Generative Adversarial Networks for scRNA-Seq Analysis

Donald Dalton
Computational Biology
ddalton@andrew.cmu.edu

Akash Ramachandran
Computational Biology
akashr1@andrew.cmu.edu

1 Introduction

For the last decade the cost of next generation sequencing (NGS) has decreased at a rate outpacing Moore's Law [1]. As such there is an increasing amount of biological data available for large scale genomic studies. However, integration of diverse biological datasets is a notoriously challenging problem primarily due to the sensitivity of NGS platforms to variable experimental conditions and protocols. Often it is the case that comparisons between datasets from multiple sources is not possible without sophisticated batch correction techniques which are often case-specific, and thus do not generalize well out of the box [2]. Therefore there is a need for frameworks that can effectively integrate diverse datasets with minimal pre-processing, and extract biologically relevant features in the presence of batch effects.

For the last decade the cost of next generation sequencing (NGS) has decreased at a rate outpacing Moore's Law [1]. As such there is an increasing amount of biological data available for large scale genomic studies. However, integration of diverse biological datasets is a notoriously challenging problem primarily due to the sensitivity of NGS platforms to variable experimental conditions and protocols. Often it is the case that comparisons between datasets from multiple sources is not possible without sophisticated batch correction techniques which are often case-specific, and thus do not generalize well out of the box [2]. Therefore there is a need for frameworks that can effectively integrate diverse datasets with minimal pre-processing, and extract biologically relevant features in the presence of batch effects.

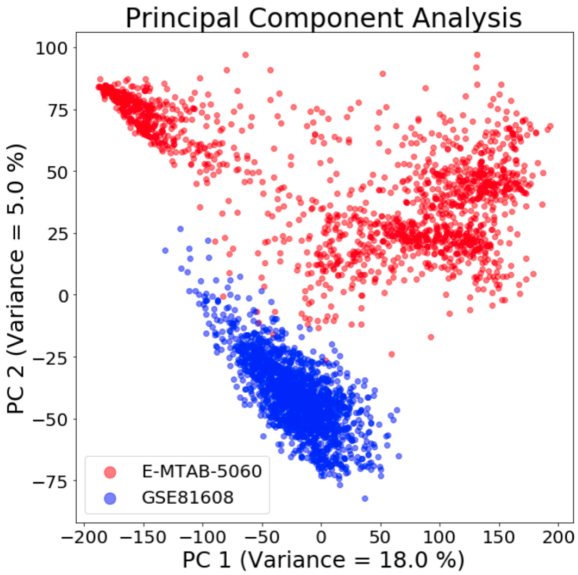

Figure 1: Model adapted from [3]. Left: The deep generative model. Middle: Greedy layer-wise pretraining in which the weights of each layer are treated as a single layer RBM. Right: The RBM's are unrolled to form a stacked autoencoder which is then further trained in a fine-tuning phase via the backpropagation algorithm.

For the last decade the cost of next generation sequencing (NGS) has decreased at a rate outpacing Moore's Law [1]. As such there is an increasing amount of biological data available for large scale genomic studies. However, integration of diverse biological datasets is a notoriously challenging problem primarily due to the sensitivity of NGS platforms to variable experimental conditions and protocols. Often it is the case that comparisons between datasets from multiple sources is not possible without sophisticated batch correction techniques which are often case-specific, and thus do not generalize well out of the box [2]. Therefore there is a need for frameworks that can effectively integrate diverse datasets with minimal pre-processing, and extract biologically relevant features in the presence of batch effects.

2 Background & Related Work

2.1 Generative Adversarial Networks (GANs)

GANs are a relatively new class of unsupervised machine learning algorithms introduced by Ian Goodfellow *et. al* in 2014. At a high level it involves 2 networks competing with each other in a zero-sum game. Previous studies have shown that GANs can be used to learn a meaningful latent representation of diverse datasets [2,3]. Furthermore, the generative nature of these models can capture complex non-linear relationships amongst input variables and provides a flexible framework for post-hoc interpretation and analysis of learned features. In the further sections we talk about the architecture of a conventional GAN and details about a variant called the Wasserstein GAN, which we have employed in this project

2.1.1 Architecture

The 2 networks competing with each other in a GAN are:

1. Generator: This is a multi-layer perceptron that takes a noise vector from a latent space and tries to map it to a higher-dimensional distribution of interest (in our case, gene expression vectors)
2. Discriminator (or Critic): This is also a multi-layer perceptron that tries to differentiate between real and generated gene expression vectors (or distributions). It's output (D) will, ideally after effective training, be 1 and 0 if the inputs are real and fake vectors respectively, Only the Discriminator gets to see real high-dimensional data.

