# Generating Election Data Using Deep Generative Models

**Jui Chien Lin**[1], **Farhad Mohsin**[2], **Sahith Bhamidipati**[2], **Lirong Xia**[2]

[1]Tsinghua University
[2]Rensselaer Polytechnic Institute

**Abstract**

Generating synthetic election data using deep generative models, such as generative adversarial networks (GAN) and variational autoencoders (VAE), is a previously unexplored field of deep learning. Voting or preference aggregation-related algorithms are normally tested on little available real election data and more synthetic election data. To generate this synthetic data, statistical models for ranking such as random utility models or distance-based models are generally used. Some works focus on simpler assumptions based on the uniform distribution, such as impartial culture or single-peaked preferences. In this work, we propose the usage of GANs and VAEs, which are powerful and proven generative models. We train these generative models using real election data with two goals: generate realistic data that it closely resemble the training data; and generate this data such that it does not overfit, simply memorizing the training data, thus diversifying the overall election data. We present evaluation measures and objectives for generative models that are specifically suited for elections. Our work indicates that GAN and VAE manage to generate reasonably high quality data without overfitting on the training data.

## 1 Introduction

Voting is the most popular method for preference aggregation to make group decisions, thus making it a very important topic of social choice research. Computational social choice has also been an important topic to compute the likelihood of scenarios like manipulation, bribery or paradoxical outcomes of voting rules. Most of the empirical analysis on such tasks are often done on synthetic data due to lack of available real preference datasets. Also, a few recent works have looked into designing new voting rules by training a classifier on synthetic data or predicting the will of the people directly to determine the voting rule. All of these tasks mostly make use of synthetic preference profiles that are sampled using simple distributions, e.g., uniform distribution over rankings (also called impartial culture), statistical models for rankings such as random utility models (e.g., Plackett-Luce), or distance-based models (e.g., Mallows). Some real election data sets exist, but in many cases they are limited. For example, political elections usually occur in long gaps and many of them simply use plurality voting which only considers the top choice of voters. This contributes to the rarity of election data being readily available in bulk, especially for more deep and complex models. Some resources can be found in data libraries like PrefLib or Fairvote, but even those are limited to an extent. For example, PrefLib has only 315 total election datasets that have strict full order ranking information.

This led us to the task of generating synthetic election data that is similar to existing ones. Deep generative models such as generative adversarial networks (GAN) and variational auto-encoders (VAE) have been used successfully for this purpose in other domains. The most successful examples are in Computer Vision of generating highly realistic images of a wide variety of subjects, ranging from human faces to inanimate objects. In this paper, we want to explore if we can use similar techniques to generate synthetic election data that can contribute to the limited realm of election data. We ask two main research questions in this paper, the second stemming from the first:

*1. How can we use modern deep learning techniques to generate synthetic but realistic election data similar to some existing dataset?*

*2. How to effectively evaluate goodness of generated election datasets in a quantitative manner?*

The question of evaluation requires us to expand on what we want from synthetically generated data. We want the generated data to closely resemble the training data, but we also do not want simple sampling or boosting-based techniques that simply make copies of existing data. That is, we want to minimize overfitting or memorization of the training data.

**Our contributions.** Our primary contribution in this paper is twofold. First, we use deep generative models, in particular generative adversarial networks (GAN) and variational autoencoders (VAE), to generate new data similar to existing preference profiles. Second, we adapt existing evaluation measures in the generative model literature to measure the quality of our generated election data.

In this paper, we focus on preference profiles with strict (no ties) and complete rankings (no partial rankings). We conduct extensive experiments by training models on real preference profiles that were created using the Netflix preference dataset

To be presented at AI4SG workshop at AAAI-23.

## Related Work

Common methods of generating sample preference profiles is by assuming some sort of distribution. The most common, yet unrealistic, is to sample individual voters' preferences uniformly from the set of all possible votes and then combine them in a preference profile. Other methods assume some restrictions on the preferences, such as single-peaked preferences (Moulin 1980; Brams, Jones, and Kilgour 2002). More complex methods of generating preference profiles include using a statistical model for rankings such as random utility models (Azari Soufiani, Parkes, and Xia 2012) or distance-based models. Popular examples of random utility models include the Plackett-Luce model (Plackett 1975; Luce 1959) and Thurstone's case V model (Thurstone 1927). Likewise, Mallows model (Mallows 1957) is a common distance-based model. These statistical models are also used in learning preferences of voters and thus can be used to generate synthetic preference profiles. Beyond that, spatial models have also been used where voters and alternatives are considered as belonging to a latent space, with the distance between them indicating preference (e.g., (Merrill 1985)).

