InfoGAN-CR and ModelCentrality

GANs: generative adversarial networks
CR: Contrastive Regularizer

Modar Sulaiman

November 26, 2020
Figure 1: Examples of Photorealistic GAN-Generated Faces [1].

The face generations were trained on celebrity examples, meaning that there are elements of existing celebrities in the generated faces, making them seem familiar, but not quite.
Figure 1: Examples of Photorealistic GAN-Generated Faces [1].
Figure 1: Examples of Photorealistic GAN-Generated Faces [1].

The face generations were trained on celebrity examples, meaning that there are elements of existing celebrities in the generated faces, making them seem familiar, but not quite.
1. Some applications of Different Architecture Of Generative Adversarial Networks (GANs)

2. Disentanglement
   - Designing of disentangled generative models
   - Disentangled Representation
   - Disentangled Generative Models

3. GANs & infoGAN

4. Contrastive Regularizer

5. ModelCentrality
Use GANs to create art (Monet style) - CycleGAN
CycleGAN is a very popular GAN architecture primarily being used to learn transformation between images of different styles.
CycleGAN is a very popular GAN architecture primarily being used to learn transformation between images of different styles.

Unpaired Image-to-Image Translation.
CycleGAN is a very popular GAN architecture primarily being used to learn transformation between images of different styles.

Unpaired Image-to-Image Translation.

As an example, this kind of formulation can learn:

- a map between artistic and realistic images,
- a transformation between images of horse and zebra,
- a transformation between winter image and summer image and so on.
CycleGAN

CycleGAN is a very popular GAN architecture primarily being used to learn transformation between images of different styles.

Unpaired Image-to-Image Translation.

As an example, this kind of formulation can learn:

- a map between artistic and realistic images,
- a transformation between images of horse and zebra,
- a transformation between winter image and summer image and so on.

FaceApp is one of the most popular examples of CycleGAN where human faces are transformed into different age groups.
Increasingly realistic synthetic faces generated by variations on Generative Adversarial Networks (GANs)

Figure 2: Example of the Progression in the Capabilities of GANs from 2014 to 2017. In order, the images are from papers by Goodfellow et al. (2014), Radford et al. (2015), Liu and Tuzel (2016), and Karras et al. (2017). (see [2]).
Two main obstacles arise in the design of disentangled generative models:

- Designing architectures that achieve good disentanglement and good sample quality.
- Hyperparameter tuning and model selection given a fixed learning architecture.
A disentangled representation can be defined as one where single latent units are sensitive to changes in single generative factors, while being relatively invariant to changes in other factors.
A disentangled representation can be defined as one where single latent units are sensitive to changes in single generative factors, while being relatively invariant to changes in other factors.

A vector representation is called a disentangled representation with respect to a particular decomposition of a symmetry group into subgroups, if it decomposes into independent subspaces, where each subspace is affected by the action of a single subgroup, and the actions of all other subgroups leave the subspace unaffected.
A disentangled generative model takes a number of latent factors as inputs, with each factor controlling an interpretable aspect of the generated data.
A disentangled generative model takes a number of latent factors as inputs, with each factor controlling an interpretable aspect of the generated data.

We say a generative model has a better disentanglement if changing one latent code (while fixing other latent codes) makes a noticeable and distinct change in the generated sample (referred to as “informativeness” and “disentanglement”).
GAN
A generative adversarial network (GAN) has two parts:
A generative adversarial network (GAN) has two parts:

- The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.

- The discriminator learns to distinguish the generator’s fake data from real data. The discriminator penalizes the generator for producing implausible results.
Figure 3: GAN Generative Adversarial Networks Architecture. Both the generator $G$ and the discriminator $D$ are neural networks. The generator output is connected directly to the discriminator input. Through backpropagation, the discriminator’s classification provides a signal that the generator uses to update its weights.
A deep neural network generative model maps a latent code $z \in \mathbb{R}^d$ to a desired distribution of the samples $x = G(z)$. 
A deep neural network generative model maps a latent code $z \in \mathbb{R}^d$ to a desired distribution of the samples $x = G(z)$.

