A performance analysis of the generative adversarial networks

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

A performance analysis of the generative adversarial networks

Master's Thesis

Heiki Riesenkampf
Department of Information Technology and Electrical Engineering

Advisors: Dr. Radu Timofte
Dr. Carsten Eickhoff
Supervisor: Prof. Dr. Luc Van Gool

August 1, 2018

Abstract

Generative Adversarial Networks (GANs) have been applied in many computer vision problem domains, but generating unique photo realistic images has remained one of its most popular use cases. The quality of such networks is measured by the quality of the images they are able to generate. Despite many recently proposed metrics, measuring the quality of generated image sets remains an unsolved problem. In our work we propose a GAN training setup which allows us to optimize for either maximum image quality or maximum sample diversity within the generated data set. We achieve this by using a conditional GAN architecture and using synthetic class labels. The image labels are generated using unsupervised clustering and we are able to guide the network training process by changing the numbers of cluster centers. We evaluate our GAN performance with the two most popular methods: the inception score (IS) and the Fréchet inception distance (FID). Our GAN improves the best reported performance for both metrics by 2% and 10% respectively. We discuss the shortcomings of both metrics and propose improved methods for evaluating the quality of synthetic data sets by comparing original and generated image sets using unsupervised clustering methods.

Acknowledgements

First and foremost I am extremely grateful to my two advisors: Radu Timofte for finding time every week to discuss my research and Carsten Eickhoff for being so forthcoming by being my supervisor even after changing his role and institution. I am also grateful to my supervisor professor Luc Van Gool for giving me the opportunity to do my master thesis in the Computer Vision Lab at ETH Zrich. I also thank Nikolay Kobyshev for introducing me to CVL and helping me with my practical work. Finally, I would like to thank Kevin Klein and Igor Susmelj for interesting discussions and their advice during my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision research has for a long time mainly focused on better understanding the visual domain with algorithms. There has been very fast progress in the visual computing domain since the discovery of deep learning and back propagation. Deep convolutional neural networks have improved the best object recognition beyond human capabilities achieving near perfect results on many popular public data sets. One of the most recent big advancements was the introduction of Generative Adversarial Networks (GANs)[6] which have greatly progressed many new research areas in the computer vision domain. These include but are not limited to photo realistic image generation, semantic style transfers and image infilling. We also use an adversarial network architecture in our work.

Compared to previous machine learning architectures GANs are unique because they consist of two neural networks playing an optimization minimax game against each other. One network, called the generator (G), receives random noise vector z as input which the network learns to convert into a photo realistic image. The other network, the discriminator (D), receives either a real image or a generated image as input and it learns to predict whether the input is from the original data set or has been generated. The two networks try to outsmart each other during the training as one is trying to maximize G(z) while the other tries to minimize the same value. After training finishes the generator is able to create new unique realistic images and the discriminator can well distinguish between real and fake images. An overview of GAN's architecture can be seen on 1.1.

The value of generating new realistic images lowers the cost of acquiring image data for training machine learning algorithms. Supervised algorithms require large amounts of labelled data for achieving good results and GANs are very helpful in generating new data. The original GAN architecture did not allow generating images with pre-defined objects. This became possible with conditional GANs, where G and D receive extra information to guide the training process. We use a GAN with auxiliary classifier (AC-GAN)[13] in order to generate images that feature certain pre-defined objects. In AC-GAN setup, the G uses both the image class information c and noise z to produce fake samples $X_{fake} = G(z, c)$. We propose a new method for giving the class label information to G using conditional instance normalization (CIN) layers. Our hypothesis is that it will allow us to generate class conditional images and improve the overall image quality.

In addition to being able to generate labelled image samples, a conditional GAN also helps in improving the quality of the synthetic images. The photo realism of generated images is best measured by human

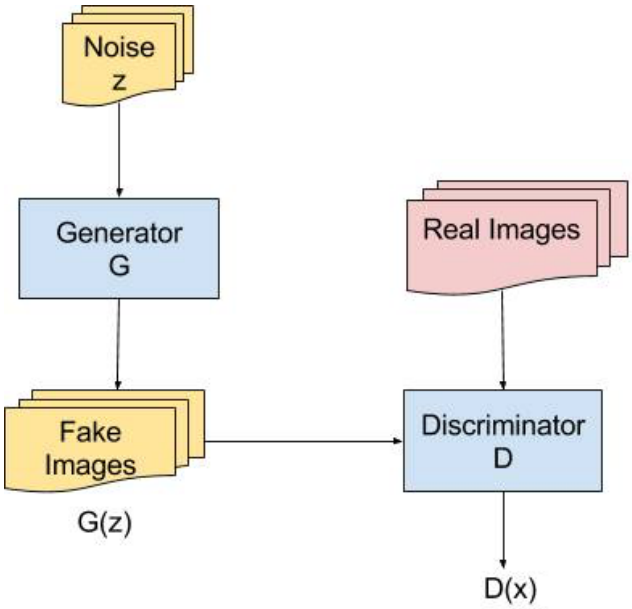


Figure 1.1: GAN architecture

evaluation but since such tasks demand a lot of resources, automated calculated metrics have been proposed to measure the quality of generated images. The two most popular metrics for evaluating the quality of synthetic images are the Fréchet Inception distance (FID) and the Inception score. In our work we use both metrics and discuss their shortcomings and propose additional methods for more robust image quality evaluation.

The purpose of an adversarial network is able to generate images that are similar to the original data set, but also retain the diversity of the original data set. For example, when the original data consists of images featuring ten different object classes, then the generated data set should also include different versions of objects from all original classes. In our work we use the CIFAR-10 public dataset, that has around 60000 images with 10 different object classes. Because most real life image data sets are not labelled, we generate synthetic labels for our images using unsupervised clustering methods. We assume that there is correlation between the image clustering statistics and final GAN training performance.

1.1 Focus of this Work

This work focuses on methods how to guide a conditional GAN's training process in order to either achieve best image quality or maximum data set diversity. We are trying to prove our assumption that using different synthetic label generation setups we can guide the GAN training process. Using our new CIN-GAN architecture, we also evaluate current data set quality metrics and propose better alternatives. Our contributions can be summarized as follows:

- We propose a method for choosing a good number of clusters in generating synthetic image labels for achieving best image quality and data set diversity
- We develop a novel method for inserting class information for conditional GANs using instance normalization layers.
- We achieve the best reported Inception score and FID on CIFAR-10 data set using fewer training iterations with proposed architecture and training setup.
- We show and discuss how the Inception score and FID progress over time during the GAN training process for different hyper-parameters.
- We discuss shortcomings in recently proposed image set quality metrics and propose a more robust alternative.