As mentioned in the introduction, there are not many large election datasets with rich properties. For example, datasets are available for US Presidential and Senate elections (Data 2020a,b), but being single-winner elections, it is hard to learn too much from the information. PrefLib (Mattei and Walsh 2013) has a collection of different types of election data. In this paper, we focus on strict and complete rankings. We notice that for the strict and complete rankings in PrefLib, most of them are generated from user preferences in the Netflix Prize dataset (Bennett, Lanning et al. 2007). We take motivation from that and create our realistic preference profiles by sampling users' strict rankings over movies as expressed in the Netflix Prize dataset.

While generative models have always existed, examples including aforementioned Plackett-Luce and Mallows models to more general mixture of Gaussian models, it is only with the advancement of deep learning models that generative models have started to become adept at generating very realistic samples in different domains (see (Harshvardhan et al. 2020) for a recent survey). Early methods in simulating larger datasets based on a small real dataset include sampling and boosting, some making use of nearest neighbor-based sampling (Chawla et al. 2002). Kingma and Welling (2013) introduced Variational Autoencoders (VAE), which makes the assumption that samples belong to some underlying latent distribution. By training an encoder and a decoder to learn the latent distribution, VAEs can further be used to sample from the latent distribution, acting as a generative model. The use of deep neural networks for both the encoder and decoder causes the VAE to be a rich generative model, due to being able to learn very good latent models. Beyond the goal of learning latent models, Generative Adversarial Networks (GAN) (Goodfellow et al. 2020) were developed with the goal of being a good generative model. GANs work by using a pair of deep learning models together, a generative model whose goal is to generate new data, and an adversarial model, whose goal is to differentiate between real and generated data. By training both models together, GANs are able to generate very realistic data, as particularly seen in the field of computer vision by generating photo-realistic images. Different variations of GANs have been created, both from functional and architectural perspective. Conditional GANs make use of conditions to specifically generate different types of data. On the other hand architectures like DCGAN (Radford, Metz, and Chintala 2015) make use of rich models like convolutional neural networks to improve the generative performance of GANs.

We are not aware of any work that tries to quantitatively evaluate the quality of generated election data. So we instead look into work that propose evaluation methods that can measure different aspects generative models. Many of these measures were designed to evaluate GANs with computer vision tasks in mind, trying to find image similarity etc. but most can be generalized to evaluate any generative model. There are many proposed measures such as Inception score (Salimans et al. 2016), precision and recall (Lucic et al. 2018) etc. but the most commonly used similarity meausre is Freć het Inception Distance (FID) (Heusel et al. 2017). The main idea behind FID is to learn a latent representation for each image and assume that the latent distributions are sampled from a multivariate Gaussian distribution and fiind the distances between the distributions. This idea can be generalized to compute Frechet distance from any latent distribution that can be learned. Because of the dependency on domain information for most of these measures, new measures depending on classifier accuracy trained and tested on generated data was proposed: GAN-train and GAN-test (Shmelkov, Schmid, and Alahari 2018). Beyond generating realistic samples, another concern for generative models is overfitting, in whether the model is overfitting on training set, even memorizing some training data. The simplest method of finding if memorization has occured is by computing the nearest neighbor of generated samples in the training set and seeing if samples very close to training samples have been computed (Bai et al. 2021). See (Borji 2019, 2022) for a comprehensive study of evaluation measures.

# 2 Preliminaries

## Elections and preference profiles

Assume we have $n$ voters and a set of $m$ alternatives, $\mathcal{A}$. Let every voter have a strict ordering (ranking) over the full set of $\mathcal{A}$. Let $\mathcal{L}(\mathcal{A})$ be the set of all rankings. Thus, every voter will have a ranking in $\mathcal{L}(\mathcal{A})$. We call the set of $n$ rankings a *preference profile*. For $m$ alternatives, there are $m!$ possible rankings. Thus, we can represent a preference profile with an $m!$-length vector $\mathbf{x}$, where element $x_\ell$ indicates how many people have the $\ell$-th ranking as their preference. We can also consider the normalized version of $\mathbf{x}$ as a preference profile as well, where the sum of elements is 1. Thus a preference profile, in our case, can be represented by any $\mathbf{x} \in \mathbb{R}^{m!}$

**Netflix dataset:** As the training set for our generative models, we use real preference data from Netflix users to create three-alternative and four-alternative preference profile. This is similar to how majority of the strict order pref-

erence profiles in PrefLib (Mattei and Walsh 2013) was created as well. For $m = 3$ or $4$, we sample $m$ movies at random, and find all users who have strict rankings for all $m$ movies. That gives us a single preference profile. For both $m = 3$ and $4$, in this way, we can sample a large number of preference profiles. However, we understand that in realistic scenarios we will not have countless amount of data. So, we limit ourselves to 1000 preference profiles for training and 1000 preference profiles for validation.