$z$ is typically drawn from a Gaussian distribution with identity covariance or a uniform distribution.
A deep neural network generative model maps a latent code $z \in \mathbb{R}^d$ to a desired distribution of the samples $x = G(z)$.

$z$ is typically drawn from a Gaussian distribution with identity covariance or a uniform distribution.

The discriminator provides an approximate measure of how different the current generator distribution is from the distribution of the real data.
GANs update weights of a generator $G$ and discriminator $D$ using gradient updates on the following adversarial loss:

$$
\min_G \max_D \mathcal{L}_{ADF}(D, G).
$$

We train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$. We simultaneously train $G$ to minimize $\log(1 - D(x))$ (see [3]).

$\mathcal{L}_{Adv}(D, G)$ provides an approximation of the Jensen-Shannon divergence between the real data distribution $P_{data}$ and the current generator distribution $P_{G}$. 
GANs update weights of a generator $G$ and discriminator $D$ using gradient updates on the following adversarial loss:

$$\min_G \max_D \mathcal{L}_{ADF}(D, G).$$

$$\mathcal{L}_{Adv}(D, G) = \mathbb{E}_{x \sim P_{\text{real}}} [\log(D(x))] + \mathbb{E}_{P_G} [\log(1 - D(x))] \quad (1)$$
GANs update weights of a generator $G$ and discriminator $D$ using gradient updates on the following adversarial loss:

$$
\min_G \max_D \mathcal{L}_{ADF}(D, G).
$$

$$
\mathcal{L}_{\text{Adv}}(D, G) = \mathbb{E}_{x \sim P_{\text{real}}} \left[ \log(D(x)) \right] + \mathbb{E}_{P_G} \left[ \log(1 - D(x)) \right] \quad (1)
$$

We train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$. We simultaneously train $G$ to minimize $\log(1 - D(x))$ (see [3]).
GANs update weights of a generator $G$ and discriminator $D$ using gradient updates on the following adversarial loss:

$$
\min_G \max_D \mathcal{L}_{ADF}(D, G).
$$

$$
\mathcal{L}_{Adv}(D, G) = \mathbb{E}_{x \sim P_{real}} [\log(D(x))] + \mathbb{E}_{P_G} [\log(1 - D(x))] \quad (1)
$$

We train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$. We simultaneously train $G$ to minimize $\log(1 - D(x))$ (see [3]).

$\mathcal{L}_{Adv}(D, G)$ provides an approximation of the Jensen-Shannon divergence between the real data distribution $P_{data}$ and the current generator distribution $P_G$. 
InfoGAN
**InfoGAN** is based on maximising the mutual information between a subset of latent variables and observations within the generative adversarial network (GAN) framework.

The idea is to provide a latent code, which has meaningful and consistent effects on the output.

In **InfoGAN** we split the generator input into two parts: the traditional noise vector and a new “latent code” vector.

The codes are then made meaningful by maximizing the Mutual Information between the code and the generator output.
Figure 4: InfoGAN architecture. New components outlined in red.
Figure 4: InfoGAN architecture. New components outlined in red.

\[ Q(c|x) \] “auxiliary” distribution, which is modeled by a parameterized neural network, and is meant to approximate the real \[ P(c|x) \] (the probability distribution for \( c \) given the image \( x \)).
Figure 4: InfoGAN architecture. New components outlined in red.

$Q(c|x)$ “auxiliary” distribution, which is modeled by a parameterized neural network, and is meant to approximate the real $P(c|x)$ (the probability distribution for $c$ given the image $x$).

$P(c|x)$ represents the likelihood of code $c$ given the generated input $x$. 
InfoGAN has regularizer based on mutual information.

InfoGAN split the latent codes into two parts: the disentangled code vector \( c \in \mathbb{R}^k \) and the remaining code vector \( z \in \mathbb{R}^d \) that provides additional randomness.

