Правительство Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

Основная образовательная программа

Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему

“Условные генеративные конкурирующие сети для синтеза биологических изображений”

Выполнил студент группы БПМИ 141, 4 курса,

Скрипнюк Владислав Олегович

Научный руководитель:

к. ф.-м. н., Осокин Антон Александрович

Москва 2018

Contents

Abstract

Generative Adversarial Networks formulate the task of learning probability distributions as a game theoretic problem. They have been especially popular for generating images due to their ability to generate sharp, plausibly looking samples. GANs already can successfully generate handwritten digits, human faces and indoor scenes, but this list is growing steadily. Osokin et.al. [6] have shown that GANs can be fruitfully applied to biological images, in order to model mutual spatial arrangement of proteins in a cell. This is a task of immediate practical interest, since current biological imaging techniques have certain limitations, and machine learning can, in principle, help to overcome these constraints. In this work, we modify the conditioning technique of Mirza et.al. [20] and construct a GAN-based architecture that is able to jointly generate images of 6 proteins in connection with one another. We also show, that projection discriminator technique, which was recently proposed by Miyato et.al. [21], enables generation of 41 proteins. In addition to that, we explore properties of several GAN architectures and conditioning methods using a number of evaluation metrics on a synthetic data and some conventional datasets, like MNIST and CIFAR-10.

Keywords: Generative Adversarial Networks, Conditional Generative Adversarial Networks, synthesis of biological images.

Генеративные конкурирующие сети формулируют задачу восстановления вероятностных распределений как задачу теории игр. Эти модели получили особенно широкое распространение для генерации изображений, благодаря их способности генерировать четкие и естественно выглядящие картинки. Генеративные конкурирующие сети успешно применялись для моделирования рукописных цифр, лиц людей и интерьеров, но этот список постоянно растет. Осокин и др. [6] продемонстрировали, что генеративные конкурирующие сети могут принести пользу при применении к биологическим изображениям для моделирования взаимного расположения белков в биологических клетках. Эта задача представляет практический интерес, потому что современные способы получения изображений клеток имеют определенные ограничения, и машинное обучение может помочь их преодолеть. В этой работе мы предлагаем модификацию процедуры условной генерации, предложенной Мирза и др. [20] и конструируем GAN-подобную архитектуру, которая позволяет одновременно генерировать изображения 6 белков совместно. Также было показано, что предложенный Миято и др. [21] дискриминатор с проекцией позволяет успешно генерировать фотографии 41 белка. Кроме того, мы проводим исследование свойств разных моделей генеративных конкурирующих сетей с помощью нескольких метрик на искуственных данных и известных датасетах, таких как MNIST и CIFAR-10.

Ключевые слова: Генеративные Конкурирующие сети, Условные Генеративные Конкурирующие сети, синтезирование биологических изображений.

1 Introduction

For several decades generative models have been an area of active research. Generative modeling can be beneficial in several aspects. Learning high-dimensional distributions usually requires understanding of hidden structure of data, and numerous approaches to generative modeling were designed to learn such structure. Basically, all models with hidden variables learn such structure in some way. Another appealing application, which is more relevant to this work, is that generative modeling can potentially be used to simulate objects, which can not be observed in experiments. Especially impressive results have been achieved by models built with deep neural networks applied to speech, text and images. Several successful models have been developed for images, including RBM [9], VAE [14], PixelCNN [5] and GANs [8]. The latter have shown an ability to generate better looking images on various benchmarks, like MNIST, CelebA and LSUN. An especially remarkable result was achieved by Karras et.al. [24], as they generated absolutely plausibly looking portraits in 1 megapixel resolution.

In our work we follow research direction, which was set in [6] and use generative adversarial networks to model mutual spatial arrangement of several fluorescent proteins in fision yeast cells. Fluorescent proteins can be photographed with help of fluorescent microscopy. There are dozens of proteins of interest in a cell, but the main limitation of fluorescent microscopy as an imaging technique is that it can only capture small subset of all proteins simultaneously. To tackle this drawback we trained GANs to learn correlation patterns between different proteins. Hereby, we expect, that the trained model will be able to generate all observed proteins simultaneously. Thus, it can be a useful tool for biologists, as it can reveal connections between proteins, which were not observed before. These findings are supposed to be useful to propose novel hypothesis about processes in the cell, which can be later tested in experiments on biological materials.

This work considers several adversarial models, namely, GAN, Wasserstein GAN, Least Squares GAN, Spectral Normalization GAN and conditioning methods: conditional GAN, multichannel GAN and GAN with projection discriminator. First, we test them on a synthetic dataset of 2d points and explore properties of each model with help of several evaluation techniques. Then we experiment with image datasets, namely MNIST, CIFAR and a dataset of biological images LIN. We propose a modification of the star-shaped model [6] to generate images of 6 proteins. We are also able to generate 41 proteins with help of projection discriminator technique [21]. Performance of projection discriminator motivated us to try it for the task of text to image synthesis, and we report results of our experiments.

The thesis is organized as follows. Section 2 makes a brief overview of existing work, which is relevant to our research. Section 3 presents a thorough explanation of all GAN models, which we used in our experiments and some existing evaluation metrics for generative models. Section 4 documents our experiments on synthetic data, MNIST, LIN and Caltech-UCSD Birds datasets and describes all our results. Then, our work comes to an end with conclusion section, which briefly summarizes our findings.

2 Related Work

The framework of GANs was proposed by Goodfellow et.al. [8]. This paper attracted a lot of attention and gave birth to a new research direction. Generative Adversarial Networks are commonly applied to continuous objects, especially images. Their distinctive feature comparing to other approaches is the ability to generate sharp images. However, GANs have several deficiencies, like mode collapse and unstable training, and many efforts were made to mitigate those issues. Radford et.al. [26] proposed network architecture, which works well on many image datasets with low spatial resolution and is now considered to be a standard architecture for GANs. Several attempts were made to improve GAN training through refinement of training objective, for instance Arjovsky et.al. [2], Gulragani et.al. [12] and Mao et.al. [17] modified loss function via minimization of different distance between probability distributions. Nevertheless, some researchers claim that recent advances are arguable [1], as thorough large-scale investigation did not identify any significant differences between them, and that generative models should be evaluated more carefully [30].

Generative Adversarial Networks have also been applied to modeling conditional distributions. A range of works explored conditioning on a discrete variable, corresponding to a class label [20], [23], [21]. Other researchers have also used more complex conditions, like generating images based on images [10], [31] and images based on text [29], [28] and [13].

A proper evaluation of generative models is still an open problem, and several metrics were designed in recent years. Salimans et.al. [11] and Heusel et.al. [7] suggested evaluation methods for real world images, based on features extracted with help of classifiers, which are pretrained on large-scale datasets, like ImageNet. In our work we mostly use the Classification Two Sample Test (C2ST) method proposed by Oquab et.al. [19], which makes use of classifiers to distinguish data and model distributions. Jiwoong et.al. [25] uses similar principle to measure differences between

two distributions, though they suggest to use discriminator to approximate divergence between two distributions, as it is done during training [22].

3 Methods

3.1 GANs

The framework of Generative Adversarial Networks was formulated in [8]. This framework formulates the task of learning probability distributions as a minimax two-player game between two neural networks: generator and discriminator. This process is illustrated on a scheme below:

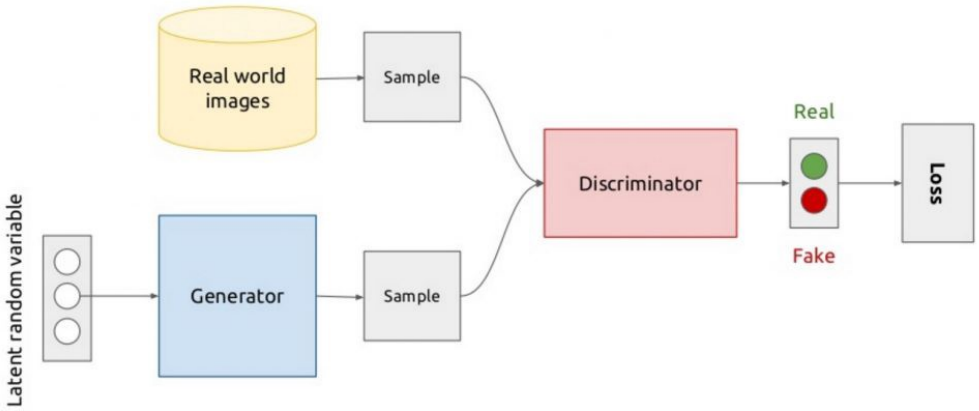


Figure 1: Scheme of Generative Adversarial Network

The generator is used to approximate the data distribution by transforming some simple distribution, for instance, multivariate normal distribution. Discriminator is used to find the differences between the original and the approximation distributions, so that the generator can modify itself to fit the distribution of interest better.

Mathematically, this problem is formulated as follows:

$$J(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{data}} \log D(x, \theta_D) + \mathbb{E}_{\epsilon \sim p_{noise}} \log(1 - D(G(\epsilon, \theta_G), \theta_D))$$

discriminator's task:

$$\max_{\theta_D} J(\theta_G, \theta_G)$$

generator's task:

$$\min_{\theta_G} J(\theta_G, \theta_G)$$

However, many modifications of this scheme were suggested later.

3.2 Least Squares GAN

The previous method is prone to the problem of vanishing gradients, when the discriminator classifies samples into real and fake too accurately. The prevents generator from effective learning, and [17] suggested a modification of the previous objective function, which is supposed to relief this problem.

$$J(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{data}}(D(x, \theta_D) - 1)^2 + \mathbb{E}_{\epsilon \sim p_{noise}}(D(G(\epsilon, \theta_G), \theta_D) + 1)^2$$

This objective prevents the discriminator from being too confident, by forcing the margin $y_i D(x_i)$ to be closer to 1 (it is assumed, that $y_{real} = 1$, $y_{fake} = -1$).

3.3 Wasserstein GAN

This method is based on optimization of an advanced divergence between distributions, namely Earth-Mover distance and Wasserstein-1 distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} ||x - y||$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions $\gamma(x, y)$, such that the marginals $\gamma(x)$ and $\gamma(y)$ are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

It can be shown, that the above divergence is equal to:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||D||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} D(x) - \mathbb{E}_{x \sim \mathbb{P}_g} D(x)$$

This formulation resembles the objective functions of GAN and can therefore be optimized in the same way. However, constrained optimization appears to be unstable, and [12] have suggested practical relaxation.

$$J(\theta_G, \theta_D) = \inf_D \left[\mathbb{E}_{x \sim \mathbb{P}_g} D(x) - \mathbb{E}_{x \sim \mathbb{P}_r} D(x) + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} (||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2 \right]$$

However, this formulation requires more resource-intensive computations, since one need to make two backward passes to make 1 optimization step.