1.2 Thesis Organization

In Chapter 2 we give an overview of related previous work in the same research domain. In Chapter 3 we focus on the tools and methods that we used in our experiments. This includes all the used algorithms, formulas and the architecture of our adversarial network. In Chapter 4 we give an overview of our experimentation setup and evaluation methods. It also includes the most relevant data and results from completed experiments. In Chapter 5 we discuss the meaning of our achieved results. We also compare our results to previously reported findings and suggest how our work could be practically applied. In the last chapter we conclude our work and propose new potential areas for further research.

Chapter 2

Materials and Methods

In this chapter we give an overview of the research that our work is built upon. It gives a brief understanding of the latest research in the area of GANs and different methods for evaluating image quality.

2.1 Generative Adversarial Networks

Generative Adversarial Networks were first proposed by Goodfellow *et al* in 2014. Their goal is to learn generator's distribution p_g over data x. Given an input noise variable $p_z(z)$ the generator maps noise vectors into data domain by $G(z)$. The discriminator receives input x and outputs a scalar probability $D(x)$ of x belonging to original data set rather than p_g. D needs to learn how to maximize the probability of assigning the correct label to samples both from training set and samples from G. We simultaneously train G to minimize $log(1 - D(G(z)))$ and D to maximize $log(D(x))$. The two-player minimax formula can be formulated by (2.1).

$$\min_G \max_D \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r} [\log(D(x)] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z} [\log(1 - D(G(z)))] \tag{2.1}$$

In an ideal setup after the training process D would not be able to distinguish between real and fake examples, thus $D(x) = \frac{1}{2}$. Since the publishing of the generative network architecture, there have been many new proposals how to improve the quality and training process of GANs.

2.1.1 Wasserstein GAN

The original GAN architectures are very hard to train due to instability of training process and frequent mode collapses. Instability would have resulted in increasing gradients and non-converging training and mode collapse would result in the network only learning a small subset of the original data distribution. To solve the problem of vanishing/exploding gradients and mode collapse, Arjovsky *et al.* proposed to use Earth Mover (EM) or Wasserstein-1 distance instead of the original Jensen-Shannon (JS) divergence as the loss function for the GAN[1]. The Wassertein-1 distance is defined as (2.2).

$$W\left(\mathbb{P}_r, \mathbb{P}_g\right) = \inf_{\gamma\in\Pi(\mathbb{P}_r,\mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{2.2}$$

Since the EM distance provides a continuous differentiable gradient over the entire feature space, it is always possible to update the network during the back propagation phase in the right direction. The updated loss function for D becomes (2.3).

$$L(D) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathop{\mathbb{E}}_{x\sim\mathbb{P}_g}[\log(1 - D(x))] \tag{2.3}$$

Using the Wassertein-1 distance as the GAN's loss function proves to avoid mode collapses and improves the stability of the training process. The updated minimax optimization can be formulated by (2.4).

$$\min_G \max_{D\in\mathcal{D}} \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[(D(x)] - \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[(D(\tilde{x})] \tag{2.4}$$

Therefore, using the updated loss function has become the default for many new proposed GAN architectures.

2.1.2 Gradient penalization of WGANs

The original work introducing the Wassertein-1 loss for training GANs proposed to use weight clipping in order to enforce a Lipschitz constraint. Since that still sometimes leads to unstable training process, Gulrajani *et al.* proposed to use gradient penalty for enforcing the same constraint[7]. In addition to the original WGAN loss function they add a penalizing term given by (2.5).

$$\lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right] \tag{2.5}$$

In addition to the added penalty term, they use layer normalization instead of batch normalization in the D as the penalization objective is not valid in a batch optimization setting. Since their setup has proven to improve the training quality significantly over the original weight clipping, we also use their formulation for our GAN architecture.

2.1.3 Deep Convolutional GANs

In addition to improvements to the loss function, there are works improving the architecture of the initial GAN and one of the popular ones is DCGAN, which uses deep convolutional neural networks both in the G and D[14]. Convolutional nets have been very successful at object recognition tasks and they prove to be as useful in GANs for generating new photo realistic images. Their main advantage comes from combining the learning of simple patterns in the first layers of the network into more defined high level objects in the last layers of the network. To adapt the original convolution net for the GAN architecture Radford *et al.* introduced batchnorm in both discriminator and generator, replaced pooling layers with strided convolutions and removed fully connected layers from the original architecture. Due to their architecture's superior performance, all of our experiments also use deep convolutional nets for both G and D.

2.1.4 Conditional GANs

The original GAN network architecture can be augmented by adding extra information for both G and D. One proposed method is to add class labels as input for both G and D and produce class conditional samples. This has been proven to improve the quality of generated examples [18]. An overview of previous conditional GAN architectures and AC-GAN architecture can be seen on 2.1

The D can also be augmented with an auxiliary decoder network that learns to predict the sample image class in addition to the probability of the sample being from the real data set. Odena *et al.* combines both methods in their AC-GAN, where the generator has in addition to the noise also a class label c as input for

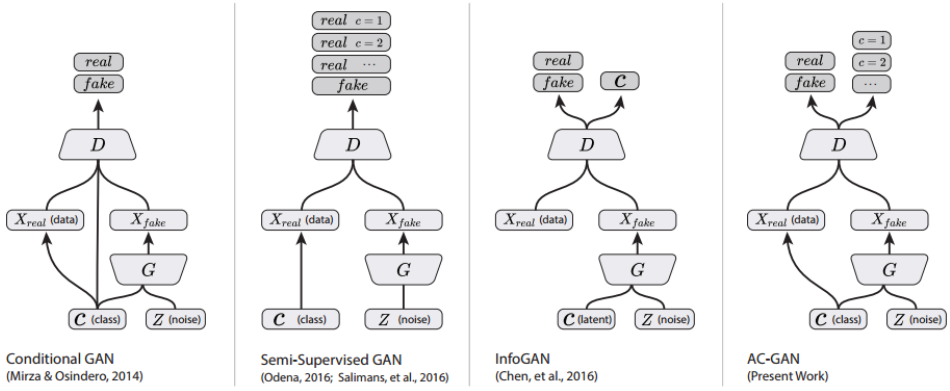


Figure 2.1: Conditional GAN architectures [13]

generating the fake image: $X_{fake} = G(c, z)$. The D outputs both the class probability of given image and the source probability. In such a setup the loss function has two components, the source loss (2.6) and the class loss (2.7)

$$L_S = E[\log P(S = real|X_{real})] + E[\log P(S = \text{ fake } |X_{fake})] \tag{2.6}$$

$$L_C = E[\log P(C = c|X_{real})] + E[\log P(C = c|X_{fake})] \tag{2.7}$$

Odena *et al* also show that AC-GAN architecture stabilizes the training process and increases the quality of the generated samples. They also note that they are able to generate good quality samples from each class of ImageNet by training separate GANs for different subsets of ImageNet classes.