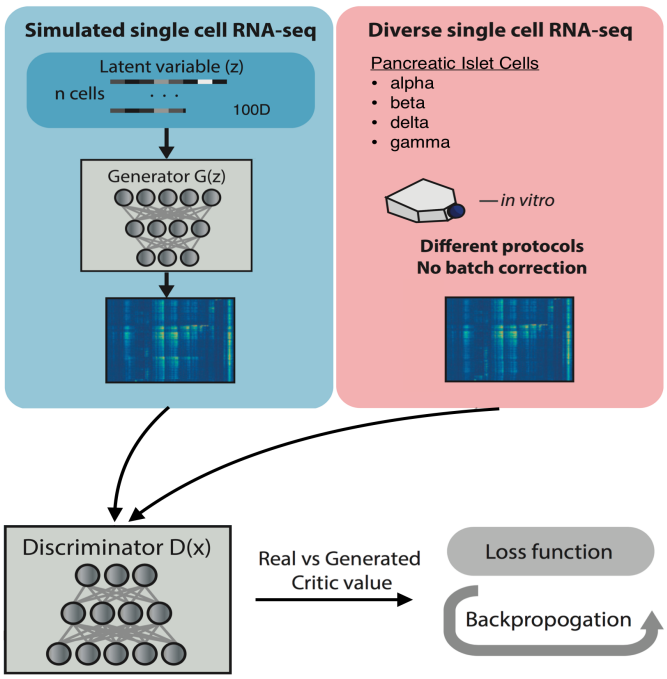

Figure 2: Schematic view of a GAN adapted from [2]

In the process of training, the generator improves the authenticity of the generated vectors and the discriminator gets better at extracting robust features from real and fake vectors that aid their differentiation. In the process,

a study [2] claims that the lower-dimensional features learnt by the pre-output hidden layer of the discriminator may help cluster same-cell types across datasets together, thus overcoming the dataset-based batch and protocol-differences mentioned earlier.

From here on, let the real and fake distributions be p_r and p_f respectively. Let D_{KL} and D_{JS} represent the Kullback–Leibler (KL) and Jensen–Shannon (JS) distributions respectively.

2.1.2 Vanilla GAN Loss

The KL divergence measures the distance between 2 probability distributions (in our case p_r and p_f).

$$D_{KL}(p_r||p_f) = \int_x p_r(x) \log \frac{p_r(x)}{p_f(x)} dx$$

The JS divergence is similar to KL although it symmetric, while KL is not.

$$D_{JS}(p_r\|p_f) = \frac{1}{2} D_{KL}(p_r\|\frac{p_r + p_f}{2}) + \frac{1}{2} D_{KL}(p_f\|\frac{p_r + p_f}{2})$$

When the discriminator is optimal (D^*), the loss of both the generator and discriminator can be combined as [4]:

$$L(G, D^*) = 2D_{JS}(p_r\|p_f) - 2\log 2$$

2.1.3 Wasserstein GAN

2.1.4 Problems with GANs

The main problems with vanilla-GANs are listed below. Note that proofs have been omitted.

1. GANs Loss Functions: GANs tries to minimize the difference (say KL or JS divergences) between the real and generated distributions. When the supports of these two distributions are disjoint, $D_{KL} = \infty$ and D_{JS} has sudden jump that is un-differentiable at certain points. Consequently it very easy to find a discriminator that classifies real from fake perfectly without necessarily going through meaningful learning. [4]
2. Vanishing Gradients: If the discriminator is perfect, i.e. $D(x) = 1 \forall x \in p_r$ and $0 \forall x \in p_f$, then the vanilla GAN loss function falls to 0 and training stops. If the discriminator cannot differentiate between real and fake then the generator lacks the feedback to generate accurate vectors. Thus learning is again stalled.