3.4 Spectral Normalization GAN

Spectral Normalization GAN was first proposed in [27]. The main motivation behind this method is that it is important for the discriminator to be a Lipschitz function, because it provides the generator with gradients for learning. To demonstrate, that unconstrained gradients can damage the learning process, authors consider the optimal discriminator:

$$D_G^*(x) = \frac{q_{data}(x)}{q_{data}(x) + p_G(x)} = \text{sigmoid}(f^*(x))$$

where $f^*(x) = \log q_{data}(x) - \log p_G(x)$

$$\nabla_x f^*(x) = \frac{1}{q_{data}(x)} \nabla_x q_{data}(x) - \frac{1}{p_G(x)} \nabla_x p_G(x)$$

it is clear, that this quantity can be arbitrarily big and lead to unstable training of the generator.

To overcome this issue, it is suggested to set Lipschitz constant of each layer to be equal to 1. Then the whole network will be a Lipschitz function as a composition of Lipschitz functions. Discriminators for images are usually composed of convolutional layers. Convolution is a linear mapping, and it has matrix of size $n_{out} \times n_{in} HW$. To set the Lipschitz constant of a linear mapping to be equal to 1, it is enough to divide it's matrix by a spectral norm of this matrix.

3.5 Conditioning for GANs

The first method for conditioning on class label was proposed almost instantly after invention of GANs. Such methods can be useful in several ways, for instance conditioning on class labels introduces some additional knowledge about the structure of data.

3.5.1 Conditional GAN

This method was proposed by [20] after the original GAN paper and since then was considered to be a standard method of conditioning for GANs. The scheme of the proposed method can be seen on the picture below.

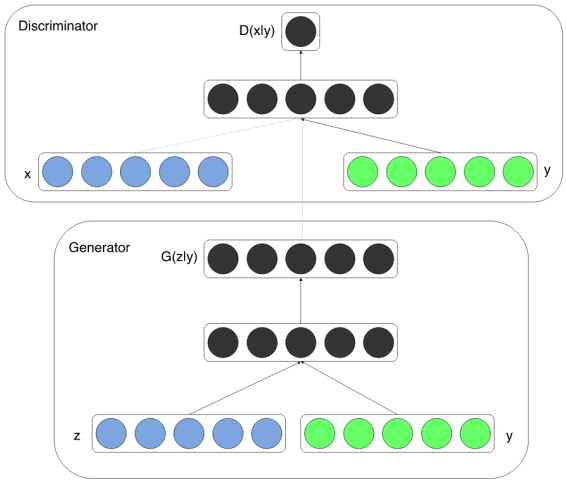
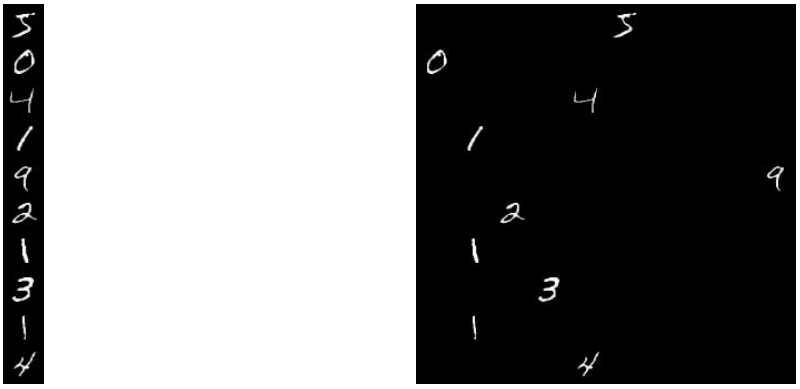


Figure 2: Scheme of conditional GAN.

Class label is passed to generator and discriminator as an additional argument. This makes the discriminator learn relations between class labels and appearance of specified classes, because images with distinctive features of some class are always passed in conjunction with the same class label. Therefore generator has to replicate these relations to fool the discriminator.

3.5.2 Multichannel GAN

We have created this method as a modification of the previous one. Previous method implies, that discriminator will learn differences in appearance between different classes. We wanted to make the task simpler for the discriminator, by spatially separating images of different classes into different channels. This process is illustrated below:



MNIST samples before spatial separation. MNIST samples after spatial separation.

Figure 3: Visualization of the spatial separation process.

We visualized spatial separation for 10 samples from MNIST. Each row corresponds to independent sample on both figures. Images before separation were white and black and hence the had only one channel. Images after separation have 10 channels, with one-to-one correspondence between channels and classes. So, the right image has 10 columns, and channels of one image are arranged in a row. Therefore, discriminator does not have to distinguish images of different classes, since they are located in different channels.

However, real images after separation have only one non-zero channels, while generator generates all channels simultaneously. Hence, it is necessary to mask out all channels except one randomly chosen channel in the generator during training phase.

3.5.3 GAN with projection discriminator

This method was proposed in [21] as an improvement of conditional GAN of [20], which was also used to learn conditional distribution of images, conditioned on class label. The main difference between these two methods is a way, in which class label is passed into discriminator. The scheme of projection discriminator is presented on the picture below.

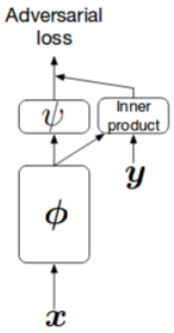


Figure 4: Scheme of projection discriminator.

The general idea of the method is that images and class labels are not homogeneous and therefore should not be concatenated. In opposite, one could follow the example of convolutional networks, designed for classification. Convolutional networks usually have a fully connected layer on the top, and each class has a dedicated row of weights in the weight matrix of this layer. So it is suggested

to have a fully connected layer on the top of discriminator, and the weights of this layer should be chosen with respect to the class label.

That being said, first, the network extracts some representation from the image with help of layers ϕ. Then there is fork in network architecture. One branch is a conventional linear layer ψ, which has the same set of weights for all classes. The other branch is also a linear layer, though it chooses weights based on the class label. The outputs of these two branches are averaged, and passed to the output of the discriminator.

3.6 Evaluation metrics

3.6.1 Likelihood estimation using KDE

It is rather popular to use likelihood as metric to evaluate generative models. Unfortunately, GANs have intractable likelihood, as it is defined implicitly $p(x) = \int p(x|z)p(z)dz$. Nevertheless, sampling is rather simple for GANs, hence, Kernel Density Estimation (KDE) methods can be used to estimate likelihood. The general idea of those methods is simple: we need to get rather big sample and assign some probability mass to each sampled point with help of some kernel function. However, those methods have high variance and depend heavily on the sample size and hyperparameters of the kernel function.

3.6.2 Inception Score

Inception Score was first introduced [11] to evaluate images, which were learned from datasets like ImageNet. This metric makes use of a classifier, trained on large-scale dataset of images, which are similar to the domain of interest. Let us denote $p(x)$ to be the learned distribution, and $p(y|x)$ to be the distribution, returned by the classifier. Then, Inception Score is based on two assumptions:

1. Meaningful images should be recongnizable by a trained classifier, therefore label distribution $p(y|x)$ should have low entropy.

2. We expect the generator to learn varied images, so the marginal distribution $p(y) = \int p(y|x = G(z))dz$ should have high entropy.

If we add these two quantites with equal weights and appropriate signs, we get:

$$\int\int p(y|x)\log p(y|x)p(x)dydx - \int p(y)\log p(y)dy =$$
$$= \int\int p(y|x)\log p(y|x)p(x)dydx - \int\int p(y|x)\log p(y)p(x)dxdy =$$
$$= \mathbb{E}_x D_{KL}(p(y|x)||p(y))$$

To make it more handy, authors suggest to exponentiate this metrics, so:

$$\text{Inception Score} = \exp\left(\mathbb{E}_x D_{KL}(p(y|x)||p(y))\right)$$

3.6.3 Frechet Inception Distance (FID)

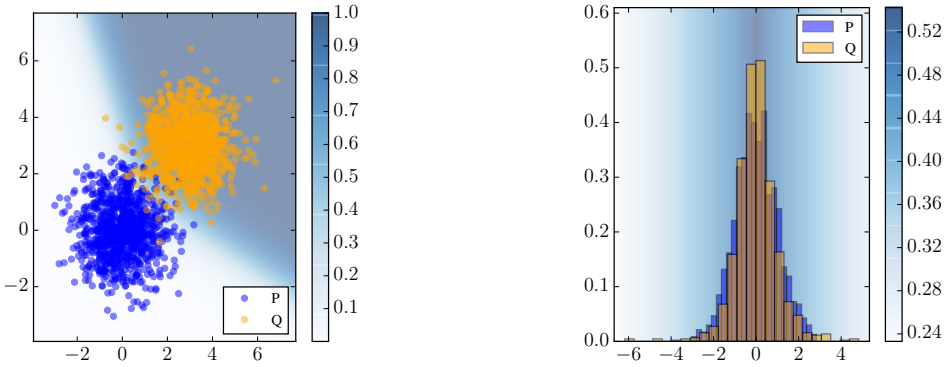
This evaluation metric also makes use of a pretrained classifier. It computes certain statistics of real and approximation distributions with help of a pretrained classifier. To be more specific, it measures how differently are distributed activations of the last layer of the network, when it receives real and generated data.

To compute this distance, one has to compute mean m and covariance matrix C of activations for real and fake data and then compute the distance as follows:

$$d^2\left((m_r, C_r), (m_f, C_f)\right) = ||m_r - m_f||_2^2 + \mathrm{Tr}\left(C_r + C_f - 2(C_r C_f)^{1/2}\right)$$

3.6.4 Classifier Two Sample Test

The idea, proposed in [19] is very simple: absolutely identical distributions can not be distinguished with a classifier. Therefore, it is suggested to train a model to accurately classify samples from two distributions, and see, whether the classifier succeded or not. The more accurate the classifier is, the less precise is the approximation distribution. An example of two poorly matched distributions and almost perfect classifier is provided below.



Poorly matching distributions, accurate classifier. Perfectly matching distributions, poor classifier.

Figure 5: Visualization of a classifier, used to find differences between distributions.