### Generative models

The most common and simplest method of data generation is up-sampling, where random samples in the data are chosen as the "generated" data and increase the data size. The drawback to this method, though, is the resulting duplicate samples in the data. For data-driven models, the repeated samples can ruin the training of the model by either forcing a lack of convergence or causing the model to learn these samples more heavily than the others, leading to reduced generalization. A good generative model should ideally be free from this issue. In this subsection, we discuss models that we explore in this paper.

**Synthetic Minority Over-sampling Technique** The Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al. 2002) introduced the notion of k-nearest neighbor(kNN) based up-sampling for minority data to generate new balanced dataset. But the main idea can be used to without consideration of data class as a generative model by itself. The idea of SMOTE is that based on the k-nearest neighbor samples of each sample point, N neighboring points are randomly selected for difference multiplication by a threshold in the range of [0,1] to synthesize the data. The core of this algorithm is that by using neighboring points on the feature space, the samples chosen to extend the data maintain the distribution of the previous data set, so the newly generated data set reliably provides a better sample of the original data set. This eliminates some of the issues found in the traditional sampling method, as models that train on the generated data are learning from the same distribution as the training data.

**Variational autoencoder** With the introduction of deep networks to tackle big data and computation problems, generative networks also developed to incorporate these powerful models. A prime example of this is the Variational autoencoder (Kingma and Welling 2013), which involves training a model that uses an encoder model to compress the features of input data into a latent space, then learns to decode that latent space back into an equivalent representation in the original feature space with a decoder model. After the overall model is trained in this fashion, synthetic data can be generated by passing noise into the decoder. The objective function of VAE is as below:

$$\mathbf{E}_{h\sim\mathbf{E}(h|x)} \log \mathbf{D}(x|h) - \mathbf{KL}(\mathbf{E}(h|x)\|\mathbf{Q}(h))$$

The former part uses $\mathbf{E}(h|x)$ to infer and maximize the likelihood $\log \mathbf{D}(x|h)$ of the latent message $h$. In other words, it lowers the reconstructed error between the input and the the synthesized data. The latter part, on the other hand, restricts the estimated $h$ to not be too different from the prior latent message so that it can meet the assumption of the VAE and achieve a better reconstruction result.

**Generative adversarial network** The objective function of the VAE tunes the model to solely make the reconstructed error as small as possible, which limits the creativity of the model and thus the uniqueness of the generated data. To combat this hindrance, in 2014, Goodfellow et al. proposed the generative adversarial network (GAN) (Goodfellow et al. 2020), which is designed specifically to learn to generate realistic data based on its distribution. Most current generation tasks are based on this GAN model. The concept of a GAN is that it contains a generator and a discriminator. The discriminator learns to distinguish real data from fake data, while the generator learns to fool the discriminator with its output. Through this method, ideally, the generator eventually produces realistic enough synthetic data that the discriminator believes the data is real. This competition process is what allows the GAN to learn the distribution of the data, rather than simply a reconstruction of the data. The objective function of GAN is as below:

$$\min_G \max_D \big(\mathbb{E}_{\mathbf{x}\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{\mathbf{z}\sim p_z(z)}[log(1 - D \circ G(z))]\big)$$

The generator $G$ samples the noise $z$ from a gaussian distribution and transforms it from the latent space into the feature space as a new sample $G(z)$. Then, $G(z)$ and the training data $x$ are used as input for the discriminator $D$ to distinguish real from fake data, which then adjusts the parameters of both networks.

## 3 Election Data Generation Using Generative Models

Althought GANs have been used in many fields, usually with outstanding results, there are few applications of GANs with regards to election data. Compared with other types of data such as images, time series, and text, there exists a unique pattern in election data. Many elections use just the top ranked alternative provided by voting agents to determine a winner. But in most voting rules, a full ranking is used, i.e., every voter provides a full ranking over all of the alternatives. In this paper, we propose a GAN to deal with voting profiles or preference profiles consisting of full rankings and no ties. First, we use the number of voters who rankings of the alternatives as features rather than the number of votes. These rankings are transformed into images to be fed into the network. From here, the GAN learns the distribution of the election data and the relationships within the rankings, and finally generates realistic synthetic samples.

The election data is in form as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \in \mathbb{R}^{N\times m!}$ representing there are *N* elections, with the *i*-th observation, $\mathbf{x}_i \in \mathbb{R}^{m!}$ corresponding to *m* alternatives and the feature size is $m!$, the total number of rankings over $m$ alternatives. The number of features is different for different number of alternatives, we discuss the preprocessing process to bring them to the same dimension below.