InfoGAN then uses the GAN loss with regularization to encourage informative latent codes \( c \):

\[
\min_G \max_D \mathcal{L}_{Adv}(D,G) - \lambda \ I(c; G(c, z))
\]

where \( I(c; G(c, z)) \) denotes the mutual information between the latent code \( c \) and the sample \( G(c, z) \) generated from that latent code, and \( \lambda \) is a positive scalar coefficient (see [4]).
CR - Contrastive Regularizer
InfoGAN-CR a novel architecture for training disentangled GANs.

The contrastive regularizer (CR) is inspired by a natural notion of disentanglement: latent traversal. The disentanglement should be measured via changes in the images when traversing the latent space.

This suggests a natural disentanglement approach: run latent traversal experiments and encourage models that make distinct changes.
Figure 5: Each row shows how the image changes when traversing a single latent code under the proposed InfoGAN-CR architecture (dSprites dataset). Latent codes capture desired properties: shape, rotation, scale, x-pos, ypos, of the image.
We generate two (or more) images from the generator, while fixing one of the latent codes \( c_i \) to be the same for both images.

We draw the rest of the latent codes uniformly at random, and let \((x, \hat{x}) \sim Q^{(i)}\) denote the resulting distribution of paired samples when factor \( c_i \) is fixed.

We propose measuring the distinctness of this latent traversal with Jensen-Shannon divergence among \( Q^{(i)} \)'s defined as

\[
d_{JS}(Q^{(1)}, \ldots, Q^{(k)}) \triangleq \frac{1}{k} \sum_{i \in [k]} d_{KL}(Q^{(i)} \parallel \bar{Q})
\]

where \( \bar{Q} = \frac{1}{k} \sum_{j \in [k]} Q^{(i)} \)

This measures how different each latent code traversal is.
We introduce an additional discriminator $H : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^k$ that performs multi-way hypothesis testing.

Building upon InfoGAN’s architecture, we add contrastive regularization and refer to the resulting architecture as InfoGAN-CR.

For non-negative scalars $\lambda$ and $\alpha$, this architecture is trained as

$$\min_{G,H,Q} \max_D \mathcal{L}_{ADV}(G,D) - \lambda \mathcal{L}_{Info}(G,Q) - \alpha \mathcal{L}_c(G,H)$$
We use the standard cross entropy loss:

$$\mathcal{L}_c(G,H) = \mathbb{E}_{I \sim U([k]), (x, \hat{x}) \sim Q(I)} [\langle I, \log H(x, \hat{x}) \rangle]$$

where $Q(I)$ denotes the joint distribution of the paired images,

$I$ denotes the one-hot encoding of the random index,

and $H$ is a k-dimensional vector-valued neural network normalized to be $\langle 1, H(x, \hat{x}) \rangle = 1$ for all $x$ and $\hat{x}$.

This naturally encourages each latent code to make distinct and noticeable changes, hence promoting disentanglement.
Figure 6: Like InfoGAN, InfoGAN-CR includes a GAN discriminator $D$ and an encoder $Q$, which share all convolutional layers and have separate fully-connected final layers. In addition, the $CR$ discriminator $H$ takes as input a pair of images $x$ and $\hat{x}$ that are generated by sharing one fixed latent factor $c_i = \hat{c}_i$ for a randomly chosen $i \in [k]$, and randomly drawing the rest. The discriminator is trained to correctly identify $i$, the index of the fixed factor.
The pair of coupled images $x$ and $\hat{x}$ are generated according to a choice of a coupling that defines how to traverse the latent space.

The discriminator $H$ tries to identify which code $i$ was shared between the paired images.

Both the generator and the discriminator try to make the k-way hypothesis testing successful.
We draw a random index $I$ over $k$ indices, and sample the chosen latent code $c_I \in \mathbb{R}$.

Two images are generated with the same value of $c_I$; the remaining factors are chosen independently at random.