4 Experiments

4.1 Mixture of Gaussians

To get an idea of performance of Generative Adversarial Networks, we have trained a number of models on a two-dimensional data, which forms mixture of 25 gaussians as described at [12]. We experimented with vanilla GAN [8], WGAN-GP [12] and LSGAN [17] as well as with their conditional versions. We introduced conditioning in a similar manner to [20]. We followed [12] and chose discriminator and generator to be an MLP with 4 hidden layers, 512 neurons in each. For optimization we used Adam [15] with $\alpha = 0.5$ and $\beta = 0.999$ as it was widely used for GAN training since [26].

We trained each model for 6k iterations and plotted samples and kernel density estimation of the model's density. Samples and KDE estimates for trained models can be seen in appendix A.1.

It is clear that GAN suffers from mode collapse i.e. model distribution covers data only partially. LSGAN is less prone to aforementioned problem, however it also drops some modes. WGAN-GP has very blurry distribution which covers all data points. We decided to train WGAN-GP for 100k iterations to reproduce results from [12]. This resulted in much sharper modes without a hint of mode collapse. Increased training time did not help to GAN and LSGAN as it only made mode collapse more severe. Samples and KDE estimates for trained models can be seen in appendix A.1.

We have also used KDE to estimate log-likelihood of training data under model distribution. However, those estimates have high variance and should be treated with caution, as suggested by [30].

Model	Log-likelihood
GAN	-2638.92 ± 156.12
LSGAN	-2309.23 ± 192.84
WGAN-GP	-2295.63 ± 130.31
WGAN-GP 100k	-1658.55 ± 68.32

Table 1: KDE estimation of real data log-likelihood

Only vanilla GAN has significantly lower log-likelihood, since it assigns extremely low probability to some data points due to mode collapse.

We have also tested Classifier Two Sample Test for evaluation of generative models.

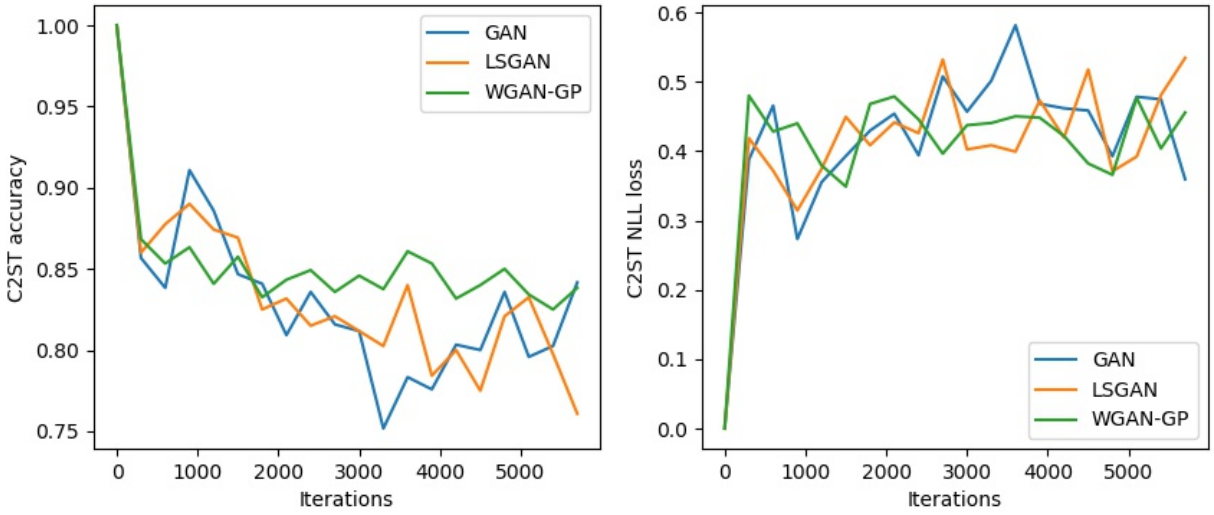


Figure 6: C2ST for 3 types of GANs

We observed, that C2ST metric values have monotonous trend in the course of training, nevertheless they are highly unstable and did not give consistent results for comparing models in this experiment.

Experiments with conditional models have shown, that conditional GANs capture differences between classes well, while demonstrating the same behaviour inside classes as unconditional models. Samples and KDE estimates for trained models can be seen in appendix A.2.

4.2 MNIST

4.2.1 Unconditional setting

To validate our models we have experimented with MNIST dataset of handwritten digits [18]. This dataset is widely used in machine learning in general and especially for training of generative models. We achieved results comparable to those, which were previously published in other

works and conducted some additional experiments to evaluate conditional training of GANs and test Classifier Two Sample Test.

We trained GAN, Least Squares GAN and Wasserstein GAN with Gradient Penalty on MNIST. We demonstrate samples generated from trained models on a figure below.

GAN LSGAN WGAN-GP

Figure 7: Samples from models trained on MNIST

All samples look plausible with minor defects, however it is difficult to compare models by visual inspection of samples. To that end we exploit Classifier Two Sample Test to compare these models quantitatively. This metric has shown that WGAN-GP is better than two other models, since it is more difficult to discriminate between samples from WGAN-GP and real data. However, C2ST metric demonstrates unstable behaviour and we try to alleviate this problem by computing C2ST 10 times. Error bars were constructed using 10%, 50% and 90% quantiles of computed values.

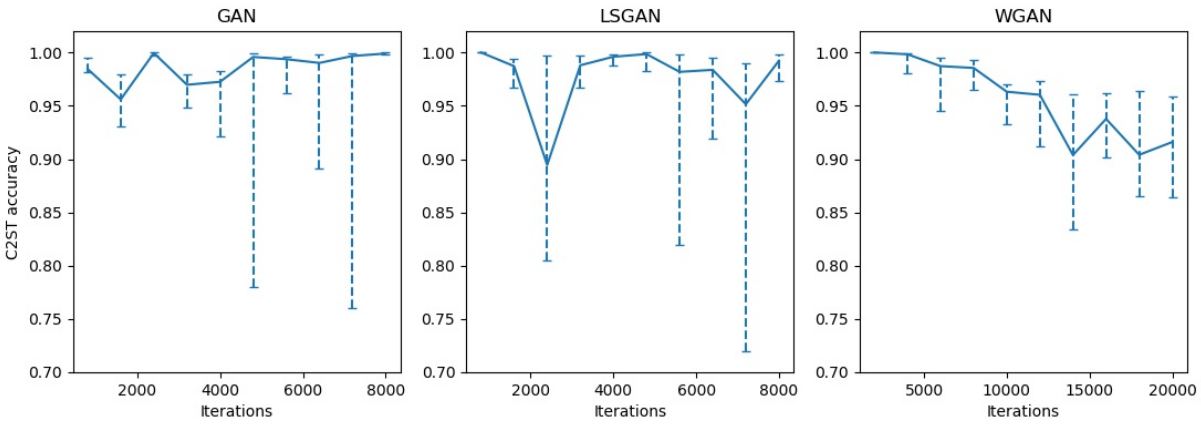


Figure 8: Quantitative evaluation of GAN training methods using C2ST. Lower values are better.

4.2.2 Conditional setting

The MNIST dataset [18] contains images of handwritten digits of 10 classes. We trained class aware models on MNIST in two ways. First, we trained conditional GAN, as suggested in [20], to model all 10 classes simultaneously. Second, we have divided MNIST into 10 datasets, each dataset containing images of one class, and trained 10 individual GANs, one model for each class.

It was particularly interesting to check if conditional GAN learns some classes worse than distinct models, fitted for each class separately. [3] have shown, that a distribution, originating from a trained GAN model has a finite and rather small support. Therefore it was natural to hypothesise that it would be more difficult for a conditional model to remember all classes, as opposed to distinct models, since each of them has to remember only one class.

However, training of separate models on classes of MNIST appeared to be very unstable and shown very poor results. Interestingly, drop in performance happens at some point during training, i.e. samples start fading or acquire various artefacts, though training up to this point seemed to be ordinary. This phenomenon also appeared in implementation of DCGAN from official PyTorch [4] repo, so we guess, that such behaviour may be caused by some peculiarities of MNIST dataset.

We demonstrate samples, produced by trained models below. Conditional GAN learned all classes well, while samples from one-class models have obvious defects.

conditional GAN

one-class GANs

Figure 9: Samples from models trained on MNIST

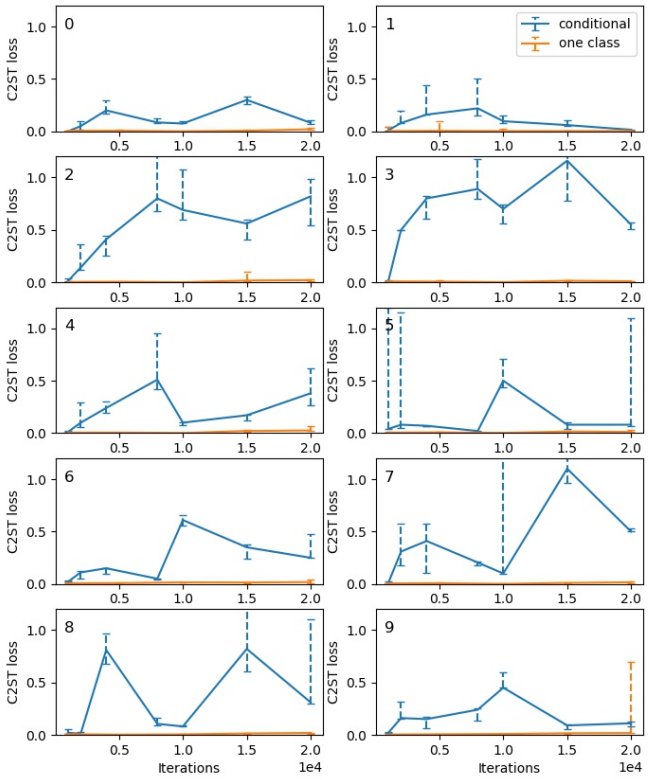


Figure 10: C2ST for conditional and one-class models GANs. Greater values are better.

In addition to that, those models can also be clearly distinguished using Classifier Two Sample Test. We wanted to compare conditional model and one-class model for each class separately, but the results are almost the same for all digits. Samples generated by one-class models are almost perfectly distinguishable from the real ones.