### Featurization of election data

We explain the featurization with examples. If there are three alternatives $A$, $B$ and $C$, then all possible rankings would be $A > B > C, A > C > B, B > A > C, B > C > A, C > A > B, C > B > A$. The preference profile contains the count of voters who has each preference as their ranking, almost like a histogram of the rankings. However, the feature size is in the factorials of the number of alternatives. For small number of alternatives, the feature size would be few. Besides, every ranking is represented by means of one value, but a single value does not contain enough information. Since CNN is good at capturing the spatial pattern consisting of many features, we do feature refinement by repeating each ranking, i.e. feature, and reshaping it into an image so that there would be enough features for CNN to learn. This also makes the number of features equal for any number of alternatives. The transformed data is displayed in Figure 1, and we can see that after the refinement, each feature, i.e. ranking, consisted of many pixels and data with different numbers of alternatives in the same dimensions.

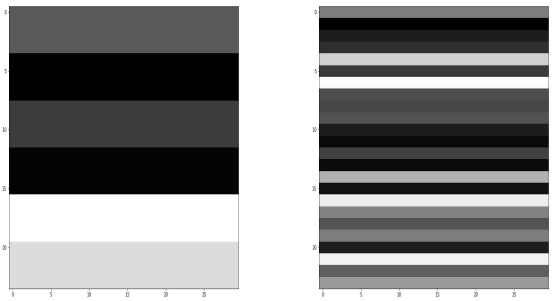

Figure 1: The left image is transformed from election data with 3 alternatives, and the right is from 4 alternatives. In left image, there are 6 different regions representing 6 different rankings, and the right one has 24 regions by the same logic.

For our experiments, we label the data with same amounts of alternatives as one class. This is necessary in particular in our GAN model learning because we use the class as conditions and learn conditional GANs (CGAN) (Mirza and Osindero 2014) in order to use the common knowledge within the election data but not to confuse the training process.

### Generative model architecture

**GAN** After the election data are converted to images, we need to build a model that can capture all available information from the images and correctly learn from it. When building the images, lots of data is repeated to achieve correct sizes, making the image pattern generally much simpler than typical image data.

We choose the convolutional neural network (CNN) as the architecture of the GAN, which is good at capturing the spatial relation within the data. To ensure the model extracts only the necessary information - the difference in rankings - from the data, instead of utilizing a strong spatial calculation model as usually required for images, we adopt a conditional Deep Convolution GAN (DCGAN), in which the underlying generator and discriminator are both CNNs. The architecture of our version of this model is shown in Figure 2. In addition, to ensure that the sum of all features in a single sample add to 1 to match general normalized ranking data structure, we also add a penalty to restrict the sums of features for each sample in the synthetic data to as close to 1 as possible.

Specifically, the generator network is composed of 1 fully connecting layer (FCL) and 3 convolution layers. Firstly, the gaussian noise and condition, which indicate the number of the alternatives, would be concatenated, and projecting them into high dimension space by inputting them into the FCL, which has 1536 neurons. Then, calculated by a series of convolution computation, and after each convolution computation, the result would also pass the batch normalization and a ReLU activation. The first convolution layers has 256 filters, a kernel sizes of $3 \times 3$, strides $1 \times 2$, and valid padding, the second one has 128 filters, a kernel sizes of $3 \times 3$, strides $3 \times 3$, and equal padding, and the third one has 1 filters, a kernel sizes of $3 \times 3$, strides $2 \times 2$, and equal padding. After the computation of the generator, the noise would be converted to an array with the same dimension as the training data whose shape is $24 \times 30 \times 1$ (height, width, channel), as known as the generated data. Then, the generated data and training data would be input into discriminator which also has 1 FCL and 3 convolution layers. The FCL has 3072 neurons, and the first convolution layer has 256 filters, a kernel sizes of $4 \times 5$, strides $2 \times 2$, and equal padding, the second one has 256 filters, a kernel sizes of $4 \times 5$, strides $2 \times 2$, and equal padding, and the third one has 256 filters, a kernel sizes of $4 \times 5$, strides $2 \times 2$, and equal padding. Firstly, the data would be concatenated with its condition, and concatenated data would be calculated by a series of convolution calculation, and each convolution calculation comes by a ReLU activation. Then, the convoluted result would be projected into 1 by the FCL for a better comparison dimension of the real data and generated data. We use wasserstein distance (Arjovsky, Chintala, and Bottou 2017) to calculate the difference of the real data and generated data for a better stable training process.

**VAE** As for VAE generative model, the architecture of encoder and decoder is almost the same as the discriminator and generator of the GAN, which is discussed in detail above. The difference is that in VAE, we don't use condition, and we train different VAE for the data with different number of alternatives. Besides, mean square error is used to calculate the difference of the real and generated data.