Letting $c^m_j$ denote the $j$th latent code for image $m \in [1, 2]$, the contrastive gap is defined as $\min_{j \in [k] \setminus I} |c^1_j - c^2_j|$. The larger the contrastive gap, the more distinct the pair of samples.

It is mentioned in the paper that reducing the contrastive gap during training significantly improves FactorVAE scores.
Evaluation
We use the following metrics to evaluate various aspects of the trained latent representation: disentanglement, independence, and generated image quality.

We use the popular disentanglement metric FactorVAE proposed in [5]. This metric is defined for datasets with known ground truth factors.

We additionally compute the (less common) disentanglement metric of (DCI) (see [6]). The disentanglement metric (DCI) is computed using the random forest regressor, as implemented in the scikit-learn library.

The other metrics we compute are SAP [7], Explicitness [8], Modularity [8], MIG [9], and BetaVAE [10].
3DTeapots DATASET

<table>
<thead>
<tr>
<th>Model</th>
<th>FactorVAE</th>
<th>DCI</th>
<th>SAP</th>
<th>Explicitness</th>
<th>Modularity</th>
<th>MIG</th>
<th>BetaVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactorVAE</td>
<td>0.79 ± .03</td>
<td>0.55 ± .04</td>
<td>0.49 ± .05</td>
<td>0.84 ± .01</td>
<td>0.72 ± .02</td>
<td>0.24 ± .03</td>
<td>0.94 ± .02</td>
</tr>
<tr>
<td>InfoGAN (modified)</td>
<td>0.76 ± .06</td>
<td>0.62 ± .06</td>
<td><strong>0.57 ± .06</strong></td>
<td>0.82 ± .04</td>
<td><strong>0.98 ± .01</strong></td>
<td>0.34 ± .04</td>
<td>0.90 ± .07</td>
</tr>
<tr>
<td>InfoGAN-CR</td>
<td><strong>0.82 ± .02</strong></td>
<td><strong>0.66 ± .01</strong></td>
<td>0.53 ± .02</td>
<td>0.81 ± .01</td>
<td>0.97 ± .00</td>
<td><strong>0.38 ± .02</strong></td>
<td>0.89 ± .02</td>
</tr>
</tbody>
</table>

**Figure 7:** Comparisons of the popular disentanglement metrics on the 3DTeapots. We show in the next Table that with the proposed model selection scheme, we achieve the best performance on all metrics.
3DTeapots DATASET

Figure 7: Comparisons of the popular disentanglement metrics on the 3DTeapots. We show in the next Table that with the proposed model selection scheme, we achieve the best performance on all metrics.

![Table](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>FactorVAE</th>
<th>DCI</th>
<th>SAP</th>
<th>Explicitness</th>
<th>Modularity</th>
<th>MIG</th>
<th>BetaVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactorVAE</td>
<td>0.79 ± .03</td>
<td>0.55 ±.04</td>
<td>0.49 ±.05</td>
<td>0.84 ± .01</td>
<td>0.72 ±.02</td>
<td>0.24 ±.03</td>
<td>0.94 ±.02</td>
</tr>
<tr>
<td>InfoGAN (modified)</td>
<td>0.76 ±.06</td>
<td>0.62 ±.06</td>
<td>0.57 ±.06</td>
<td>0.82 ± .04</td>
<td>0.98 ±.01</td>
<td>0.34 ±.04</td>
<td>0.90 ±.07</td>
</tr>
<tr>
<td>InfoGAN-CR</td>
<td>0.82 ± .02</td>
<td>0.66 ± .01</td>
<td>0.53 ± .02</td>
<td>0.81 ± .01</td>
<td>0.97 ±.00</td>
<td>0.38 ±.02</td>
<td>0.89 ±.02</td>
</tr>
</tbody>
</table>

Figure 8: On 3DTeapots dataset, InfoGAN-CR models selected with ModelCentrality improves significantly upon those from supervised hyper-parameter tuning (see the previous table). The standard errors are less than 0.01 and we omit them in this table.
Why do we need unsupervised model selection and NOT supervised hyperparameter tuning?
Why do we need unsupervised model selection and NOT supervised hyperparameter tuning?