Lopez-Paz and Oquab [19] have shown, that Deep Neural Networks can distinguish real data from samples generated by the model almost perfectly, so they should be somehow weakened to provide more demonstrative metric. In our experiments we use rather expressive model, so we weaken it in a different way. We train discriminator only for 300 iterations. The following figures show, that this is enough for a discriminator to converge, while it is clear, that it becomes more and more difficult for a discriminator to identify samples generated by the model, as generator becomes better in the course of training.

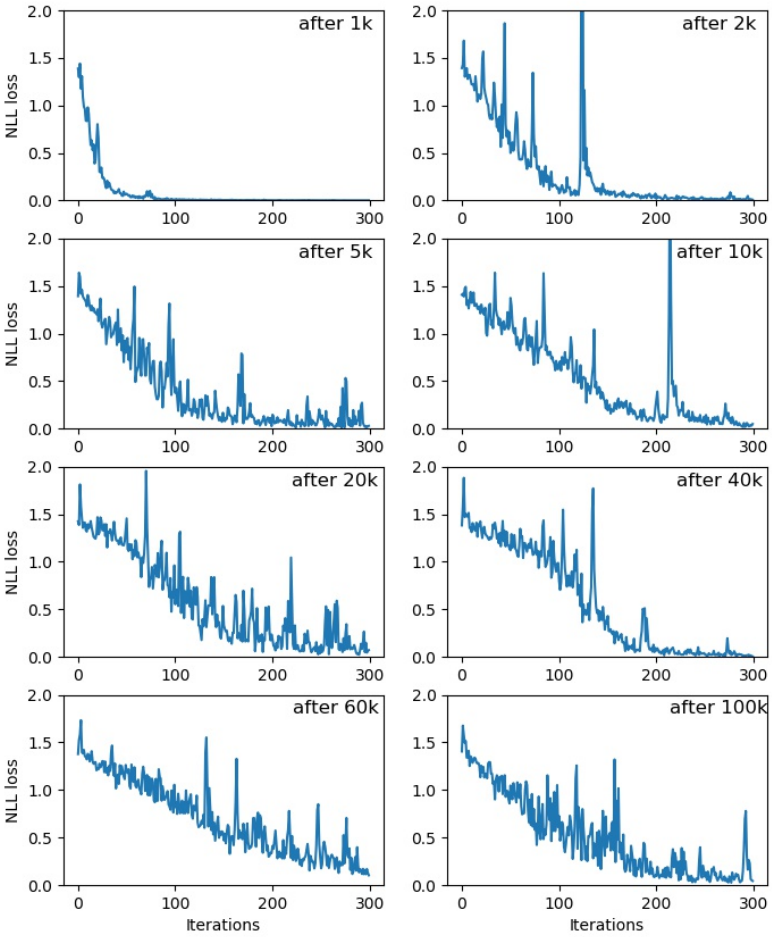


Figure 11: Convergence of C2ST discriminators, used for evaluation of GAN trained for 100k iterations.

However, significant oscillations happen even at the end of training. To compensate these instabilities we run discriminator training 10 times and report quantile values.

4.3 LIN

In our work we trained Generative Adversarial Networks on LIN dataset, which images of fission yeast cells. The next section describes this dataset in more details.

4.3.1 LIN dataset

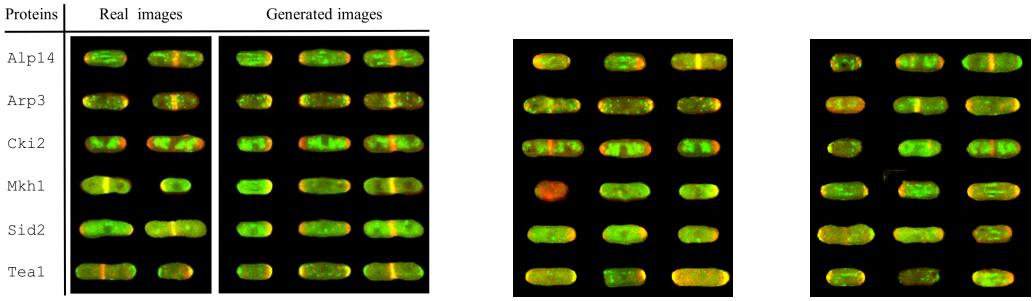
LIN dataset was fully described in [6]. For the sake of comprehensibility and integrity we will provide the reader with necessary background in the following two paragraphs.

Fluorescent microscopy Fluorescent microscopy is a powerful method, which lends itself well to monitoring biological processess in living cells and organisms. It can be used to locate proteins of interest in a cell with help of fluorescent compounds. Fluorescent compounds are attached to the proteins of interest with help of genetic engineering. Then they can be easily traced, as they emit light at a specified wavelength (the emission spectrum) when a light at some other wavelength (the absorption spectrum) is shed on them. Multiple fluorescent compounds with different absorption and emission spectra can be used to image several proteins of interest in a cell and reveal patterns in their mutual arrangement. However, aforementioned approach is only applicable to photograph 3-4 proteins at a time, due to collisions in emission and absorption spectra of different fluorescent compounds. It should be noted, that colors in resulting images do not have any physical meaning, as they are defined by auxiliary fluorescent compounds and do not depend on the properties of the proteins of interest. The only things that matter are the intensity of the light and co-localization of different colors.

LIN dataset LIN dataset contains images of fission yeast cells, which were obtained with help of fluorescent microscopy. A detailed description of fission yeast (Schizosaccharomyces pombe) cells is beyond the scope of this writing. LIN dataset contains 170000 images in spatial resolution of 48×80 Each image has 2 color channels, which we consider as a red and green channel for coherence with the previous work [6]. Red channel of each image depicts a Bgs4 protein, which is known to be highly correlated with the growing cycle of a cell. Green channel of each image corresponds to one of other 41 proteins. Thus, we are going to use "red" protein as a mediator to capture dependencies between other 41 proteins. For simplicity, we divide all images into 41 classes in accordance to the protein in the green channel.

4.3.2 Baselines

In this section we follow [6] and consider a small subset of the LIN dataset, containing images of only 6 "green" proteins. On the leftmost picture below one can see real photographs of fission yeast cells.



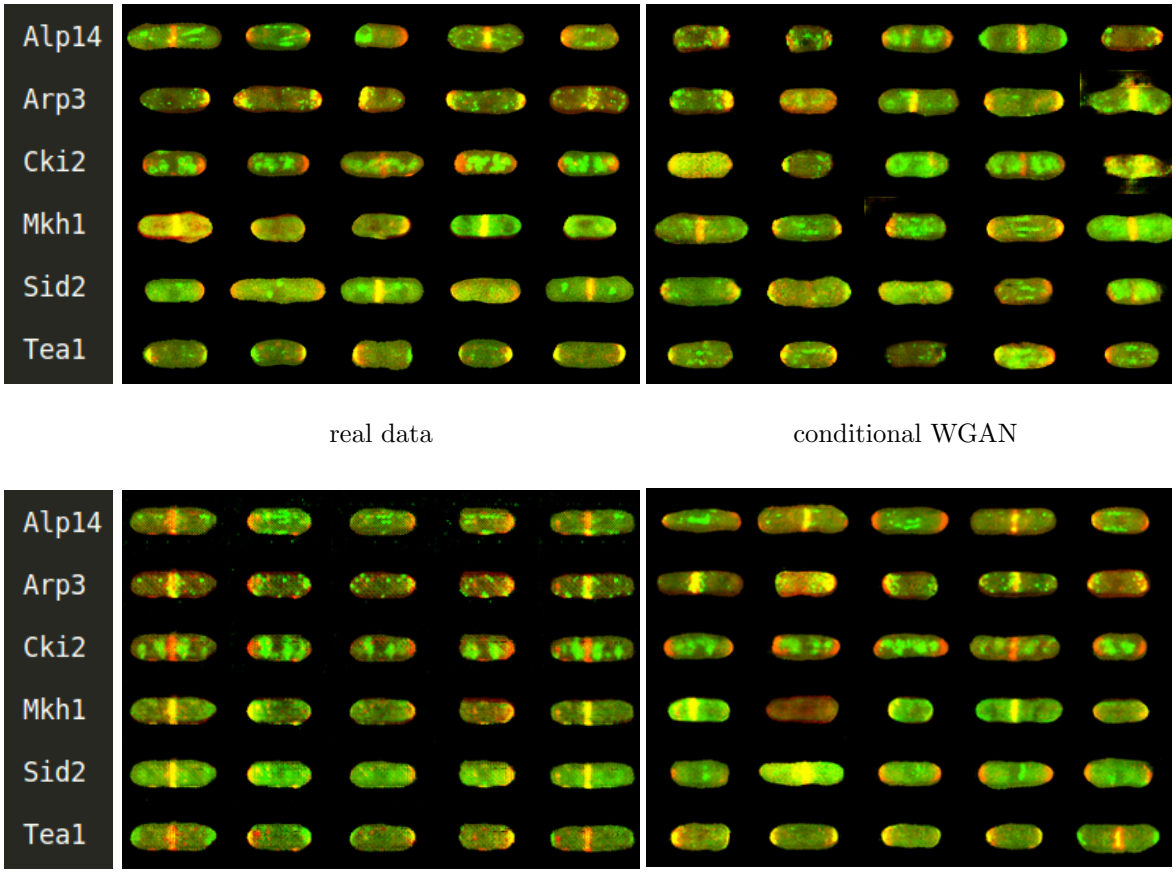
real data and star-shaped model [6] one-class WGANs conditional WGANs

Figure 12: Samples from baselines trained on 6 classes of LIN

One-class WGANs As a first step we trained 6 separate WGAN models for each class. Trained models have learned distinctive features of each class. That means, that images of such nature can be successfully learned with Generative Adversarial Networks.

Conditional WGAN We have also trained conditional GAN in a way, suggested in [20]. However, conditional model has shown poor performance. First, training process was unstable, and sometimes trained model generated implausible samples. Second, conditional model failed to learn class specific features, and generator ignored class labels, while generating, in principle, satisfactory samples. That can be seen on extended pictures below.

Star-shaped model To overcome aforementioned shortcoming of conditional model, [6] have designed a star-shaped model. This model is basically a combination of two previous models, as it used one common network to generate "red" channel, while keeping separate networks, which take all intermediate tensors from "red" network as an input, to generate all "green" proteins. This model combined strong features of one-class models and conditional model, as it learned appearance of each class well, while preserving ability to generate all classes in connection with each other.



real data conditional WGAN

multichannel GAN GAN with projection discriminator

Figure 13: Visual comparison of models trained on LIN.