2.1.5 Batch Normalization

Convolutional networks usually include batch normalization (BN) as part of their architecture. The method speeds up the training process by normalizing the input feature weights' mean and variance during training[10]. During training the network tried to learn two parameters: the weight and the bias (2.8).

$$\frac{\gamma(x - \mu)}{\sigma} + \beta \tag{2.8}$$

BN had mostly been used for making neural network training process more efficient until De *et al.* proposed to use the layer for introducing extra information into the network. They use the batch normalization layer for adding linguistic information into their convolutional net in order to improve its performance[4]. Another alternative to BN proposed by Ulyanov *et al.* is instance normalization (IN) which is similar to BN, but applies the normalization on the images individually, not on the entire batch. IN has shown to encouraging diversity in the generated data sets[17].

2.2 Image set quality metrics

The purpose of GAN architecture research is to propose new methods for generating high quality images that are both photo realistic and have high diversity. In order to measure these qualities many methods have

previously been proposed. The most robust method is to ask humans to evaluate the quality of the images, but this method is very limited due to lack of resources and does not scale to larger data sets which are required to also evaluate the diversity aspect.
In addition to human evaluation methods researchers have proposed many automated ways how to statistically measure image data set quality.

2.2.1 Inception score

The most popular method for evaluating image quality was proposed by Gulrajani *et al.* which compares the KL divergence of Inception model conditional label distribution $p(y|x)$ and marginal class distribution $\int_x p(y|x)p_g(x)$. The score is calculated using the pre-trained Inception v3 Network on the ImageNet public data set with 1008 distinct object classes[5]. Inception score is calculated by taking the exponent of the distributions' KL divergence: (2.9)

$$\mathrm{IS}(G) = \exp\left(\mathop{\mathbb{E}}_{x\sim\mathbb{P}_g} D_{KL}(p(y|x)\|p(y))\right) \tag{2.9}$$

The intuition why Inception score is well correlated with human judgment comes from the assumption that the generated images should only feature one clearly distinctive object, thus resulting in low entropy for $p(y|x)$. In order to evaluate the data diversity the formula looks at class distribution and in case the data set features images with every potential object from ImageNet data set the class distribution $p(y)$ will be high.

2.2.2 Fréchet Inception distance

A new alternative metric called Fréchet Inception distance (FID) was proposed by Heusel *et al.* for better measuring the quality of image sets[9]. Instead of only looking at the generated images class distribution, FID takes into consideration also the original data set distribution. FID uses the first two moments (mean and variance) of the Gaussian distributions of the Inception distance of the original (m, C) and syntethic data sets(m_w, C_w). The difference of the two data sets is calculated using the Wasserstein-2 distance given by (2.10)

$$d^2\left((m, C), (m_w, C_w)\right) = \|m - m_w\|_2^2 + \mathrm{Tr}\left(C + C_w - 2\left(CC_w\right)^{1/2}\right) \tag{2.10}$$

The clear advantage of the FID is that it takes into consideration also the real life data set. The intuition behind FID is that the first term measures the difference of means of the two distributions which is supposed to capture how different are the Inception network activation values of the two data sets. The second term captures the trace values of the covariances of the two matrices, in effect capturing how different the variances of each predicted class label distributions are.

2.3 Generating synthetic class labels

A conditional GAN uses both the original images and image labels as input for the G and D. Even though research image data sets often also include the human produced labels, in most settings it is too resource expensive to collect high quality human labels. Because of that different automated methods have been proposed to generate labels for training purposes.

2.3.1 K-means clustering

In order to group the images in meaningful manner many unsupervised learning approaches could be used. We chose to use batched K-means++ clustering proposed by [2] in order to generate equally distributed sets of images. The purpose of the algorithm is given number of classes k and n data samples from $\mathbb{R}^d$ to find cluster center locations so that the sum of the mean squared distance between the data samples and the nearest centroid would be minimal. The difference between K-means++ and the original version is in the initial centroid initialization. In K-means++ each centroid $c_i$is picked $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$ where $D(x)$ is the shortest distance from data point to the closest centroid we have already picked. Such initialization significantly speeds up the clustering process and makes the use of our labelling algorithm much more practical.

2.3.2 Specifying the number of clusters

Choosing a good number of clusters for K-means algorithm is an unsolved problem in unsupervised learning. Many different solutions have so far been proposed that we considered when doing our experiments. Most common method for determining a good number of clusters in called the elbow technique[12]. It looks at the mean squared error (2.11) by calculating the distance between the data points y_i and their closest cluster center c_k and doing it repeatedly for multiple values of K, where S is the partition of the entity set.

$$MSE(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} \|y_i - c_k\|^2 \tag{2.11}$$

The elbow technique is a visual method, which looks at the graph of MSE for different cluster numbers and determines a good value for K by looking at which point the cost drops significantly and reaches a plateau afterwards. The rationale for this technique is that after a certain point of K further increasing the number of clusters would add a cluster which would end of very close to an existing one and thus not significantly reducing the MSE of the final result. Even though the elbow point is fairly ambiguously determined it is a fast and simple method for coming up with a good number of clusters. Other techniques that have been proposed are more complex and are not proven to offer better performance, which is why we only tested the elbow method in our experiments.

2.4 Our contribution related to previous research

Our work builds upon a large body of previous research and our contributions can be summarized as follows:

- We augment the previously proposed AC-GAN[18] architecture with conditional instance normalization layers, which allow us to use the CIN-GAN for generating conditional samples.
- We use existing K-means[2] clustering on inception network activation values to evaluate synthetic image labels quality and effect on GAN training performance.
- We propose more robust image set evaluation metric by using the difference of the means and the variance difference of the generated and original data sets in order to optimize and debug the GAN training process with more precision.

Chapter 3

Experimental setup

In the following chapter we give an overview of the technical methods and frameworks used for implementing the experiments in our research. It also covers the more theoretical aspects of our work.

3.1 Software frameworks

All experiments were conducted using different machine learning networks implemented in python. Both python versions 2 and 3 were used at different stages of the research. Many parts of the work was based on previous implementations of open sourced code.