The practice of supervised hyperparameter tuning is problematic;
Why do we need unsupervised model selection and NOT supervised hyperparameter tuning?

The practice of supervised hyperparameter tuning is problematic;

- In important real-world applications we do not have ground truth data,
- More complex model with a larger space to tune could get better scores by an extensive search
Why do we need unsupervised model selection and NOT supervised hyperparameter tuning?

The practice of supervised hyperparameter tuning is problematic;

- In important real-world applications we do not have ground truth data,

- More complex model with a larger space to tune could get better scores by an extensive search

We resolve this issue using unsupervised model selection.
Why do we need unsupervised model selection and NOT supervised hyperparameter tuning?

The practice of supervised hyperparameter tuning is problematic;

- In important real-world applications we do not have ground truth data,
- More complex model with a larger space to tune could get better scores by an extensive search

We resolve this issue using unsupervised model selection.

As we saw in the last table, the unsupervised model selection finds a better model than that found via supervised hyperparameter tuning.
Suppose there is a notion of an optimal disentanglement that we want to discover from the data.
Suppose there is a notion of an optimal disentanglement that we want to discover from the data.

We start from a premise that well-disentangled models should be close to that optimal model, and hence also close to each other.
Suppose there is a notion of an optimal disentanglement that we want to discover from the data.

We start from a premise that well-disentangled models should be close to that optimal model, and hence also close to each other.

To measure similarity between models, we borrow insights from a long line of research in measuring disentanglement.
Suppose there is a notion of an optimal disentanglement that we want to discover from the data.

We start from a premise that well-disentangled models should be close to that optimal model, and hence also close to each other.

To measure similarity between models, we borrow insights from a long line of research in measuring disentanglement.

In particular, prior work suggests that models with good disentanglement metrics tend to exhibit qualitatively good disentanglement properties, e.g., via latent traversals.
Suppose there is a notion of an optimal disentanglement that we want to discover from the data.

We start from a premise that well-disentangled models should be close to that optimal model, and hence also close to each other.

To measure similarity between models, we borrow insights from a long line of research in measuring disentanglement.

In particular, prior work suggests that models with good disentanglement metrics tend to exhibit qualitatively good disentanglement properties, e.g., via latent traversals.

This suggests that disentanglement scores can be used to measure how close the disentangled latent codes of one model are to the latent codes of another.
Consider a trained model $G : \mathbb{R}^k \rightarrow \mathbb{R}^n$ as mapping a disentangled latent code to the image.
Consider a trained model $G : \mathbb{R}^k \rightarrow \mathbb{R}^n$ as mapping a disentangled latent code to the image.

Existing metrics also require the corresponding encoder $Q : \mathbb{R}^n \rightarrow \mathbb{R}^k$ that maps samples to estimated disentangled latent factors.
Consider a trained model $G : \mathbb{R}^k \rightarrow \mathbb{R}^n$ as mapping a disentangled latent code to the image.

Existing metrics also require the corresponding encoder $Q : \mathbb{R}^n \rightarrow \mathbb{R}^k$ that maps samples to estimated disentangled latent factors.

The FactorVAE metric in [5] of a trained generative model $G$ measures how well its encoder $Q$ can estimate, from real samples, the true latents of real samples (for example the ground truths of a dataset with also the true disentangled latent factors).
Consider a trained model $G : \mathbb{R}^k \to \mathbb{R}^n$ as mapping a disentangled latent code to the image.

Existing metrics also require the corresponding encoder $Q : \mathbb{R}^n \to \mathbb{R}^k$ that maps samples to estimated disentangled latent factors.

The FactorVAE metric in [5] of a trained generative model $G$ measures how well its encoder $Q$ can estimate, from real samples, the true latents of real samples (for example the ground truths of a dataset with also the true disentangled latent factors).

Instead of the original FactorVAE score, we use other trained models as a surrogate for the ground truths.
ModelCentrality - 3

ModelCentrality treats the distribution of another model $G_j$ as the ground truth.
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$. 
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$.