2.2 Wasserstein distance

Arjovsky *et. al* [3] proposed the usage of the Wasserstein / Earth Mover distance as the loss function to overcome the above problems of vanilla GANs. It is a smooth measure, that is continuous and differentiable everywhere, even when the supports of lower-dimensional p_r and p_f do not overlap. This allows for the discriminator to be trained well, providing improved, non-vanishing reliable gradients to train the generator. The Wasserstein Loss (W) between 2 distributions is given below:

TBD! Refer to [2]

2.3 PCA

2.4 t-SNE

A big part of our project is visualizing the reduced dimensional representations of the data produced by PCA and GANS, to check if same cell-types cluster across datasets. As mentioned in the introduction this is a problem of Manifold Learning[5]. The many algorithms in this area are based on the principle that the dimensionality of many data sets is only artificially high.[5] Consequently many Manifold Learning algorithms try to find the best lower-dimensional manifold by being sensitive to linear and non-linear structure in the data. It is also unsupervised, which helps our case where cell-labels are hard to obtain.

t-distributed Stochastic Neighbor Embedding, is a Manifold Learning algorithm, that models each point in the high-dimensional space as a 2 or 3 dimensional point. It does this in a way that similar or nearby points in

the high-dimension and represented as nearby points and far-off points in the high dimension are modeled by distant points in the lower dimension with high probability.

The distance in the higher dimension are modeled by Gaussian joint probabilities and those in the lower dimension are represented by Student's t-distribution probabilities. t-SNE learns the actual lower-dimensional positions of the points by minimizing the KL divergence of the lower-dimensional t-Student from the higher-dimensional Gaussian probability distributions.

The advantages of t-SNE are [5]:

1. It is more sensitive to local structure than other Manifold Learning algorithms
2. It identifies points belonging to different clusters

The disadvantages of t-SNE are [5]:

1. t-SNE is very computationally expensive ; it uses gradient-descent and can take hours on million-strong datasets.
2. The Barnes-Hut t-SNE algorithm can only extract 2 or 3 dimensions.

3 Methods

3.1 Code

We used Tensorflow to implement the GANS. We used Python 3.6 numpy, pandas and sklearn for PCA, t-SNE and data-preprocessing. The code repository can be found in Jupyter notebooks here.

3.2 Description of Data

We 2 obtained human pancreatic cell scRNA-Seq datasets with ids E-MTAB-5061 and GSE81608 from EMBL-EBI and NCBI Gene Expression Omnibus respectively. These have RPKM expression values.

After some pre-processing (explained in 3.3), both datasets had 20964 genes. Dataset 1 (E-MTAB-5061) and Dataset 2 (GSE81608) had 1467 and 1600 cells respectively. Dataset 1 was obtained from Illumina HiSeq 2000 and 2 from Illumina HiSeq 2500. The distributions of cell-types in both datasets were also quite similar.

Table 1: Distribution by Cell Type

Dataset	Alpha	Beta	Delta	Gamma
1	886	270	114	197
2	946	503	58	93

3.3 Data Preprocessing

These are outlaid in notebooks 1 and 2.

The main steps are explained below. Unless otherwise specified, all operations were performed on both datasets.

1. Removing all NaNs: All rows and columns with 0s only were deleted.
2. EntrezID to Gene Symbol Mapping: Dataset 2 did not have Gene Symbols. An R script was written using AnnotateDB to map all the Entrez IDs to match Gene Symbols found in dataset 2.
3. Identifying Intersection Genes: Only genes common to both datasets were included.
4. Phenotype Extraction: Each dataset had meta-information files in various formats (XML etc.), that has to be parsed to extract type and disease status of each cell.
5. Transformation: We transformed the RPKM expression values to $\log_2(RPKM + 1)$. The reason for this is that while the RPKM distribution is skewed, the log-transformed distribution is closer to the normal distribution. Also, this transformation makes the variation similar across various orders of magnitude. The images below clearly show the benefit of log2-transformation. Then '+1' inside the log is to handle 0 RPKM expression values.