3.1.1 GAN architectures

Most of the experiments were done using open-source machine learning libraries. Since our GAN architecture was strongly based on DCGAN and AC-GAN our initial experiments were executed on modified versions of `https://github.com/carpedm20/DCGAN-tensorflow` and `https://github.com/alex-sage/logo-gen`. Since both of these code bases are based on TensorFlow, it was also the framework that most of our experiments were tested upon. Some later stage experiments were also run on PyTorch implementations and the code base for those experiments was the official DCGAN implementation available at `https://github.com/pytorch/examples/tree/master/dcgan`.

3.1.2 Image evaluation frameworks

For measuring image quality and evaluating different experiment results, we used open sourced implementation of inception score from OpenAI available at `https://github.com/openai/improved-gan/tree/master/inception_score` and FID calculations by Johannes Kepler University Linz available at `https://github.com/bioinf-jku/TTUR`. Since we also used new metrics for evaluating GAN performance we had to implement some functions from scratch. Example code of how to calculate the two separate components of FID can be seen in appendix B.

3.2 Hardware setup

All experiments were carried out on the BIWI compute cluster provided by the Computer Vision Lab at ETH Zurich. Most trainings were supported by GPU acceleration in order to speed up the calculations. We

used NVIDIA TITAN X GPUs for our experiments. Using such a setup, most GAN training experiments were able to be completed in 12-18 hours.

3.3 Data sets

The used data set had to fulfill two main requirements for our experiments:

- The images had to be in low dimensions due to restrictions in processing power and time
- Images had to feature distinct objects from a small number of classes

These requirements limited our data set choice significantly.

3.3.1 CIFAR-10

The data set we used in our experiments was CIFAR-10, which features 60000 images with 10 different object classes. The data set also features labels for each individual image. The image dimensions are 32x32 pixels and are colored.

3.4 Generating synthetic image labels

Using a conditional GAN architecture improves both training stability and quality of generated images. To train a conditional GAN, in addition to the data set, one also needs each individual image class label. As already mentioned, getting labelled image data is time consuming and demands a lot of human resource. To make our training approach economically feasible and applicable to any image data (both labelled and unlabelled) we propose a unsupervised learning algorithm for generating synthetic image labels. The goal is to create meaningful class labels for any given data set that would help the training process converge faster and achieve better performance only using information about the images. Labels are most commonly collected by humans, who decide what object on the image characterizes the particular image best. This means that labels received using human labor allow grouping the images by their semantic meaning. Since we are trying to replicate the ground truth semantic label information we had to come up with an alternative method for grouping images by their semantic meaning using automated methods. One way how to do that is by using pre-trained image classification networks, which try to predict which objects are featured on each image. Using such pre-coded information we can find out what are the most likely objects on the images and group the images based on the feature predictions. While such an approach is deemed to be less accurate than human labelling we show that synthetic labels created using this method actually help us achieve a similar performance compared to using ground truth label information.

3.4.1 Clustering using pre-trained classifier network

In order to group images into semantically meaningful clusters we used a pre-trained ResNet classifier. The classifier has been trained on the ImageNet public data set, which consists of over 14 million images from over 20000 different classes of objects. We use the ResNet-200 architecture[8] classifier trained by Facebook available at `https://github.com/facebook/fb.resnet.torch/tree/master/pretrained`. We feed each image into the classifier and extract the final fully connected layer activation values. Each value represents the possibility of the image featuring an object from the respective class.

We then use PCA unsupervised dimensionality reduction method to scale the feature vectors down to 2048 dimensions. Using the compressed features as latent space, we then use K-means++ clustering to generate semantically meaningful labels for the images. Considering that CIFAR-10 and ImageNet do also contain some similar classes, we are roughly generating similar class labels that human annotated labels would be like. With this method, it is possible to generate ground-truth like image labels using unsupervised learning methods. We also tested the same technique with VGG16[16] pretrained network `http://www.vlfeat.org/matconvnet/pretrained/` and received similar image clusters proving the general usefulness of the technique.

3.4.2 Choice of class order K

In K-means algorithm, one has to specify the number of clusters beforehand. As mentioned in the clustering part of related work, one can use the elbow method for determining a good number of clusters. However, this technique requires running a large number of experiments before finding a suitable candidate for K. Due to operating in a high dimensional space and with large data sets our clustering experiments were fairly time consuming (1-4 hours) and therefore one could not afford to try a large number of potential Ks to come up with a good number. This is why we tried to come up with a good heuristics for choosing a good number of clusters for the label generation task.

Since the main motivation was to generate photo realistic images we chose to evaluate the performance of clustering using inception score and FID. The goal was to come up with a method to calculate a good K based on statistics of the data set and not have to run the clustering algorithm multiple times beforehand. Our experiment was mostly based on a trial and error basis, where we first found a good solution, then developed a feasible heuristics for the correlation and then back-tested the solution on a different image set.

3.4.3 Conditional instance normalization

Since we were using a conditional GAN, we had to input the associated image labels both into the G and D. Since PyTorch only offers on a fairly high level API we had to implement our own conditional instance norm (CIN) class in order to make the GAN conditional. The code for the forward pass can be seen here. The full python code for the CIN class can be found in Appendix A.

Listing 3.1: Conditional instance normalization forward pass function

```
def forward(self, layer_input):
    self._check_input_dim(layer_input)
    n_labels = self.labels.shape[0]
    label_dim = self.running_mean.shape[1]
    offset = torch.zeros([n_labels, label_dim], dtype=torch.float32)
    scale = torch.ones([n_labels, label_dim], dtype=torch.float32)
    weight = offset[labels][:,:,None,None]
    bias = scale[labels][:,:,None,None]
    return F.instance_norm(
        layer_input, self.running_mean, self.running_var, weight, bias,
        self.training or not self.track_running_stats,
        self.momentum, self.eps)
```

As mentioned before, the batch normalization tries to learn affine parameters γ and β in order to normalize all the layer activation values between -1 and 1 with standard deviation of 1. Instance normalization

is similar, but applies the normalization not per individual batch of images, but image channels instead. Instance normalization works great for style transfer networks and we show that it also works great as a conditional GAN layer. Instead of learning the γ and β parameters separately, using the forward pass, we force the network to also learn the class condition via the one-hot encoded class label. When generating class conditional images with a trained generator network it was clear that the network had learned to generate class specific images using the extra input information.

3.5 Experiment network training process

The majority of the GAN training experiments were done using the same architecture. The hyperparameters were fine-tuned during the first couple of trial runs and then kept constant to make sure that they do not affect the results of the experiments.

3.5.1 GAN experiment architecture

As previously mentioned, all experiments were carried out using an adversarial network with a deep convolutional net for with G and D and a further modification was to add an auxiliary classifier in the D.