The similarity metric is an instance of self-supervision, as we treat one model as the target label and no ground-truth labels are needed.
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$.

The similarity metric is an instance of self-supervision, as we treat one model as the target label and no ground-truth labels are needed.

We can compute the similarity from $G_j$ to $G_i$ as the following:
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$.

The similarity metric is an instance of self-supervision, as we treat one model as the target label and no ground-truth labels are needed.

We can compute the similarity from $G_j$ to $G_i$ as the following:
- Generating samples using the target model $G_j$
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$.

The similarity metric is an instance of self-supervision, as we treat one model as the target label and no ground-truth labels are needed.

We can compute the similarity from $G_j$ to $G_i$ as the following:

- Generating samples using the target model $G_j$
- Passing those samples through the encoder $Q_i$ of model $G_i$ to estimate its latents,
ModelCentrality treats the distribution of another model $G_j$ as the ground truth.

Given two trained models: $G_i$ and $G_j$, we can measure how well the encoder $Q_i$ can estimate, from the generated samples of model $G_j$, the learned latents of the generated samples of model $G_j$.

The similarity metric is an instance of self-supervision, as we treat one model as the target label and no ground-truth labels are needed.

We can compute the similarity from $G_j$ to $G_i$ as the following:

- Generating samples using the target model $G_j$
- Passing those samples through the encoder $Q_i$ of model $G_i$ to estimate its latents,
- Using these estimated latents to evaluate the FactorVAE metric by using the latents generated by target model $G_j$ as ground truths.
We denote the symmetric similarity matrix as $B \in \mathbb{R}^{N \times N}$. 
We denote the symmetric similarity matrix as $B \in \mathbb{R}^{N \times N}$.

The similarity between a model $i$ and a model $j$ as the following:

$$B_{ij} = (1/2)(A_{ij} + A_{ji})$$
We denote the symmetric similarity matrix as $B \in \mathbb{R}^{N \times N}$.

The similarity between a model $i$ and a model $j$ as the following:

$$B_{ij} = (1/2)(A_{ij} + A_{ji})$$

where $A_{ij}$ is the disentanglement score (FactorVAE score as the disentanglement metric) achieved by model $G_i$ treating model $G_j$ as the target model with the way we explained in the previous slides.
ModelCentrality - 5

ModelCentrality a novel model selection scheme based on self-supervision.
ModelCentrality is a novel model selection scheme based on self-supervision.

ModelCentrality builds upon a premise that well-disentangled models are close together, with the closeness measured by a popular disentanglement metric from [5].
ModelCentrality is a novel model selection scheme based on self-supervision.

ModelCentrality builds upon a premise that well-disentangled models are close together, with the closeness measured by a popular disentanglement metric from [5].

We define ModelCentrality as the medoid (multi-dimensional generalization of the median) of a set of models, computed under the disentanglement metric from [5].
ModelCentrality: a novel model selection scheme based on self-supervision.

ModelCentrality builds upon a premise that well-disentangled models are close together, with the closeness measured by a popular disentanglement metric from [5].

We define ModelCentrality as the medoid (multi-dimensional generalization of the median) of a set of models, computed under the disentanglement metric from [5].

ModelCentrality assigns centrality scores to each trained model based on the self-supervised labels defined by the closeness to other models.
**ModelCentrality** a novel model selection scheme based on self-supervision.

**ModelCentrality** builds upon a premise that well-disentangled models are close together, with the closeness measured by a popular disentanglement metric from [5].

We define **ModelCentrality** as the medoid (multi-dimensional generalization of the median) of a set of models, computed under the disentanglement metric from [5].

**ModelCentrality** assigns centrality scores to each trained model based on the self-supervised labels defined by the closeness to other models.