4.3.3 Multichannel GAN

In our work we applied two novel GANs to model co-localization patterns between various proteins in a cell. First, we trained Multichannel GAN on 6 classes of LIN. Samples from the trained model can be seen on a picture below. Produced samples reflect main class-specific features, while some of them have minor artifacts, for instance checkerboard patterns, which can be caused by a specific architecture of the generator. It should also be noted, that multichannel architecture lends itself well to generation of all proteins in connection with each other. It can be clearly seen on the picture below, that samples in one column have the same shape.

4.3.4 GAN with Projection Discriminator

Another novel model, which we trained on LIN is a GAN with projection discriminator. Samples, generated by this model, are not visually distinguishable from real data. We have also trained this

model on full LIN dataset with 41 classes. Visual comparison of real and generated samples of 41 classes can be found in appendix. None of the previous conditional models were able to successfully generate 41 proteins.

4.3.5 Quantitative evaluation

To measure our results on LIN quantitatively, we have incorporated FID metric. FID usually uses the same version of a pretrained Inception network trained on ImageNet dataset. However, features learned on ImageNet are not relevant to our task, therefore, we have trained our own network to distinguish 41 classes of LIN, which achieved 81% accuracy on validation set. By that we have changed conventional procedure, which is used to compute FID. So, we have done a number of sanity checks, to make sure that FID still adequately measures visual similarity between distributions of images.

As a first step, we have measured FID between different classes of LIN. The matrix of pairwise similarities can be found in appendix D. To measure FID we sampled 500 photos from training part and 500 photos from test part of the dataset. To show consistency of FID metric, we have visualized 40 entries from the pairwise distances matrix on a figure below, 20 entries correspond to intra-class and 20 entries to inter-class similarities. It is clear, that intra-class similarities are much lower than their inter-class counterparts. This trend is also can clearly be seen on a pairwise similarity matrix.

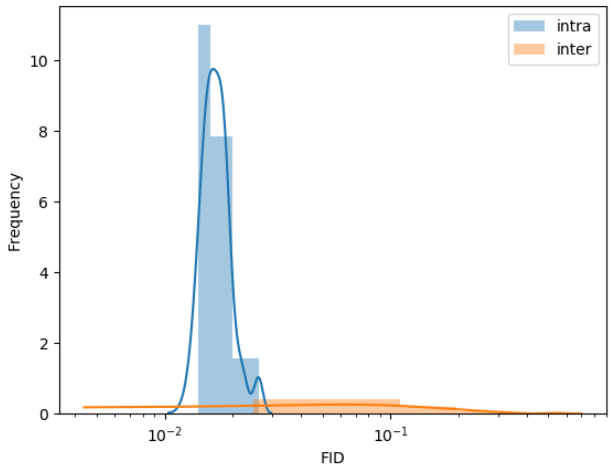
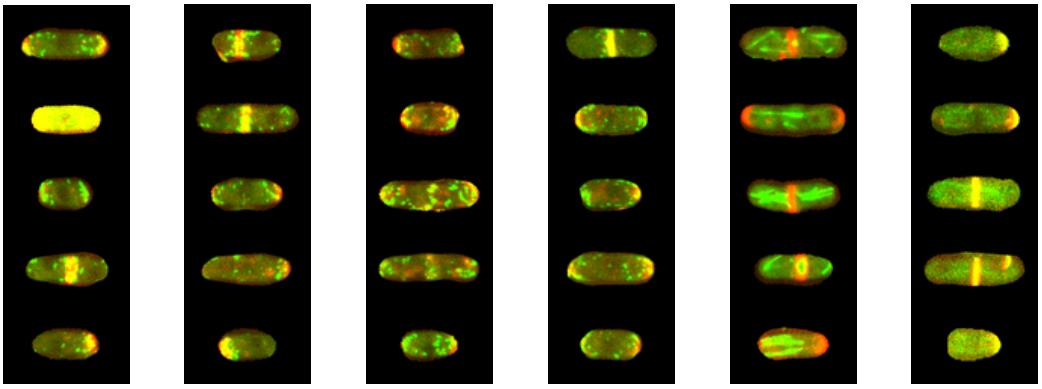


Figure 14: Intra-class and inter-class similarities.

Some off-diagonal elements of pairwise similarity matrix are as small as diagonal ones. To show, that those elements correspond to visually similar proteins, we have visualized nearest neighbours of Arp3 protein.



Arp3 0.020 (Arp3) 0.027 (Fim1) 0.031 (Myo1) 0.045 (Atb2) 0.047 (Shk1)

Figure 15: Nearest neighbours of Arp3.

It is obvious, that low values of FID correspond to high visual similarity.

Now, after we were convinced that FID is sensible, we used it to compare several conditional models.

Protein	Real data	cGAN	multiGAN	projection
Alp14	0.018	0.940	0.270	0.028
Arp3	0.020	1.334	0.300	0.034
Cki2	0.015	0.408	0.134	0.040
Mkh1	0.015	1.921	0.152	0.028
Sid2	0.015	0.468	0.272	0.027
Tea1	0.017	0.633	0.179	0.038

Table 2: Comparison of several conditional models with FID. (Full table in appendix D)

It turned out, that conditional GAN behaves very unstably on LIN and the GAN with projection discriminator demonstrates the best performance. This is consistent with the results of visual comparison. The full table for 41 proteins can be seen in appendix D.

4.4 Caltech-UCSD Birds dataset

A superior performance of GAN with projection discriminator on the dataset with cells has inspired us to make a step in orthogonal direction and try projection discriminator for the task of text to image synthesis. To that end, we have chosen StackGAN++ [28] model, as it was considered to be state-of-the-art model for the task of text to image synthesis and it used concatenation in a way, similar to [20]. In aforementioned experiments projection discriminator outperformed discriminator with concatenation, so we expected similar behaviour for this task.

Figure 16: Samples from StackGAN++ with projection generated during training phase.

original StackGAN++

StackGAN++ with projection

Figure 17: A small bird with a yellow and grey crown, grey throat and breast, yellow belly and back with green-grey wings and tail and white wingbars.

Model trained with projection discriminator appeared to suffer from overfitting. It produced nice samples during training phase, while images for descriptions from the test set are almost unrecognizable. More examples of text descriptions and generated images can be found in appendix E.

5 Conclusion

Fluorescent microscopy can be used to register location of different proteins in living cells, though this imaging technique has certain limitations. Osokin et.al. [6] have shown, that generative adversarial networks can be fruitfully used to model multiple proteins of interest simultaneously. After a thorough comparison of various GAN models on smaller datasets, we applied two novel models, namely multichannel GAN and GAN with projection discriminator, to the LIN dataset.

We designed the multichannel GAN by spatially separating images of different classes into different channels. This allowed us to generate photographs of 6 proteins in connection with each other. GAN with projection discriminator has shown much better looking samples. This technique allowed us to generate 41 proteins, which was not possible before. Quantitative evaluation has shown, that

multichannel GAN and GAN with projection discriminator perform better on LIN than conditional GAN of [20].

However, we were not able to successfully introduce projection into StackGAN++, as the trained model suffered from overfitting. It produced nice samples for text descriptions from training set, while images for test descriptions were much worse and often ignored the content of the description.

Achieved results show, that the framework of generative adversarial networks has a rather wide applicability and there is still a room for research, especially in the topic of conditioning for GANs. The reasons behind poor performance of conditional GANs on LIN dataset should be further explored, as it may provide valuable insights into the process of conditioning for GANs. Another direction of research, which becomes open after demonstration of impressive performance of GAN with projection discriminator on LIN, is conditioning not just on class labels, but on a more field specific data, which will allow to reveal direct connection between more specific properties of cells and their visual appearance.

References

1. Are GANs Created Equal? A Large-Scale Study / M. Lucic [et al.] // ArXiv e-prints. — 2017. — Nov.
2. *Arjovsky M.*, *Chintala S.*, *Bottou L.* Wasserstein Generative Adversarial Networks // International Conference on Machine Learning. — 2017.
3. *Arora S.*, *Zhang Y.* Do GANs actually learn the distribution? An empirical study // International Conference on Learning Representations. — 2018.
4. Automatic differentiation in PyTorch / A. Paszke [et al.] // NIPS 2017 Workshop Autodiff. — 2017.
5. Conditional image generation with PixelCNN decoders. / A. van den Oord [et al.] // Advances in Neural Information Processing Systems 29. — 2016.
6. GANs for Biological Image Synthesis / A. Osokin [et al.] // International Conference on Computer Vision. — 2017.
7. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium / M. Heusel [et al.] // Advances in Neural Information Processing Systems 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — Pp. 6626–6637.
8. Generative Adversarial Nets / I. Goodfellow [et al.] // Advances in Neural Information Processing Systems 27. — 2014.
9. *Hinton G. E.*, *Sejnowski T. J.* Learning and relearning in Boltzmann machines. // Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1 / ed. by D. E. Rumelhart, J. L. McClelland. — MIT Press, 1986. — Pp. 282–317.
10. Image-to-Image Translation with Conditional Adversarial Networks / P. Isola [et al.] // The Conference on Computer Vision and Pattern Recognition. — 2017.
11. Improved Techniques for Training GANs / T. Salimans [et al.] // Advances in Neural Information Processing Systems 30. — 2017.
12. Improved Training of Wasserstein GANs / I. Gulrajani [et al.] // Advances in Neural Information Processing Systems 30. — 2017.
13. *Johnson J.*, *Gupta A.*, *Fei-Fei L.* Image Generation from Scene Graphs // arXiv:1804.01622.
14. *Kingma D.*, *Welling M.* Auto-encoding variational Bayes. // International Conference on Learning Representations. — 2014.
15. *Kingma D.*, *Ba J.* Adam: A Method for Stochastic Optimization // International Conference on Learning Representations. — 2015.
16. Learning deep representations for fine-grained visual descriptions. / S. Reed [et al.] // The Conference on Computer Vision and Pattern Recognition. — 2016.
17. Least Squares Generative Adversarial Networks / X. Mao [et al.] // International Conference on Computer Vision. — 2017.
18. *LeCun Y.*, *Cortes C.* MNIST handwritten digit database. — 2010.
19. *Lopez-Paz D.*, *Oquab M.* Revisiting Classifier Two-Sample Tests // International Conference on Learning Representations. — 2017.
20. *Mirza M.*, *Osindero S.* Conditional Generative Adversarial Nets // ArXiv e-prints. — 2014. — Nov.
21. *Miyato T.*, *Koyama M.* cGANs with Projection Discriminator // International Conference on Learning Representations. — 2018.
22. *Nowozin S.*, *Cseke B.*, *Tomioka R.* f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization // Advances in Neural Information Processing Systems 29. — 2016.