## 4 Evaluation methods

Two main issues loom over data generation tasks: over-fitting and quality. As mentioned in the previous section, over-fitting refers to when a model learns its training data too closely, to the point where its output is essentially a copy of the data. The quality of output depends on how much the

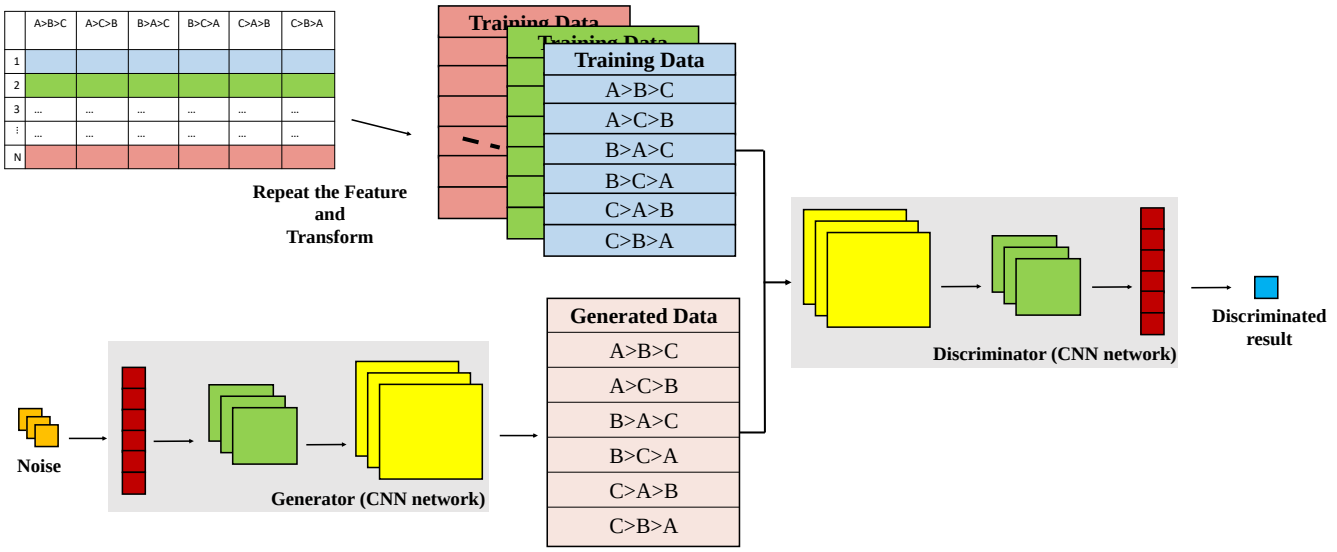


Figure 2: The architecture of our GAN.

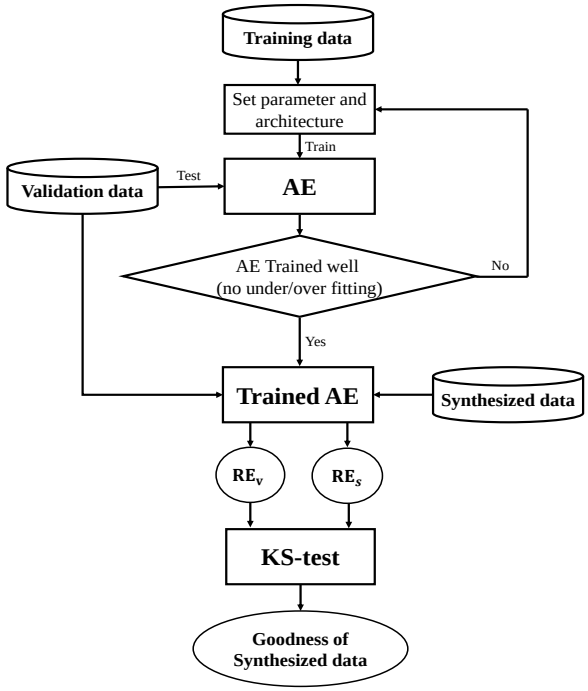


Figure 3: The Autoencoder evaluation procedure.

model learns the distribution of the training data instead of learning to emulate it as a whole. In this section, we propose some methods to tackle each of these issues: Similarity evaluation and Overfitting evaluation.

## Similarity evaluation

To our knowledge, a similarity evaluation metric for synthetic election data, that compares the data to training data does not exist. For this purpose, we appropriate a couple of popular evaluation measure in generative models, particularly in GAN literature and propose a new one. The new notion is autoencoder (AE) reconstruction error, which is domain-agnostic, thus can be used to evaluate any generative models. We also use the popularly used GAN-train and GAN-test evaluation measures. Finally, we adapt the popular Frechet Inception distance measure used for images for election data using Plackett-Luce models. We discuss each in detail below.