We define the **ModelCentrality** of a model $i$ as $s_i = \frac{1}{n-1} \sum_{i \neq j} B_{ij}$.
The proposed ModelCentrality is derived from FactorVAE scores [5], which is popular, principled, and demonstrated to be a stable measure of disentanglement.
The proposed ModelCentrality is derived from FactorVAE scores [5], which is popular, principled, and demonstrated to be a stable measure of disentanglement.

It is shown in the experiments in the paper that the model selection scheme ModelCentrality outperforms state-of-the art schemes in [11] (UDR Lasso, UDR Spearman).
Summary and some results
Summary and some results

InfoGAN-CR adds a contrastive regularizer (CR) that combines self-supervision with the most natural measure of disentanglement: latent traversal. In the paper, it was created a self-supervised learning task of multi-way hypothesis tests over the latent codes and encouraging the generator to succeed at those tasks.

InfoGAN-CR together with ModelCentrality achieves the best disentanglement across all metrics in the literature. It is suggested in the paper that disentangling VAEs and GANs require fundamentally different techniques because the total correlation regularization, a popular technique for disentangling VAEs, do not improve disentanglement in GAN training.

The proposed CR regularization could be used in any application of disentangled GANs, e.g., hierarchical image representation or reinforcement learning.
Summary and some results

- InfoGAN-CR adds a contrastive regularizer (CR) that combines self-supervision with the most natural measure of disentanglement: latent traversal.
Summary and some results

- **InfoGAN-CR** adds a contrastive regularizer (CR) that combines **self-supervision** with the most natural measure of disentanglement: **latent traversal**.

- In the paper, it was created a **self-supervised learning task** of multi-way hypothesis tests over the latent codes and encouraging the generator to succeed at those tasks.
Summary and some results

- InfoGAN-CR adds a contrastive regularizer (CR) that combines self-supervision with the most natural measure of disentanglement: latent traversal.

- In the paper, it was created a self-supervised learning task of multi-way hypothesis tests over the latent codes and encouraging the generator to succeed at those tasks.

- InfoGAN-CR together with ModelCentrality achieves the best disentanglement across all metrics in the literature.
• InfoGAN-CR adds a **contrastive regularizer (CR)** that combines **self-supervision** with the most natural measure of disentanglement: **latent traversal**.

• In the paper, it was created a **self-supervised learning task** of multi-way hypothesis tests over the latent codes and encouraging the generator to succeed at those tasks.

• InfoGAN-CR together with ModelCentrality achieves the best disentanglement across all metrics in the literature.

• It is suggested in the paper that disentangling VAEs and GANs require fundamentally different techniques because the total correlation regularization, a popular technique for disentangling VAEs, do not improve disentanglement in GAN training.
Summary and some results

- **InfoGAN-CR** adds a contrastive regularizer (CR) that combines self-supervision with the most natural measure of disentanglement: latent traversal.

- In the paper, it was created a self-supervised learning task of multi-way hypothesis tests over the latent codes and encouraging the generator to succeed at those tasks.

- InfoGAN-CR together with ModelCentrality achieves the best disentanglement across all metrics in the literature.

- It is suggested in the paper that disentangling VAEs and GANs require fundamentally different techniques because the total correlation regularization, a popular technique for disentangling VAEs, do not improve disentanglement in GAN training.

- The proposed CR regularization could be used in any application of disentangled GANs, e.g., hierarchical image representation or reinforcement learning.

Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, et al. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. 
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.

Generative adversarial nets.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel.

Infogan: Interpretable representation learning by information maximizing generative adversarial nets.
Hyunjik Kim and Andriy Mnih.  
Disentangling by factorising.  

Cian Eastwood and Christopher KI Williams.  
A framework for the quantitative evaluation of disentangled representations.  

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan.  
Variational inference of disentangled latent concepts from unlabeled observations.  

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 

*beta-vae: Learning basic visual concepts with a constrained variational framework.*

2016.

Sunny Duan, Loic Matthey, Andre Saraiva, Nicholas Watters, Christopher P Burgess, Alexander Lerchner, and Irina Higgins. 

Unsupervised model selection for variational disentangled representation learning. 

Thank You For Listening