The architectures for the G and D are very similar. Both consist of four building blocks: one linear layer followed by CIN and three down sampling convolutional blocks. The first two convolutional blocks also included a CIN layer.

3.5.2 GAN experiment hyperparameters

Most hyperparamaters for the training experiments stayed constant as they produced great results within nominal time periods.

- Optimizer: Adam
- Optimizer running average and square coefficients: 0.5; 0.9
- Learning rate: 0.0001
- Batch Normalization with a momentum of 0.9
- Batch size: 64
- Training iterations: 50000
- G to D update rate: 1:5
- G's AC-GAN loss to WGAN's loss: 1:10

Chapter 4

Experimental Results

In this chapter we summarize each of our previously mentioned assumptions and give an an overview of each experiment we conducted to test them.

4.1 Finding a good number of clusters

Our first goal was to find a good number of clusters and generate image labels using those clusters so that we could later use both the images and labels for achieving best performance on our conditional GAN. Our first approach was statistical analysis of the clustering process and later we analyzed the end-to-end training. The purpose of the experiments was to find a correlation between the number of cluster centroids and GAN performance.

4.1.1 Choosing cluster number statistically

Our first hypothesis was that it should be possible to come up with a good number of cluster centroids using statistical methods for generating the synthetic labels in order to train the conditional GAN. To test it, we ran the K-means++ algorithms on the calculated ResNet activation values of CIFAR-10 images. We picked common powers of two between 8 and 4096 as the number of clusters to experiment with and also added 10 to test the original data setup. We plotted the mean squared error (MSE) for each cluster value on graph 4.1. We also tested the VGG16[16] pre-trained network for calculating the activation values, but the results were similar with previous findings, which is why we decided to continue experiments only using ResNet features. The reported values are the average of three runs to achieve more robust results. The experiments showed, that results using the same cluster number were very similar. We were hoping to see a clear elbow point so it would be possible to propose a good cluster number candidate, but the experiment showed that the MSE values for increasing cluster sizes increased linearly without any outliers. We also measured clustering convergence speed, but there we found no correlation between the centroid count and speed.

4.1.2 Evaluating GAN performance on different clusters using IS and FID

Because we did not manage to propose a suitable class number using only statistical methods, we extended the experiments with training GAN using the generated image labels. This way we were trying to prove the correlation between the cluster centroid number and GAN performance using end-to-end tests. We measured GAN performance using both the Inception score and the FID score. We calculated both measures by generating 10000 image samples after every 2500 training iterations and calculating both scores using

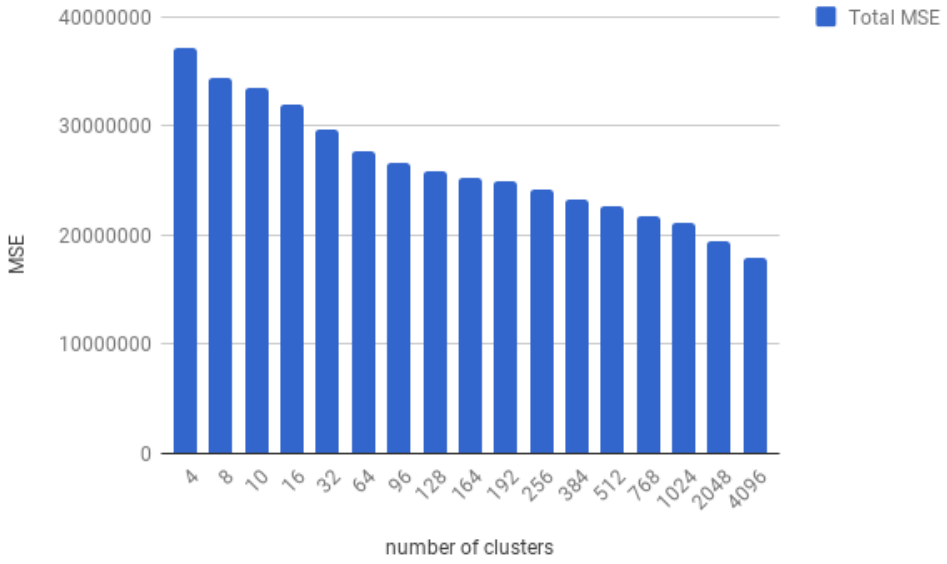
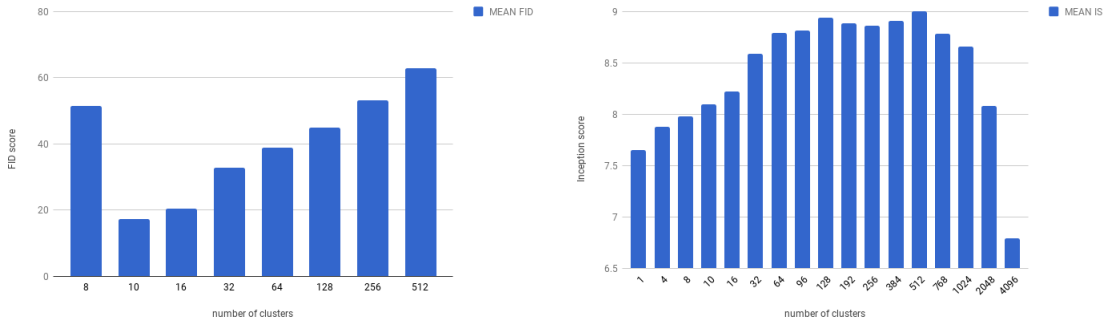


Figure 4.1: K-means++ clustering MSE values for different K values

the same generated data set. A data set of 10000 has proven to be enough to avoid any bias and make the effect of outliers negligible[9]. To report the scores, we ran each experiment 5 different times and report the average scores of 5 different runs with the same hyperparameter setup. The reported scores are calculated taking the average of the 4 last generated image sets (after 40000th iteration) to get a robust quality measure. We used the same number of cluster centroids in those experiment, which we had already tested in the clustering analysis. In case we found a strong correlation between the cluster order and GAN performance we would have been able to analyze why such a correlation exists.

We plotted the performance metrics for each cluster number that we experimented with and the complete results can be seen in 4.2. Looking at the IS plot it seems quite clear that there is no linear correlation between the number of clusters and IS performance. It turns out that an optimal value for best performing GAN measured by IS is from a range and not one clear setup. We suspected that such affect might be due to an optimal number of images within each individual cluster. Following this analysis, it turns out that an optimal range seems to be around 128-512 clusters as the IS was fairly consistent in that range. It was clear from the experiments that the performance dropped significantly in the lower and higher end of the experimental range. The decrease in performance with very low number of clusters potentially came from very large and thus not very semantically consistent image clusters. More diversity within the same image class would mean that it is harder for the GAN to learn a good semantic representation and would therefore hurt performance. The degraded performance in the higher end of the cluster number is due to the number of images being too small for each cluster label. Even though the images are semantically very close, the low sample number makes it harder for the GAN to learn the specific cluster features.