23. *Odena A.*, *Olah C.*, *Shlens J.* Conditional Image Synthesis with Auxiliary Classifier GANs // International Conference on Machine Learning. — 2017.

24. Progressive Growing of GANs for Improved Quality, Stability, and Variation / T. Karras [et al.] // International Conference on Learning Representations. — 2018.

25. Quantitatively Evaluating GANs With Divergences Proposed for Training / D. Jiwoong Im [et al.] // International Conference on Learning Representations. — 2018.

26. *Radford A.*, *Metz L.*, *Chintala S.* Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks // International Conference on Learning Representations. — 2016.

27. Spectral Normalization for Generative Adversarial Networks / T. Miyato [et al.] // International Conference on Learning Representations. — 2018.

28. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks / H. Zhang [et al.] // arXiv: 1710.10916.

29. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks / H. Zhang [et al.] // International Conference on Computer Vision. — 2017.

30. *Theis L.*, *van den Oord A.*, *Bethge M.* A note on the evaluation of generative models // International Conference on Learning Representations. — 2016.

31. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks / J.-Y. Zhu [et al.] // International Conference on Computer Vision. — 2017.

A Mixture of Gaussians

A.1 Unconditional setting

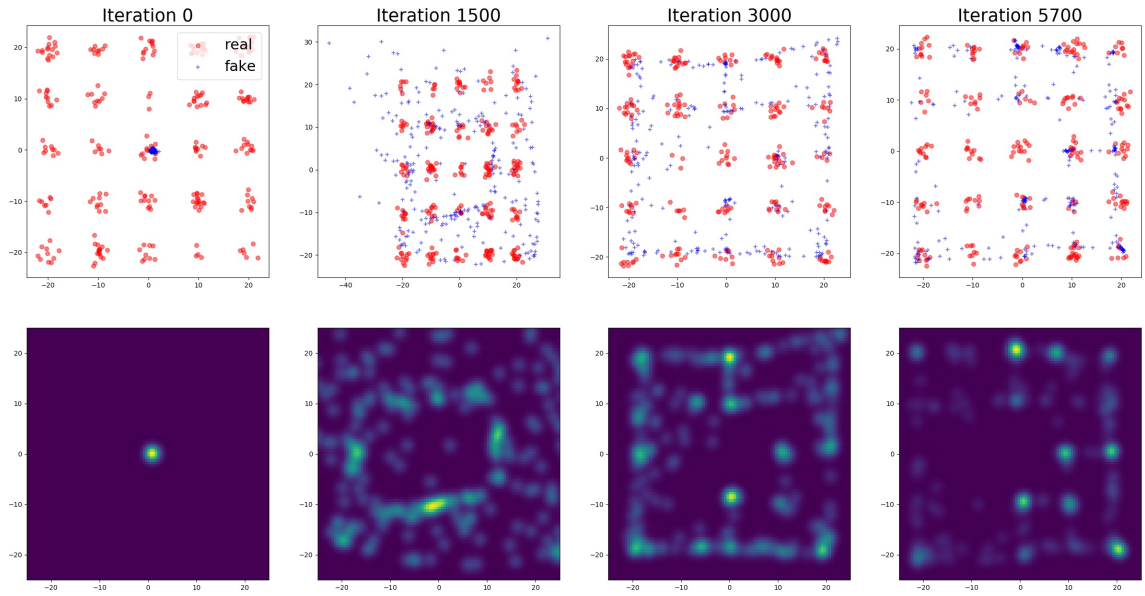


Figure 18: Samples and KDE for GAN model

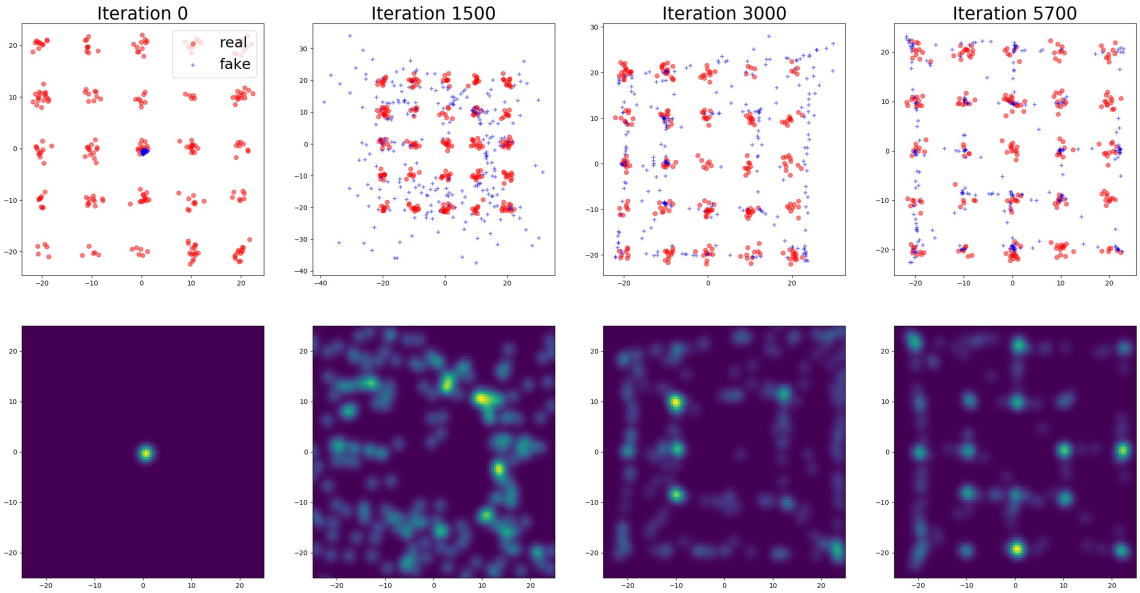


Figure 19: Samples and KDE for LSGAN model

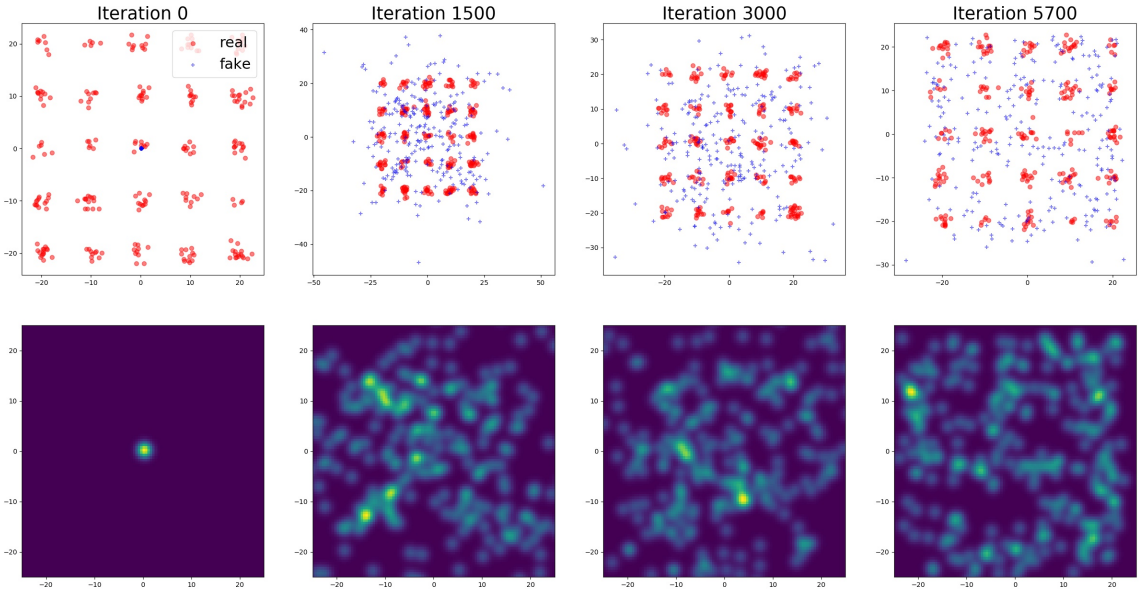


Figure 20: Samples and KDE for WGAN-GP model

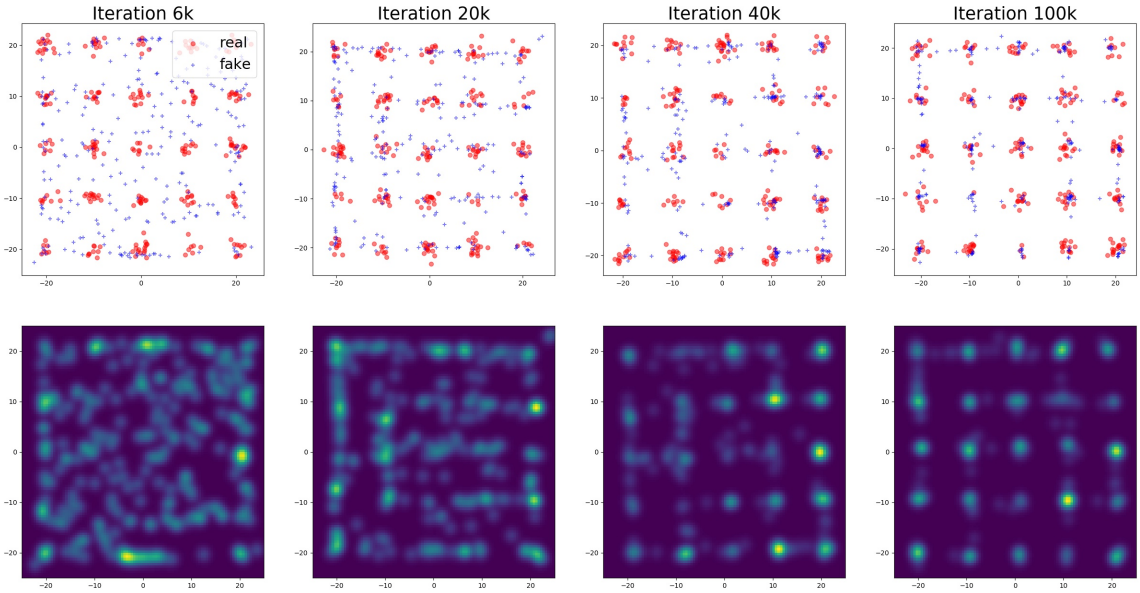


Figure 21: Samples and KDE for WGAN-GP model trained for 100k iterations

A.2 Conditional setting

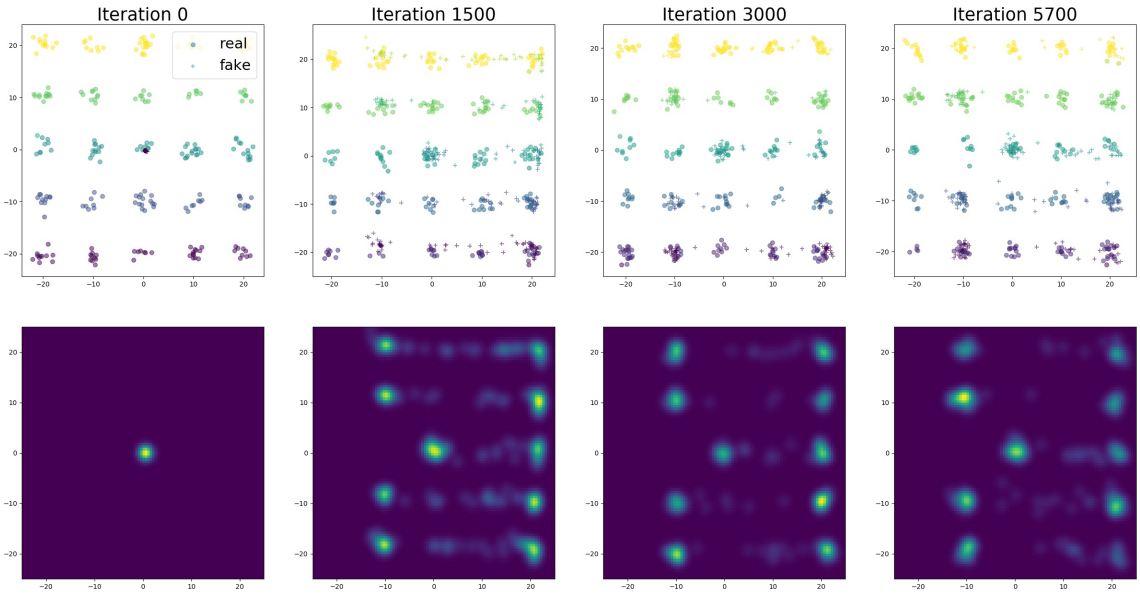


Figure 22: Samples and KDE for GAN model

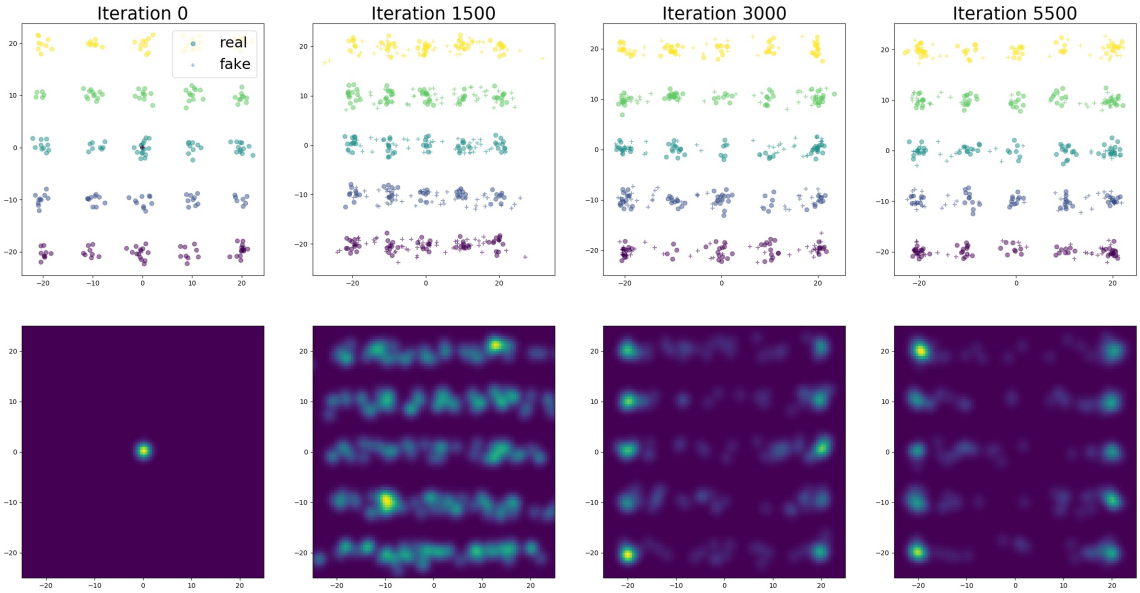


Figure 23: Samples and KDE for LSGAN model

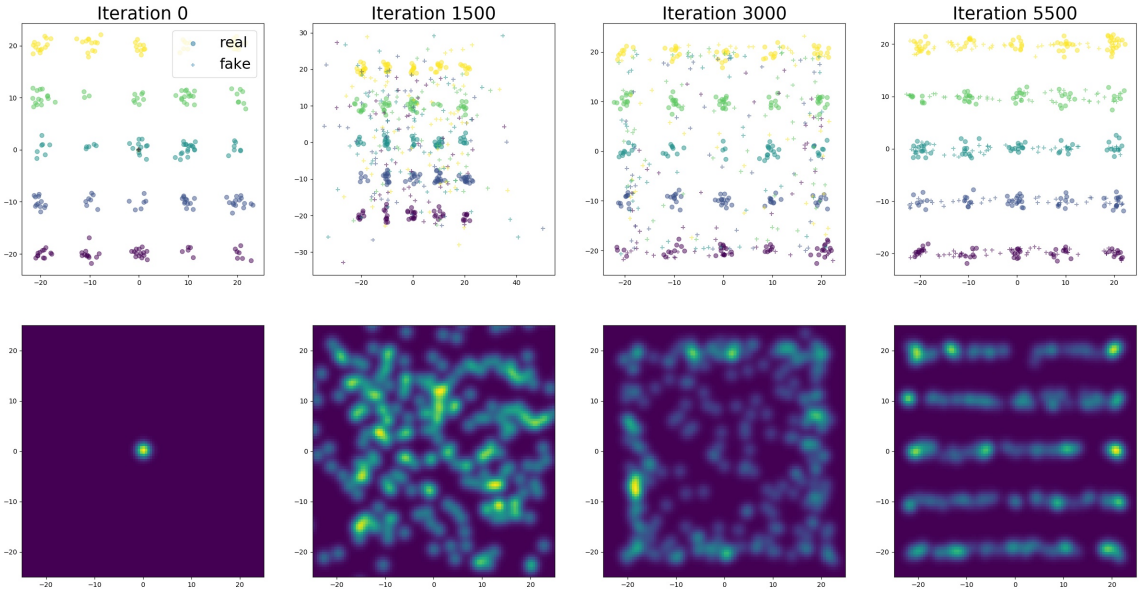


Figure 24: Samples and KDE for WGAN-GP model

B Multichannel GAN

Samples from Multichannel GAN trained on MNIST and CIFAR. Columns correspond to different classes, rows are independent samples. Classes in CIFAR are ordered as follows: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

Figure 25: Multichannel GAN on MNIST

Figure 26: Multichannel GAN on CIFAR

Samples from Multichannel GAN trained on LIN. Columns correspond to different classes, rows are independent samples. Classes are ordered as follows: Alp14, Arp3, Cki2, Mkh1, Sid2, Tea1.

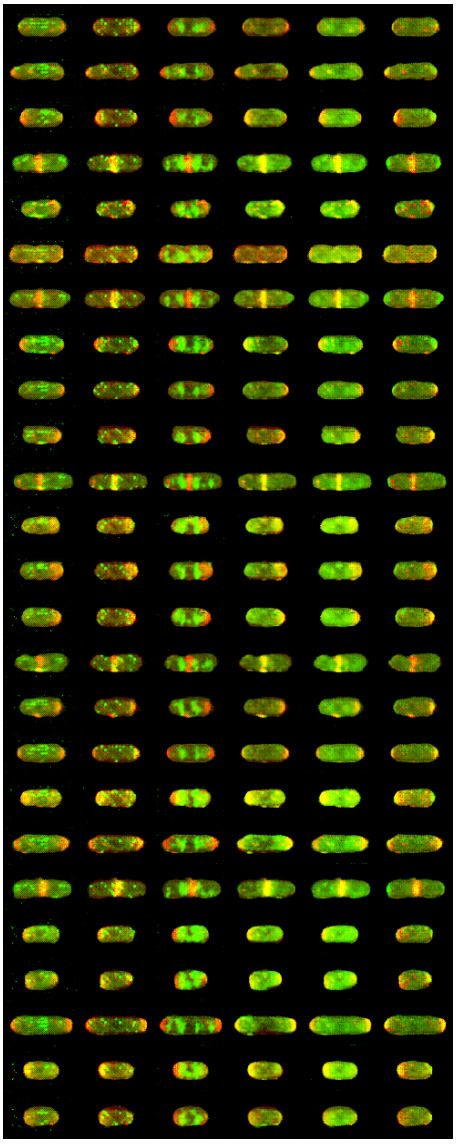
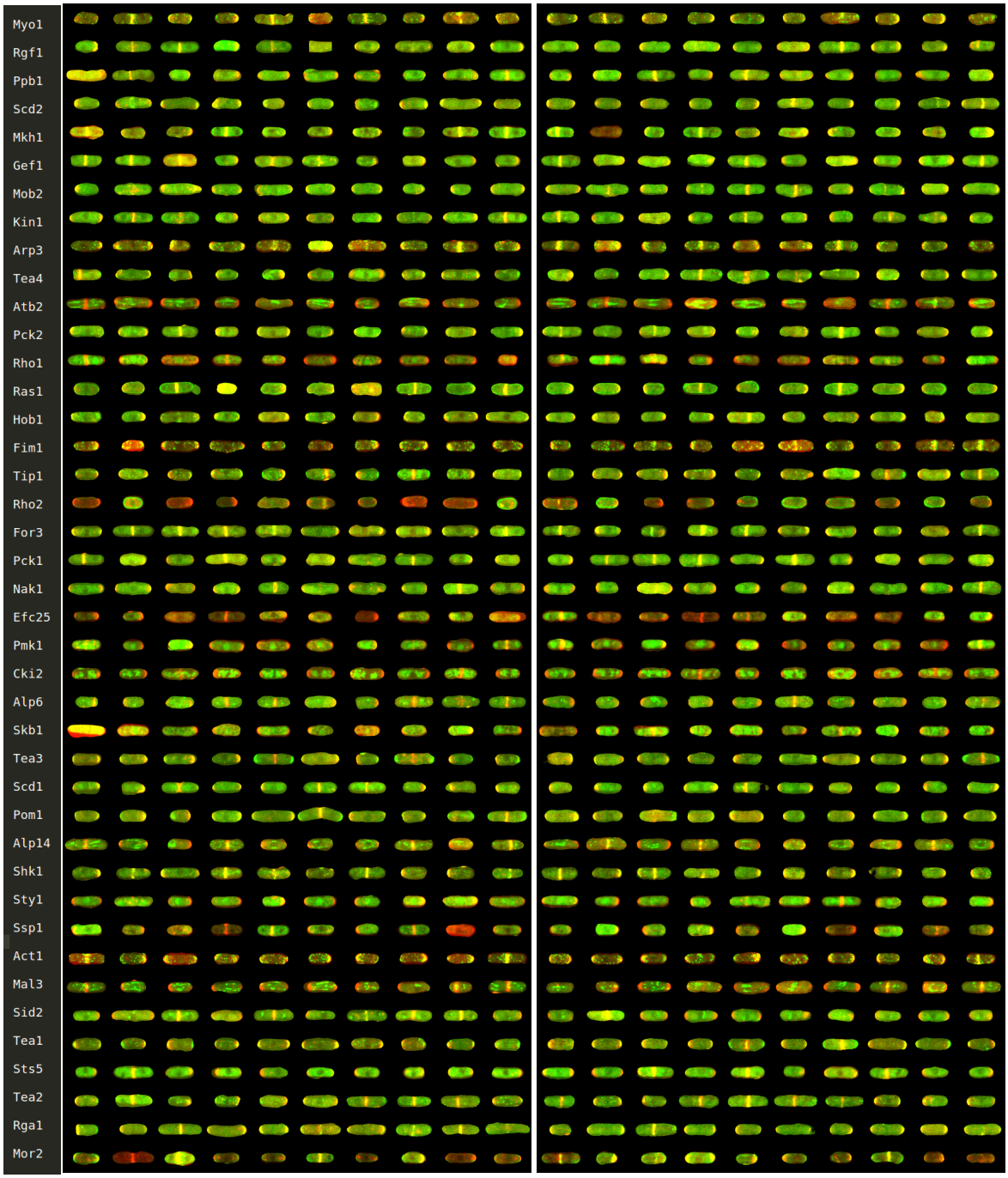