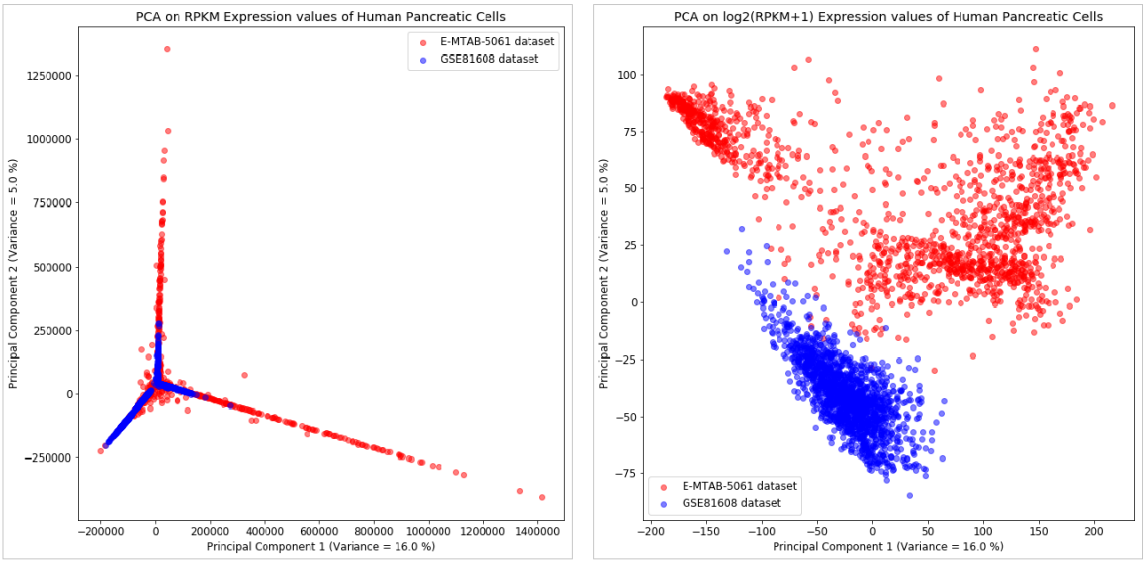

Figure 3: Left: PCA on Raw RPKM, Right: PCA on log2(RPKM + 1)

3.4 Description of Model

The wGAN we implemented can be found here. We used the same architecture as in [2].

Let bs represent batch size. Let h_i represent number of units in the i-th hidden layer. In the experiments, rows and columns represent cells and genes respectively. We trained all networks using log-2 transformed RPKM data.

Table 2: Architecture and Hyperparameters

Parameter	Generator	Discriminator
Input	(bs,100) noise	fake and real vectors (bs, 20964)
Output	(bs, 20964) fake expression vector	(32, Fake: 0, Real: 1)
Layers	2	2
h_1, h_2	600, 600	200, 200

The Adam optimizer was used, and training was done using gradient penalty as explained in [2].

3.5 Classification

4 Results

For the last decade the cost of next generation sequencing (NGS) has decreased at a rate outpacing Moore's Law [1]. As such there is an increasing amount of biological data available for large scale genomic studies. However, integration of diverse biological datasets is a notoriously challenging problem primarily due to the sensitivity of NGS platforms to variable experimental conditions and protocols. Often it is the case that comparisons between datasets from multiple sources is not possible without sophisticated batch correction techniques which are often case-specific, and thus do not generalize well out of the box [2]. Therefore there is a need for frameworks that can effectively integrate diverse datasets with minimal pre-processing, and extract biologically relevant features in the presence of batch effects.

