translated in the image
- PCs corresponding to wrinkles/makeup have no effect on children/men

Entanglement/Superposition of different attributes is displayed by some PCs:

- One PCs for StyleGAN cars corresponds to sportiness and more open-road backgrounds
- Rotating a dog causes its mouth to open



Comparison with Related Techniques

- Perturbing the latent space in **randomly-chosen directions** results in virtually indiscernible changes in mixtures of several attributes
- By contrast, PCA gives a more separated, ordered series of stylistic edit axes

- Results are similar to those achieved through supervised methods, while being simpler to compute (no extra training)
- PCs provide many more edit directions that would be costly to individually discover via supervised techniques
- More entanglement

(d) StyleGAN1 FFHQ Smile $E(\mathbf{v}_{44}, 3)$

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

- Approach seems to be limited to the specific GAN architecture

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

- Approach seems to be limited to the specific GAN architecture
- Principal components *a priori* are not linked with specific stylistic meaning: users would have to try changing values manually to see their effects

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

- Approach seems to be limited to the specific GAN architecture
- Principal components *a priori* are not linked with specific stylistic meaning: users would have to try changing values manually to see their effects
- Choosing which principal components to modify at what layers seems arbitrary

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

- Approach seems to be limited to the specific GAN architecture
- Principal components *a priori* are not linked with specific stylistic meaning: users would have to try changing values manually to see their effects
- Choosing which principal components to modify at what layers seems arbitrary
- Entanglement of principal component styles hinders practicality

Strengths/Weaknesses

A cursory list:

- Method is unsupervised and computationally easier to implement than supervised methods
- Analysis of principal component directions elucidates possible interpretations of GANs' latent space representations
- Gives users flexibility: after labelling directions, users can perform numerous edits on various styles of the image

- Approach seems to be limited to the specific GAN architecture
- Principal components *a priori* are not linked with specific stylistic meaning: users would have to try changing values manually to see their effects
- Choosing which principal components to modify at what layers seems arbitrary
- Entanglement of principal component styles hinders practicality
- Does this constitute actual interpretability for the GANs?

Questions for the Audience

- Are these principal components a useful tool for interpreting GANs?
- How does their utility for interpretability compare with other methods we have studied? Do you think this method truly satisfies interpretability?
- Finding: most conceptually-aligned PCs for StyleGAN in w-space, most useful PCs for BigGAN originated in the first linear layer (then, projected to z-space).
- ^Do you think these findings provide novel insights into the models?
- How could this tool help identify and reduce biases in a model?

Findings and Results

Experiment targets: **StyleGAN**, **BigGAN** (n.b. methods differed for the models)

Main Finding Ideas:

1. Changing the **first 20** principal components (PCs) in latent space correspond to changes in image **layout**, **configuration**, and **perspective**; later PCs dictate object appearance, background, and smaller details
2. StyleGAN's **first 100 PCs sufficiently explain overall image** appearance, and are distributed unimodally and almost independently
3. BigGAN's PCs seem to be **class-independent**, i.e. PCA gives the same results for different images in different classes