**Autoencoder(AE) reconstruction error** Although a GAN should ideally generate samples that are purely synthetic and not mimicking real data, the generated synthetic data should still be realistic enough to belong in the same distribution as its corresponding training data. To measure this relationship, we propose the Autoencoder (AE) evaluation method. An autoencoder model learns to reconstruct data by training on a data set with the objective output being the same training set. Autoencoders can be used in anomaly detection, which involves training the model using normal data so that when it is fed abnormal data, it can detect anomalies in the data through the effects on the reconstructed version of the data (Zong et al. 2018). The proposed AE evaluation method is inspired by the latter application in that it involves training an autoencoder on ”normal” election data and testing the same model on ”abnormal” election data. If we just feed generated data to a

trained autoencoder to see if it is considered abnormal data from the reconstruction error.

An autoencoder consists of two inner models: an encoder and a decoder. The encoder takes the training data set of the entire model as input, passes the data through convolution layers, and outputs a latent representation of the data. The dimensions of this latent space are dependent on the dimensions of the convolution layers used in the model. The decoder takes this latent representation as input, passes it through deconvolution layers, then reconstructs it back to the training labels.

The AE model used to evaluate our GAN uses an encoder with one convolution layer and a decoder with one deconvolution layer along with the output convolution layer. The input of the encoder is the input of the model, which is a set of images that represents election data. The convolution layer of the encoder has 16 filters, a kernel size of $2 \times 2$, a stride of 3, ReLU activation, and equal padding. Since, the deconvolution layer of the decoder has to result in the same size as the input, it uses the same parameters as the previous layer in the encoder. In other words, this layer also has 16 filters, a $2 \times 2$ kernel size, a stride of 3, ReLU activation, and equal padding. Lastly, the output convolution layer of the decoder constructs the data back to an image the same size as the input, so it has a single filter, a kernel size of $2 \times 2$, a stride of 1, sigmoid activation to ensure the output values are between 0 and 1, and equal padding. The overall model is trained using the MSE loss function and the Adam optimizer (Kingma and Ba 2014). The process of the AE evaluation method is depicted in Figure 3. The results of this method, along with other evaluation methods are discussed in the Experimental Results section of the paper.

**GAN-train and GAN-test** (Shmelkov, Schmid, and Alahari 2018) Shmelkov, Schmid, and Alahari (2018) first proposed GAN-train and GAN-test as evaluation measures that in the context of images approximate quality of the image and diversity of the generative model. GAN-train is the accuracy when a classifier is trained on the generative model's training set and then tested on generated data. On the other hand, GAN-test is the accuracy when a classifier is trained on the generated data and then tested on testing set. Note that while the naming indicates that this was designed to work with GANs, there is nothing in the definition that prevents it from being used for other generative models as well. For the voting scenario, our concern is what to use as a classifier. For images, most images are associated with a label and that is what they choose. For the voting scenario, we choose the classification task as the task of predicting a voting rule winner. This, by itself, is an interesting work. As in recent times, using ML to design voting rules has been a topic of interest (Anil and Bao 2021; Mohsin et al. 2022). So, we thought that a classifier mimicking an existing voting rule would be a natural case for the GAN-train and GAN-test measures.

**PLackett-Luce Fréchet Distance** The Plackett-Luce (PL) model for rankings is defined as follows:

The parameter space is $\Theta = \{\vec{\theta} = \{\theta^j | 1 \leq j \leq m, 0 < \theta^j < 1, \sum_{j=1}^{m} \theta_j = 1\}\}$. The sample space is $\mathcal{L}(\mathcal{A})^n$. Given a parameter $\vec{\theta} \in \Theta$, the probability of any full ranking $\sigma = a_{j_1} \succ a_{j_2} \succ \ldots \succ a_{j_m}$ is $\Pr_{\text{PL}}(\sigma|\vec{\theta}) = \prod_{p=1}^{m-1} \frac{\exp(\theta^{j_p})}{\sum_{q=p}^{m} \exp(\theta^{j_q})}$.

Given a particular preference profile, we can fit a PL model on the data using expectation-maximization type methods (Hunter 2004). So, for a training set of and generated set of preference profiles, we can learn separate PL parameters ($m$ parameters for an $m$-alternative scenario) for each of the preference profile. We can assume that these PL parameters are a latent representation of the preference profiles. Then, if we assume that the parameters are samples from a multivariate $m$-dim Gaussian distribution, we get two multivariate Gaussians: for the generated data $(\mu_G, \Sigma_G)$ and training data $(\mu_T, \Sigma_T)$. Then, we can compute the Frechet distance (Fréchet 1948; Vaserstein 1969) as follows:

$$d^2 = (\mu_T - \mu_G)^2 + trace(\Sigma_T + \Sigma_G - 2(\Sigma_T \Sigma_G)^{1/2}$$

## Overfitting Evaluation

The other important measure for evaluation is whether the model is simply memorizing training data and generating samples exactly like or close to the training data. We define memorization distance to evaluate this.