4.1 GANS + t-SNE

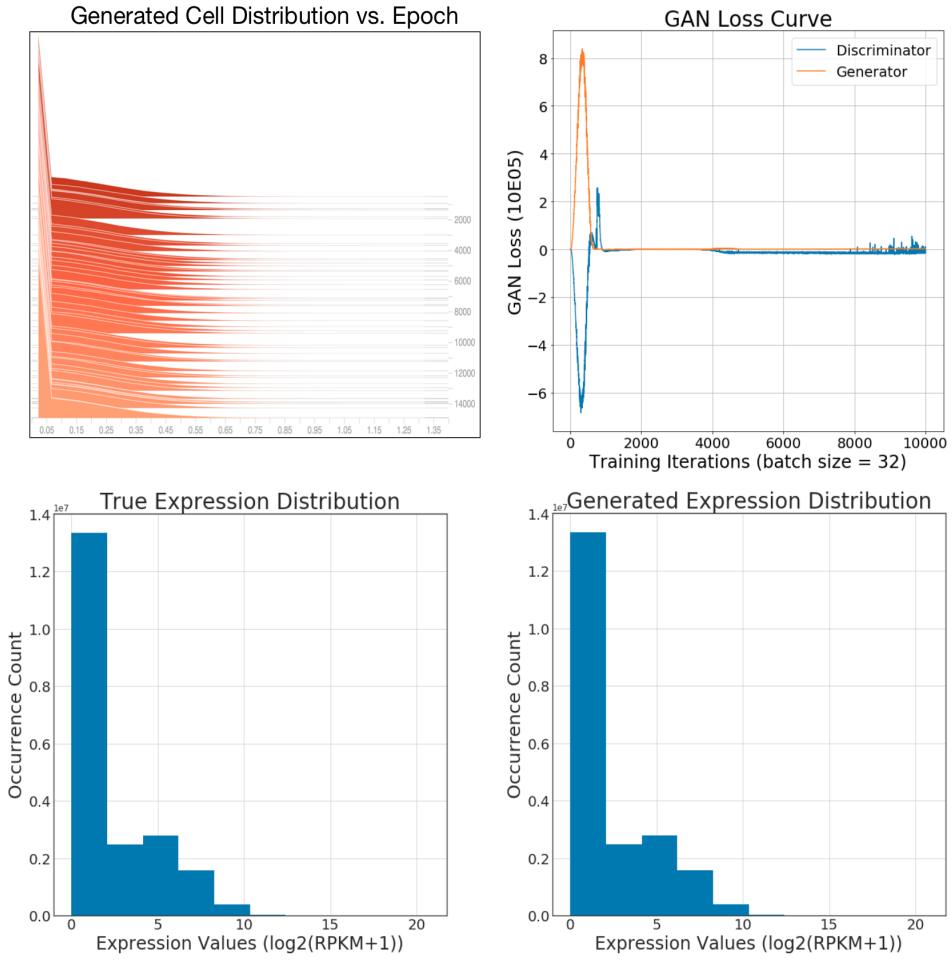

Figure 4: Model adapted from [3]. Left: The deep generative model. Middle: Greedy layer-wise pretraining in which the weights of each layer are treated as a single layer RBM. Right: The RBM's are unrolled to form a stacked autoencoder which is then further trained in a fine-tuning phase via the backpropagation algorithm.

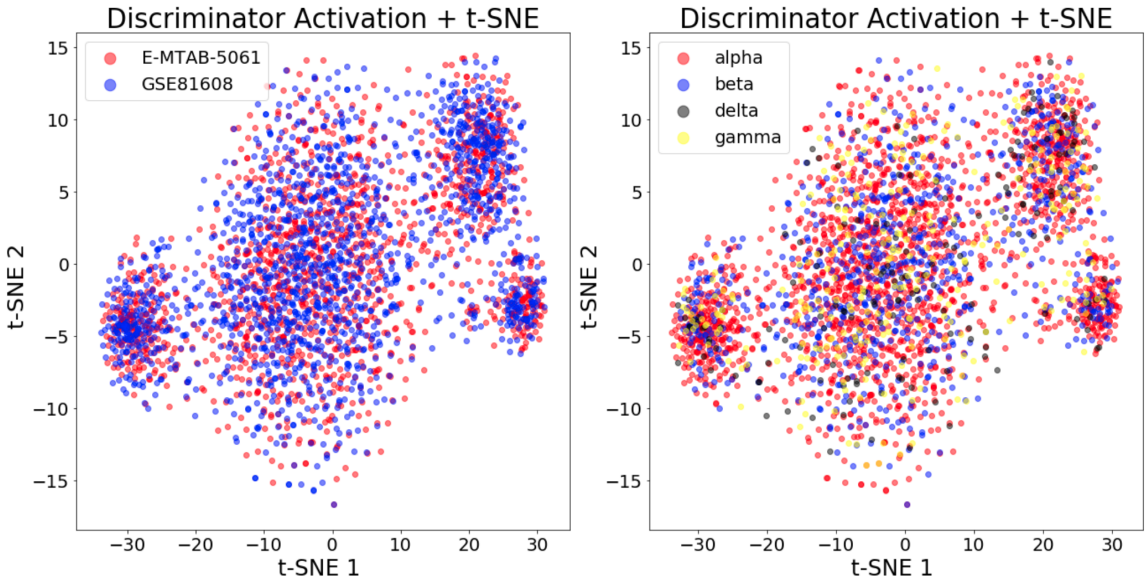

Figure 5: Model adapted from [3]. Left: The deep generative model. Middle: Greedy layer-wise pretraining in which the weights of each layer are treated as a single layer RBM. Right: The RBM's are unrolled to form a stacked autoencoder which is then further trained in a fine-tuning phase via the backpropagation algorithm.