Figure 27: Multichannel GAN on LIN

C LIN projection discriminator



real photographs generated photographs

Figure 28: Visual comparison of real cells with 41 proteins and cells generated by GAN with projection discriminator. Best viewed in color and on a screen.

D FID for LIN classes

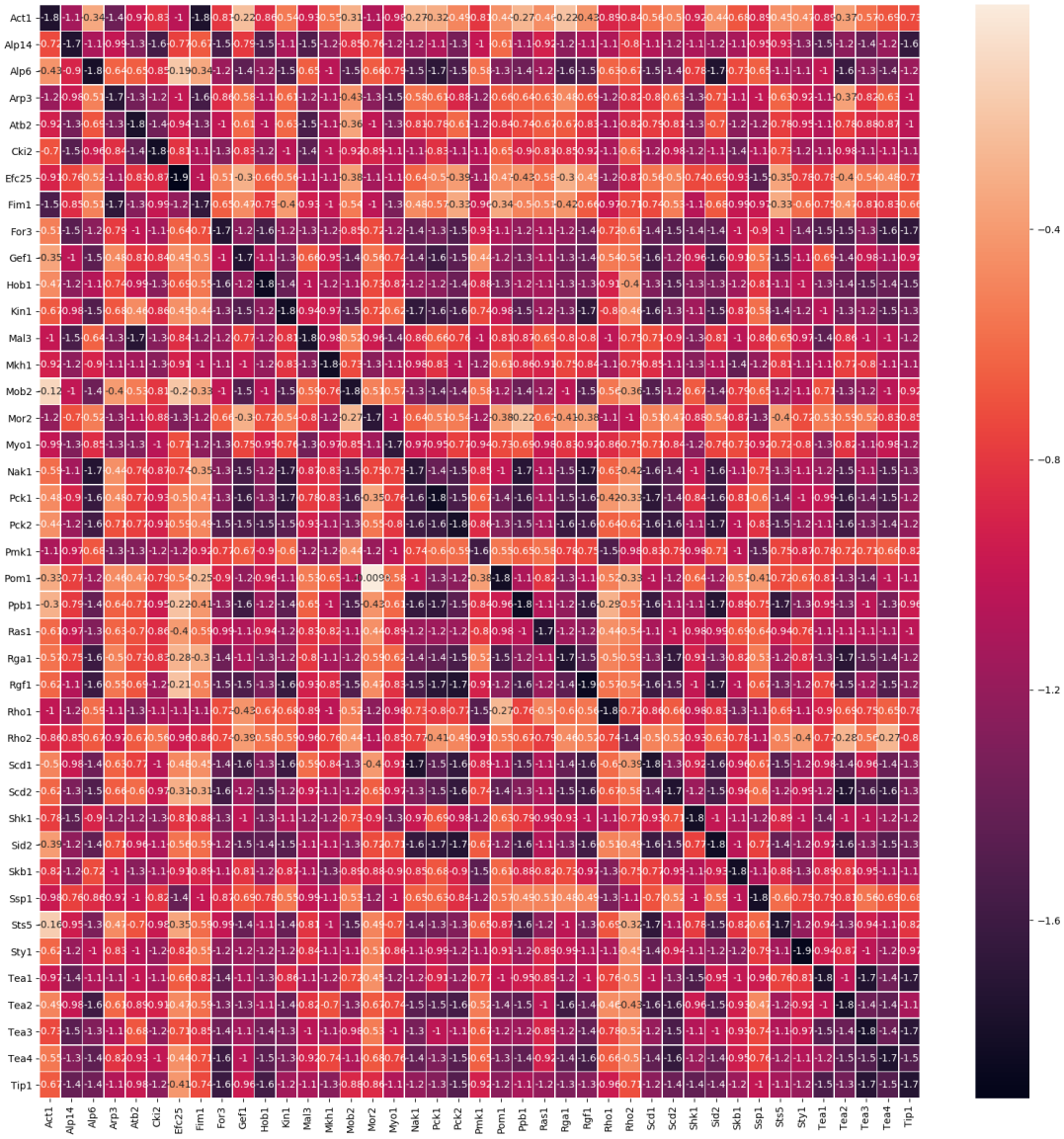


Figure 29: $\log_{10}$ of FID distances between classes of LIN

Protein	Real data	cGAN	multiGAN	projection
Act1	0.018	0.803	0.291	0.040
Alp14	0.018	0.940	0.270	0.028
Alp6	0.016	0.363	0.157	0.041
Arp3	0.020	1.334	0.300	0.034
Atb2	0.017	0.839	0.391	0.036
Cki2	0.015	0.408	0.134	0.040
Efc25	0.012	1.817	0.647	0.028
Fim1	0.022	0.455	0.207	0.037
For3	0.019	0.648	0.316	0.026
Gef1	0.018	2.087	0.303	0.032
Hob1	0.014	1.024	0.216	0.025
Kin1	0.015	0.508	0.197	0.030
Mal3	0.017	0.883	0.198	0.027
Mkh1	0.015	1.921	0.152	0.028
Mob2	0.017	0.745	0.207	0.030
Mor2	0.021	2.265	0.367	0.037
Myo1	0.022	0.991	0.074	0.032
Nak1	0.020	0.595	0.089	0.035
Pck1	0.014	0.665	0.215	0.032
Pck2	0.016	0.887	0.323	0.033
Pmk1	0.026	0.736	0.125	0.024
Pom1	0.015	0.653	0.114	0.034
Ppb1	0.015	0.615	0.129	0.039
Ras1	0.018	0.481	0.192	0.033
Rga1	0.018	1.095	0.184	0.037
Rgf1	0.014	0.654	0.199	0.028
Rho1	0.015	0.865	0.372	0.031
Rho2	0.038	0.881	0.347	0.039
Scd1	0.018	0.512	0.227	0.027
Scd2	0.018	0.639	0.263	0.031
Shk1	0.016	0.457	0.152	0.034
Sid2	0.015	0.468	0.272	0.027
Skb1	0.015	0.471	0.310	0.043
Ssp1	0.016	1.693	0.377	0.032
Sts5	0.018	0.611	0.201	0.028
Sty1	0.013	0.604	0.313	0.029
Tea1	0.017	0.633	0.179	0.038
Tea2	0.017	0.856	0.132	0.035
Tea3	0.016	0.612	0.137	0.029
Tea4	0.018	0.762	0.262	0.033
Tip1	0.019	3.105	0.201	0.034

Table 3: Comparison of several conditional models with FID.

E Caltech-UCSD Birds test samples

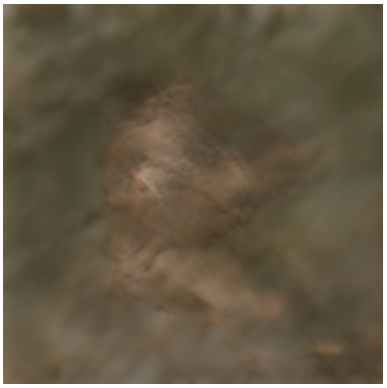
To demonstrate overfitting for StackGAN++ with projection, we have visualized samples from the original and modified models for some text descriptors.

original StackGAN++ StackGAN++ with projection

Figure 30: The medium sized bird has a dark grey color, a black downward curved beak, and long wings.

original StackGAN++ StackGAN++ with projection

Figure 31: A small bird with a green belly and grey wings.



original StackGAN++ StackGAN++ with projection

Figure 32: This bird has wings that are brown with a white body.

original StackGAN++

StackGAN++ with projection

Figure 33: A small bird with a yellow and grey crown, grey throat and breast, yellow belly and back with green-grey wings and tail and white wingbars.

original StackGAN++

StackGAN++ with projection

Figure 34: This large black bird with large outer retrices and with a large beak.