We also measured the GAN performance similarly using the FID metric. Similar results for each cluster centroid number are given in 4.2. Looking at the plot, it is clear that FID scores look very different compared to IS performance metric for the same number of clusters. In the FID experiments we saw a very clear correlation between having a higher number of clusters degrading GAN performance. The optimal number



(a) FID score

(b) Inception score

Figure 4.2: IS and FID mean results for different K values

using FID metric was clearly 10 which happens to corresponds to the number of ground truth object classes in the data set. It was not obvious why because the clusters were calculated using a pre-trained net and using closer manual inspection the synthetic clusters did not look similar with the human created ones.

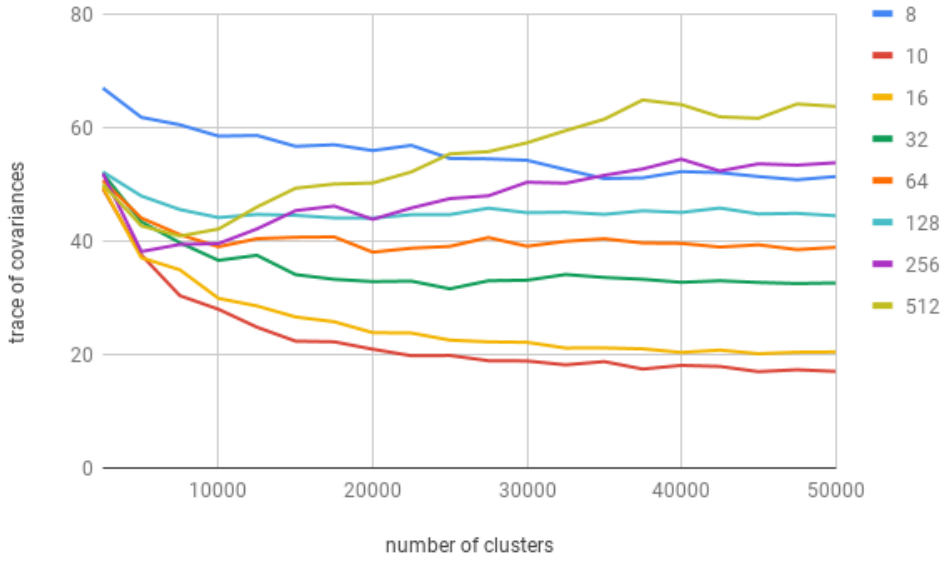
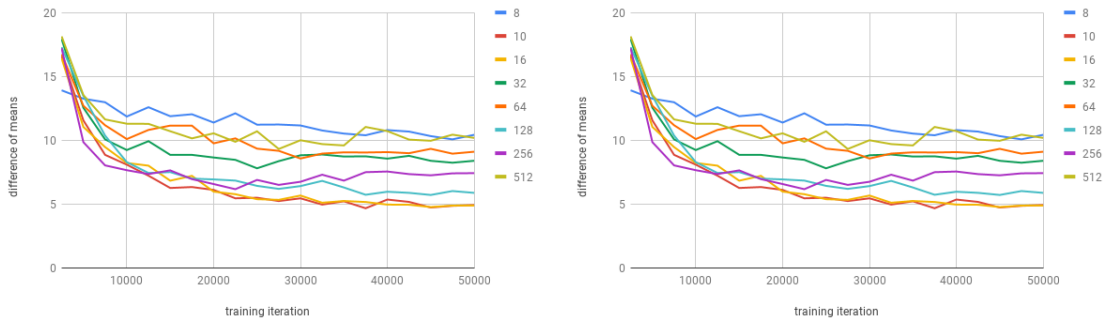


Figure 4.3: FID progression during the training

Comparing the two plots, the biggest surprise was the FID not corresponding with the inception scores for the same K values. In theory, both metrics are supposed to measure the photo realism of data sets and therefore we would have also expected similar results for both metrics. To understand what was really happening during the training process we plotted the FID progression through the training process in 4.3. Looking at the graph it became clear that for larger values of K, the FID performance started decreasing as training progressed. This was very abnormal and we tried to come up with additional metrics that would help us understand better, how the GAN was reacting to different cluster values during training.

4.1.3 Analyzing GAN performance using new methods

By now it was clear that GAN behaves very uniquely for different values of K. To understand better why this might be the case we proposed to look at both the image quality and data set diversity separately. To do that we separated both FID metric components and conducted experiments to measure both separately: the difference of the mean and the trace of the covariances of the data sets. After running the new experiments, we realized looking at the 4.4 that the difference of the means of original data set and the generated one improved throughout the training process for different cluster values, just as we expected should happen. The difference of the means measures the difference of the class activation value means and as the generated images get more defined they should look more similar to the original ones. The bigger surprise came from looking at the trace of the covariance plot shown at 4.4. This metric describes the distribution diversity difference across all individual classes. From the plot it is clear that the diversity keeps getting worse for the generated images as the training progresses. This effect can be seen for all clustering setups with higher cluster numbers than 10. This suggests that the generated images within the same semantic class are more similar. This GAN effect is also knows as mode collapse and it basically shows that for larger values of K the GAN learns to over fit the data and becomes less able to generalize for each individual feature.