**Memorization Distance** To check if the GAN is simply memorizing the training data instead of learning its distribution, we use the memorization distance (Borji 2022) to evaluate the level of overfitting occurring:

$$s\left(S_g, S_t\right) = \frac{1}{|S_g|} \sum_{x_g \in S_g} \min_{x_t \in S_t} \left(1 - \frac{|\langle x_g, x_t \rangle|}{|x_g| \cdot |x_t|}\right) \quad (1)$$

Memorization distance is calculated from the inner product between the training set $S_t$ and generated dataset $S_g$. The distance stands for the similarity of the two dataset. The lower the value is, the more memorization is done by the generative model.

# 5 Experimental Results

**Experimental setup:** To do the experiments, we compare five generative models. We compare generated datasets from GAN, VAE, SMOTE and simple sampling-based models. The SMOTE baseline is implemented using 5-nearest neighbors. For each model, we generate dataset for three-alternative and four-alternative scenarios. For each scenario, we sample 200 different datasets, each containing 100 preference profiles. Then for each of the 200 datasets, we get the mean value of all of the measures. Then, using the 200 values, we do pairwise Kolmogorov-Smirnov (KS) tests between different models to see if one performs better than the other in terms of a feature in a statistically significant way. We present the results separately for three alternative and four alternative elections.

**Quality measures:**

**Autoencoder (AE) reconstruction error** (lower is better): For four alternatives, we fine SMOTE is better than GAN/VAE. Both GAN and VAE have very similar AE reconstruction errors. The statistical test couldn't differentiate

| | Quality | | | | Overfitting |
|---|---|---|---|---|---|
| | AE Reconstruction Error | PL Frechet Distance | GAN-train | GAN-test | Memorization |
| GAN | 0.11 | 1.72E-03 | 0.96 | 0.78 | 1.75E-02 |
| VAE | 0.087 | 4.89E-02 | 0.96 | 0.80 | 3.03E-02 |
| SMOTE | 0.109 | 1.17E-01 | 0.96 | 0.78 | 3.36E-03 |
| Sample without replacement | 0.242 | 5.23E-03 | 0.93 | 0.74 | 0 |
| Sample with replacement | 0.239 | 5.23E-03 | 0.93 | 0.75 | 0 |

Table 1: Various quality and overfitting measures for generated data with four alternatives

| | Quality | | | | Overfitting |
|---|---|---|---|---|---|
| | AE Reconstruction Error | PL Frechet Distance | GAN-train | GAN-test | Memorization |
| GAN | 0.059 | 3.33E-04 | 0.99 | 0.96 | 2.27E-03 |
| VAE | 0.037 | 1.38E-02 | 0.98 | 0.97 | 3.25E-03 |
| SMOTE | 0.058 | 2.63E-03 | 0.97 | 0.97 | 7.33E-04 |
| Sample without replacement | 0.06 | 2.81E-03 | 0.96 | 0.98 | 0 |
| Sample with replacement | 0.059 | 6.62E-04 | 0.98 | 0.98 | 0 |

Table 2: Various quality and overfitting measures for generated data with three alternatives

between the errors for GAN/VAE. The error for SMOTE is the lowest. Unexpectedly though, the sample based methods have high reconstruction errors. However, the lower reconstruction error for all of the deep generative models indicate that the generated elections are not too different from the training data. For three alternative elections, VAE has the lowest AE reconstruction errors. But again, none of the reconstruction errors are significantly worse than the sample-based methods, which indicates that all models are generating similar data to training data.

**PL-based Frechet distance (PLFD)** (lower is better):

From better to worse: Sample based methods and GAN $>$ VAE $>$ SMOTE. The PLFD is lowest for GAN among the deep generative models. It is closest to the sample based baselines, but that is not unexpected. Since the samples have same samples as in the training set, chances of learning very similar PL values increase. The same conclusions are seen for both four-alternative and three-alternative elections.

**GAN-train and GAN-test for Borda** (higher is better): As mentioned before, the classification task we chose was to predict Borda winner. We also choose a tie-breaking method in lexicographic tie-breaking which makes the learning task difficult. We use a simple SVM model for the training with normalized preference profile as input feature. For all of the models, both GAN-test and GAN-train values are very similar. Doing pairwise comparisons using the KS-test for the accuracy values, all distributions seem to be identical. So, we cannot comment on comparative quality. But the reasonably high accuracy for a 4-class and a 3-class classification task indicates that the quality of the generated data is high.

**Overfitting measure**:

**Memorization distance** (higher is better):

In terms of best to worst: VAE $>$ GAN $>$ SMOTE and sampled methods. We note that the VAE performs the best in this measure. This means that for the generated data, the closest neighbor in training dataset is quite far away, which means no memorization is occurring. For GANs as well, while the memorization distance is lower than VAE, it is still high enough that it is not memorizing training samples.