4.2 PCA + t-SNE

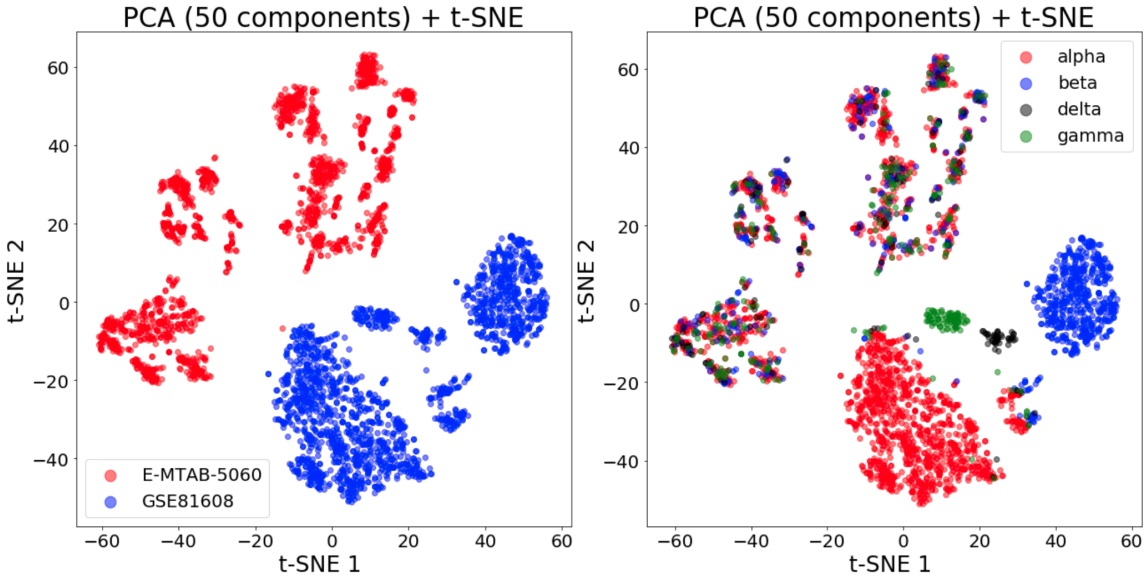

Figure 6: Model adapted from [3]. Left: The deep generative model. Middle: Greedy layer-wise pretraining in which the weights of each layer are treated as a single layer RBM. Right: The RBM's are unrolled to form a stacked autoencoder which is then further trained in a fine-tuning phase via the backpropagation algorithm.

4.3 PCA

4.4 Classification

Table 3: THIS IS THE KNN TABLE

Query	KNN	DGM
10	232.1 ± 28.8	0.18 ± 0.02
100	$1.1e3 \pm 96.4$	0.43 ± 0.02
1000	$1.3e4 \pm 150.1$	2.41 ± 0.08

5 Discussion

For the last decade the cost of next generation sequencing (NGS) has decreased at a rate outpacing Moore's Law [1]. As such there is an increasing amount of biological data available for large scale genomic studies. However, integration of diverse biological datasets is a notoriously challenging problem primarily due to the sensitivity of NGS platforms to variable experimental conditions and protocols. Often it is the case that comparisons between datasets from multiple sources is not possible without sophisticated batch correction techniques which are often case-specific, and thus do not generalize well out of the box [2]. Therefore there is a need for frameworks that can effectively integrate diverse datasets with minimal pre-processing, and extract biologically relevant features in the presence of batch effects.

6 Conclusion

In this study we assessed the performance of several unsupervised learning methods at the task of dimensionality reduction for single cell RNA-Seq data (scRNA-Seq). In addition to traditional dimensionality reduction methods such as principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE), we also implemented a relatively new framework based on Generative Adversarial Networks (GANs).

References

[1] DNA Sequencing Costs: Data Bethesda (MD): National Human Genome Research Institute; 2016. [cited 2016 Oct 1].

[2] Generative adversarial networks uncover epidermal regulators and predict single cell perturbations A Ghahramani, FM Watt, NM Luscombe - bioRxiv, 2018.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

[4] `https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html`

[5] `http://scikit-learn.org/stable/modules/manifold.html`