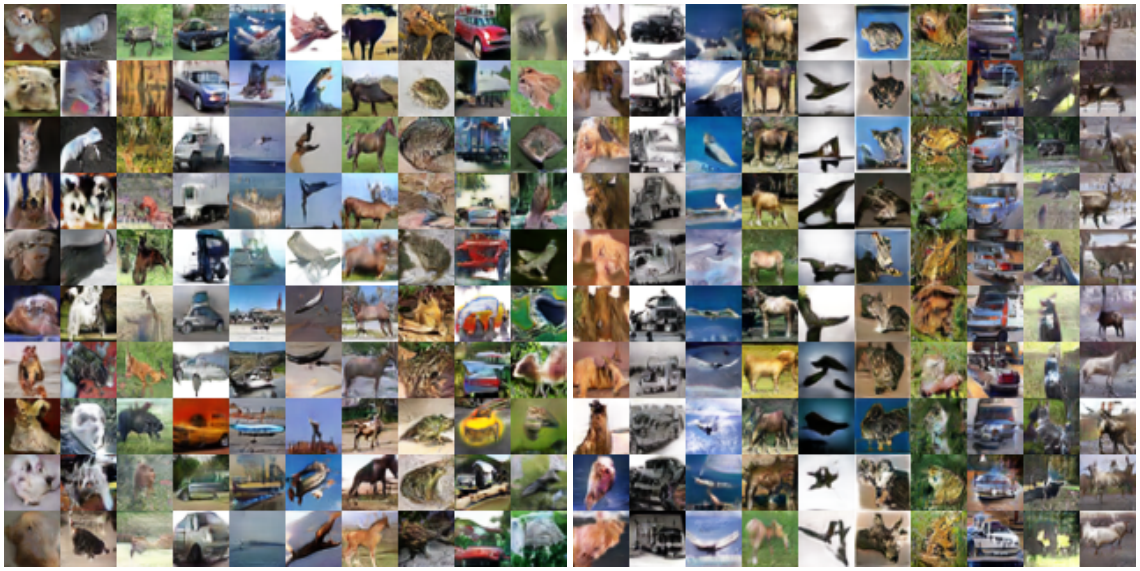


(a) Trace of the covariances

(b) Difference of the distributions' means

Figure 4.4: Measuring image quality and diversity for different K values

To make sure that the plots were not misleading we also manually checked images from within the same cluster. We assumed that using higher values of K the images within the same cluster were supposed to look much more similar to each other. An example can be seen in 4.5 for the K=64 training setup. It is clear that the samples were very similar within the same class and it proved that for larger values of K the GAN seems to over fit for each individual class. Samples generated using the best performing GAN setup can be seen in the K=10 figure. There the samples are much more diverse both in color and shape within the same class. It shows that this particular training setup helps the GAN generate better quality images while not sacrificing data set diversity, which was our main goal.



(a) Generated samples for K=10 (b) Generated samples for K=64

Figure 4.5: Generated samples

Chapter 5

Discussion

In this chapter we go more deeply into the experimental results and give reasons why the experiments turned out the way they did.

5.1 Clustering results

The clustering experiment showed that there is no MSE elbow point when clustering images based on their ImageNet features with different number of clusters. This can be largely due to the fact that the clustering was done with a very large feature set. With a large number of features, big drops in MSE are much more unlikely. The indecisive clustering results pushed us to experiment further with GAN training. Doing the same experiment using VGG16 features gave very similar results, which does prove the generality of the method. It shows that one could use different pre-trained networks given that they have been trained on a diverse enough training set. It is possible that the performance of our method would decrease when the semantic classes in the pre-trained net are completely different from the data set which is used for calculating the activation values. Using visual inspection of the image clusters it was clear that when using larger number of classes the resulting groups get more and more similar semantically. Thus when using activation values from 1008 classes on data that originates from 10 classes, the resulting groups are similar in both semantic meaning, but also often in terms of color and background.

5.2 GAN training results

The results from GAN experiments helped us prove most of our hypothesis and also guided us towards proposing different alternative methods for generating either best quality images or more diverse data sets.

5.2.1 Overall effect of using CIN-GAN

While it had been shown before that conditional batchnorm layers help imropove the performance the results proved that the CIN layers helped the GAN generate even higher quality images than other conditional GANs. The quality improvement that we achieved using a conditional GAN was strongly noticeable versus using a non-conditional architecture5.1. We are able to report average IS of 9.00 using 512 synthetic cluster centers and average FID score 17.29 using 10 clusters. Both results are highest reported scores for CIFAR-10 data set as far as we are aware. our CIN-GAN also performed better as previously proposed iWGAN-AC[15], which uses a very similar technique for generating synthetic labels, but a different method for using

Table 5.1: FID and IS comparison between our CIN-GAN and other recent popular GAN architectures trained on unsupervised images on the CIFAR-10 data set. Our best achieved IS training run used K=10 setup and our best FID training result used K=512 setup.

Method	IS	FID
BEGAN[3]	5.62	84.0
DCGAN[14]	6.16	34.3
WGAN-GP[7]	7.86	28.2
iWGAN-AC[15]	8.62	21.3
Progressive[11]	8.80	19.1
CIN-GAN (Ours)	9.00	17.29

label information. The instance normalization layers are best used for class conditioning in GANs that are used for generating images. They help to normalize each image activation value separately towards the designated class whereas the classical batch normalization forces the entire batch to have the designated class bias in the activation values. We are not sure, whether there would be a similar effect on other types of data. A sample of images generating using our best setup can be seen 4.5. An overview of how our GAN performs against other previous works using unsupervised data is seen in table 5.1. It is very likely that when combining our training methds with other non-related GAN improvement features, the results could be even better.

5.2.2 Method for achieving best image quality

Both visual inspection and evaluation with inception score proved that the best image quality is achieved within the 128-512 cluster number range for the CIFAR-10 data set. We believe this can be generalized using the average number of samples in the resulting clusters. In our setup the number of samples per cluster was between 480 and 120 and we assume that a similar selection for cluster number for different data sets would help achieve the best synthetic labels. This is mainly due to too small clusters not having enough data to help the GAN learn the general on the cluster features and too large clusters being detrimental due to the clusters containing too many different features and making it hard for the GAN to learn a good representation of the average sample.

5.2.3 Method for achieving best image diversity

We measured image quality using one component of the FID metric and it turned out that for data set diversity keeping the original data set ground truth class number is best. For our experiment it meant creating the synthetic labels using 10 cluster centers. Such a setup allowed us to create fairly high quality images and also keep the original diversity of the image set. It was clear that this was the only setup where the image quality and data set diversity improved throughout the training process. For any other setup the diversity decreased with the training process. After visual inspection of the generated images it was clear that using larger clusters numbers encouraged GAN mode collapse. The generated image examples for bigger cluster center values were very similar within the same class and the GAN was not able to generalize within the same classes any more. We believe the mode collapse happened due to clusters having too few images within themselves for the GAN to learn to generalize the classes well and thus creating almost identical copies of the same object for each class.

5.3 Shortcomings in the current evaluation methods

During our experiments we used both current most popular GAN evaluation metrics and both were initially very misleading. Both Inception score and FID report metrics that by themselves are not very informative. During our initial experiments we were achieving excellent results while not realizing that there was actually a mode collapse during training. One such example can be seen here 4.5. Also, when comparing the Inception score and FID during similar runs the values differed significantly - there was no correlation between the two even though both are supposed to measure photo realism of images.

A further problem with FID is that one does not know how to improve the GAN when only looking at the score during the training process. An improved FID does not give specific information about the quality nor the distribution. To optimize the GAN one would still have to look at the generated images to see what is actually happening and this approach can be very time consuming. Also, looking at the generated training samples gives a very bad overview of how the GAN is able to generalize for the training set since it is intractable to measure large data set diversity using manual inspection without looking at all the samples.