**Summary:** In summary, we can say that GAN and VAE are best in terms of memorization. For quality measures, SMOTE is best according to AE reconstruction error for three alternatives but VAE is best for four alternatives. GAN best according to PL-measure for both cases. All models have similar evaluation measure values otherwise. It seems to us that both GAN and VAE has managed to generate reasonably good data with low memorization of training data with GAN-generated data being slightly ahead in terms of the quality measures.

## 6 Conclusion and Future Work

We contribute to the field of data generation by using deep generative models, namely GAN and VAE, to generate synthetic election data resembling preference profiles of various sets of options. In addition, we apply various evaluation techniques existing in generative model literature to our generated data in the domain of social choice. Each of our trained generative models provide synthetic data that achieve commendable scores across the chosen evaluation metrics focused on testing for realness and diversity in generated data. Both processes are novel to the field of social choice and generative models, and their success allows us to introduce them as a new baseline for future work focused on generating unique yet realistic synthetic election data.

From here, we can expand on our work by designing a better method for feature engineering to extract only the critical information from every feature in the election data. Although our current one is successful, finding improvements in the structure of the features could lead to cleaner data or an increased efficiency of training with the data. With possible restructuring of the data, the model will likely have to be modified to capture the intricacies of the new data. Our current work provides a great stepping stone to many potential improvements to the quantity and quality of synthetic data in the social choice domain, and we will continue to explore those realms and outperform the previous best.

## References


Anil, C.; and Bao, X. 2021. Learning to Elect. *Advances in Neural Information Processing Systems*, 34: 8006–8017.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR.

Azari Soufiani, H.; Parkes, D. C.; and Xia, L. 2012. Random Utility Theory for Social Choice. In *Proceedings of Advances in Neural Information Processing Systems*, 126–134.

Bai, C.-Y.; Lin, H.-T.; Raffel, C.; and Kan, W. C.-w. 2021. On training sample memorization: Lessons from Benchmarking generative modeling with a large-scale competition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2534–2542.

Bennett, J.; Lanning, S.; et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, 35. New York.

Borji, A. 2019. Pros and cons of gan evaluation measures. volume 179, 41–65. Elsevier.

Borji, A. 2022. Pros and cons of GAN evaluation measures: New developments. *Computer Vision and Image Understanding*, 215: 103329.

Brams, S. J.; Jones, M. A.; and Kilgour, D. M. 2002. Single-Peakedness and Disconnected Coalitions. *Journal of Theoretical Politics*, 14(3): 359–383.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357.

Data, M. E. 2020a. Science Lab (2017). US President, 1976-2016. *Harvard Dataverse. Available online at https://doi.org/10.7910/DVN/42MVDX, checked on*, 9: 12.

Data, M. E. 2020b. Science Lab.(2018). County Presidential Election Returns 2000-2016. *Harvard Dataverse*.

Fréchet, M. 1948. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, 215–310.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Harshvardhan, G.; Gourisaria, M. K.; Pandey, M.; and Rautaray, S. S. 2020. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38: 100285.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hunter, D. R. 2004. MM algorithms for generalized Bradley-Terry models. In *The Annals of Statistics*, volume 32, 384–406.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Luce, R. D. 1959. *Individual Choice Behavior: A Theoretical Analysis*. Wiley.

Lucic, M.; Kurach, K.; Michalski, M.; Gelly, S.; and Bousquet, O. 2018. Are gans created equal? a large-scale study. *Advances in neural information processing systems*, 31.

Mallows, C. L. 1957. Non-null ranking model. *Biometrika*, 44(1/2): 114–130.

Mattei, N.; and Walsh, T. 2013. PrefLib: A Library of Preference Data. In *Proceedings of Third International Conference on Algorithmic Decision Theory*, Lecture Notes in Artificial Intelligence.

Merrill, S. 1985. A statistical model for Condorcet efficiency based on simulation under spatial model assumptions. *Public Choice*, 47(2): 389–403.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mohsin, F.; Liu, A.; Chen, P.-Y.; Rossi, F.; and Xia, L. 2022. Learning to Design Fair and Private Voting Rules. *Journal of Artificial Intelligence Research*, 75: 1139–1176.

Moulin, H. 1980. On strategy-proofness and single peakedness. *Public Choice*, 35(4): 437–455.

Plackett, R. L. 1975. The Analysis of Permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2): 193–202.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Shmelkov, K.; Schmid, C.; and Alahari, K. 2018. How good is my GAN? In *Proceedings of the European conference on computer vision (ECCV)*, 213–229.

Thurstone, L. L. 1927. A law of comparative judgement. *Psychological Review*, 34(4): 273–286.

Vaserstein, L. N. 1969. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3): 64–72.

Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *International Conference on Learning Representations*.