Final problem which we noticed when evaluating our data sets was the quality of the diversity metric. The second component of FID is the trace of the covariances of the two data set activation values. Looking at the trace of the covariance matrix means comparing the variances in each class activation values. For example, if the activation value variance of class A is high in the generated data set, but low in the original data set, the trace value would be high and increase the FID (thus show decreased the performance). Since GANs are supposed to be able to combine different features in the latent space, we would prefer the generated data set to have larger diversity than the original one (combining unique features in all potential combinations would result in more diversity). Generating more such unique samples would result in larger variance than the original data set which would thus decrease FID performance, which is an unwanted effect.

A potential additional unrelated method for analyzing image data set quality can be achieved by clustering the generated data using the same cluster centroid locations as achieved in from clustering the original data. Comparing the statistics of the two data sets we noticed similarities in the cluster features, but more analysis should be done in that area to be able to make stronger claims.

Chapter 6

Conclusion

With this work we successfully proved several important hypotheses we initially raised and also pointed several shortcomings with some widely used techniques in the adversarial network research field:

- We were able to connect the number of samples per cluster for generating best synthetic labels for a given image data set in order to generate most photo realistic samples.
- We proved that using our CIN-GAN setup we were able to achieve better results than both non-conditional other conditional GAN architecture have shown.
- We were able to report state-of-the-art Inception score and FID using different values of cluster centers for generating synthetic labels.
- We showed how different metrics can be used to better analyze and debug the GAN training process without having to manual inspection.
- We discussed major shortcomings in both Inception score and FID and proposed a method how to work around them.

Conditional adversarial networks have proven both to be better performing and more controllable for generating samples based desired features than non-conditional networks. Without having a conditional generator it is much harder to control the semantic output of the generator. Despite that they are not as widely used as they could be due to requiring labelled data for training. With this work we proved that using no extra resources it is possible to achieve similar performance to using supervised training process with human provided labels. This research area seems very promising and we hope to see further progress in conditional GANs.
Our work also showed that currently the most popular metrics for measuring GAN performance are lacking. It is challenging to measure photo realism using fully automated methods, but it is clear that the current metrics still need improvement.
There is still lots of room for improvement in both increasing the quality of the generated images and also making adversarial networks more easily explainable. In general, there has been little work in reasoning and proving adversarial network properties. Some potential further research areas would include:

- Developing a joint learning between the clustering and the GAN training process that that information from the unsupervised clustering process could be used in the GAN training to achieve even better results.

- A deeper look into image quality metrics to come up with even better methods for measuring image quality and diversity
- Develop a more advanced GAN loss function which would directly optimize for image quality and diversity during the training process
- Making GANs even more controllable by adding additional conditions so that it would be possible to control every semantic feature of the generator output image.

Most of the research in the adversarial network domain is focusing on improving the quality of the output. While the quality is indeed very important, so far there has been little research in making such networks more predictable. It is most likely due to the same fact that adversarial networks have seen little practice outside of research field. We hope that further research will help make adversarial networks more practical.
We thank the reader for their attention and are open for further discussion about our work and further research potential in this area.

Appendix A

Python code for conditional instance normalization layer

Listing A.1: Conditional Instance normalization implementation in python

```
class CondInstanceNorm(_BatchNorm):
    def __init__(self, num_features, eps=1e-5, momentum=0.1,
        affine=False, track_running_stats=False, labels=None):
        super(CondInstanceNorm, self).__init__(
            num_features, eps, momentum, affine, track_running_stats)
        self.labels = labels

    def _check_input_dim(self, input):
        if input.dim() != 4:
            raise ValueError('expected 4D input (got {}D input)'
                             .format(input.dim()))

    def _load_from_state_dict(self, state_dict, prefix, strict,
        missing_keys, unexpected_keys, error_msgs):
        try:
            version = state_dict._metadata[prefix[:-1]]["version"]
        except (AttributeError, KeyError):
            version = None
        if version is None and not self.track_running_stats:
            running_stats_keys = []
            for name in ('running_mean', 'running_var'):
                key = prefix + name
                if key in state_dict:
                    running_stats_keys.append(key)
            if len(running_stats_keys) > 0:
                for key in running_stats_keys:
                    state_dict.pop(key)

        super(CondInstanceNorm, self)._load_from_state_dict(
            state_dict, prefix, strict, missing_keys,
            unexpected_keys, error_msgs)
```

```
    def forward(self, layer_input):
        self._check_input_dim(layer_input)
        n_labels = self.labels.shape[0]
        label_dim = self.running_mean.shape[1]
        offset = torch.zeros([n_labels, label_dim], dtype=torch.float32)
        scale = torch.ones([n_labels, label_dim], dtype=torch.float32)
        weight = offset[labels][:,:,None,None]
        bias = scale[labels][:,:,None,None]
        return F.instance_norm(
            layer_input, self.running_mean, self.running_var, weight, bias,
            self.training or not self.track_running_stats,
            self.momentum, self.eps)
```

Appendix B

Python code for calculating difference of means and variances

Listing B.1: Mean difference and variance metric calculation implementation in python

```
def calc_mean_and_variance(stats_path, inception_path, samples, sess):
  npz_file = np.load(stats_path)
  mu_real, sigma_real = npz_file['mu'][:], npz_file['sigma'][:]
  f.close()
  fid.create_inception_graph(inception_path)  # load the graph into the current TF graph
  mu_gen, sigma_gen = fid.calculate_activation_statistics(samples, sess, batch_size=1, verb
  mean_diff = np.sum((mu_gen - mu_real) ** 2)
  trace = np.trace(sigma_gen + sigma_real - 2.0 * scipy.linalg.sqrtm(np.dot(sigma_gen, sigm
  return np.real(mean_diff), np.real(trace)
```

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] David Berthelot, Thomas Schumm, and Luke Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[4] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[12] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[13] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[15] Alexander Sage, Eirikur Agustsson, Radu Timofte, Luc Van Gool, and KU ESAT. Logo synthesis and manipulation with clustered generative adversarial networks. *arXiv preprint arXiv:1712.04407*, 2017.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor S Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, volume 1, page 3, 2017.

[18] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

A PERFORMANCE ANALYSIS OF THE GENERATIVE ADVERSARIAL NETWORKS

Authored by (in block letters):

For papers written by groups the names of all authors are required

Name(s):	**First name(s):**
RIESENKAMPF	HEIKI

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 01/08/2018	Riesenkampf

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.