Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

1st Nicolas Beltran
Department of Computer Science
Columbia University
New York City, United States of America
nb2838

***Abstract*—Generative adversarial networks have been touted as one of the most exciting ideas in deep learning during the past decade. They provide a innovative framework for building generative models and unsupervised feature extractors. One of the most succeful variants of the model are DCGANs which where introduced in 2016 by Alec Radford and Luke Metz [1] as an alternative to regular GANs. Their main innovation lies in using Transpose Convolutional layers in the generator to obtain more realistic images. My project attempts to replicate the results in the "Paper Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." In particular I focus on 1.Reproducing the results pertaining to the face daset. 2. Implementing a feature extraction architecture trained on CIFAR-10 for later classification. This project shows that although DCGANS solve various of the issues that plague regular GAN models, a lot of additional modifications are still required to achieve stable training, specially in low data settings.**

***Index Terms*—DCGAN,Adverserial Learning, Generative Models, Neural Networks.**

I. Introduction

Generative adversarial networks (GANs) are a model proposed by Goodfellow et al [4] based on a game-theoretic scenario in which two neural networks are pitted against each other with the objective of training a generative model. These two neural networks denoted by G and D have opposing tasks. G, often called the generator, attempts to create realistic samples of the problem at hand. D, often caled the discriminator, attempts to distinguish samples generated by G from those obatined from the real distribution.

By themselves GANs are challenging to train, unstable and often times don't yield good results. For computer vision applications an improvment over vanilla GANS are DCGANs which combine the GAN game-theoretic formulation alongside convolutional neural networks to improve the capabilities of model for image generation and feature extraction [1]. My project provides a reproduction of the results provided in the paper Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Neural Networks, which was the original paper where DCGANs were proposed. In particular, I will reproduce two of the main results. First, I will show that DCGANS can succesfully be used as generative model to produce faces. Second, I will show that DCGANS can be used as a feature extractor in low-data regime situations by training a DCGAN on the CIFAR-10 dataset and then using the features extracted to train an SVM classifier with better performance than one based solely on pixel-features. I will show however that the performance is limited compared to training on a larger dataset.

A. Formal definition of the model

Following the Goodfellow et al. [4] we use the following definitions. We let p_g denote the distribution over data x, we define a prior on the input latent space denoted by $p_z(z)$, usually taken to be a standard normal distribution, and represent the generator (the mapping into the data space) by $G(z;\theta_g)$, where G is neural network with parameters θ_g. Similarly, we represent the discriminator as $D(x;\theta_d)$.

We then assume that G and D play the following zero-sum minimax game

$$\min_G \max_D E_{x\sim p_{\text{data}}}[\log(D(x))] + E_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

Often, as we do in this implementation, we replace $\log(1 - D(G(z))$ by $-\log(D(G(z))$. This is beneficial because it helps with gradient convergence and is suggested by Goodfellow [4]. The Nash equilibrium for this game, the point where no player has an incentive to change its actions, is attained at the point where the generator can produce images that are just like the real ones, and the discriminator can do nothing other than randomly guess. Additionally, we see that from the model definition above, to generate new images one simply needs to sample from the latent space and then map the sampled vector to the data space by using the generator. This is the strategy I will use later in the paper.

B. Training

To train GANs one usually follows a series of repeated steps in which the discriminator is first updated and then generator is updated. In particular, the vanilla algorithm by which GANs can be trained is provided by Goodfellow et al [4]. As usual, some form of gradient descent and ascent are used to the minimize (or maximize depending on the player) the loss. However, unlike for regular models, due to the adversarial nature of the training process, the loss for D and for G osccilates and does not tend to a minimum. The formal description of this algorithm is provided below (the figure is reproduced form Goodfellow's paper [4])

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Fig. 1. Algorithm for training GANs [4]

Unfortunately, this training process is seldom smooth. The objective of training a GAN is to find a Nash Equilibrium of a non-convex objective function with high dimensional parameters. This is very challanging and it is very difficult for the GAN to actually converge to such an equilibrium [7]. Instead one might observe situations where the Generative network is not able to provide realistic examples or where the network experiences "mode collapse," which is the phenomenon in which the generative network only learns to produce a very small number of samples. [7]

DCGANs, the model explored in this paper, solve some of those issue by using architectural as well training innovations. In addition to these innovations, there are also tricks that help stabilize the learning. Among those, some that where implemented in this paper are label smoothing [7] in which the training labels for the DCGAN (labels with respect to the crossentropy function) are smoothed, the use of dropout for regularization in the discriminator, and the separation of minibatches. A comprehensive list of these tricks can be found in the repository Gan hacks.

II. Summary of the Original Paper

A. Methodology of the Original Paper

As described above, training is quite difficult which might lead to suboptimal results. Moreover, architectural condisiderations might have a drastic impact on the quality of the results. The paper this project is meant to implement attempts to provide various improvements that help with the training process and the architecture of the Generator. In particular, it made 4 contributions.

- It proposed the DCGAN model, which is a new architecture based on transposed convolutions as opposed to a feedforward architecture.
- It showed that the learned features of the discriminator can later be used as feature extractors in a supervised setting.
- It showed that the filters learned by the GAN have learned to draw specific objects.
- It shows that the generators have intersting arithmetic properties in the latent space.

In my project I focused on objectives 1 ,2 and 3, and therefore I will only focus in summarizing these results.

B. DCGAN Architecture

On the architectural side the paper proses modifying the normal GAN by doing three things. First, they add a convolutional net to the Generator. Second they eliminate fully connected layers. Third they use batch normalization [6] to stabilize training. Finally, they use Relu for the generator and leaky Relu for the disciminator as activation functions. This architecture can then be vizualized with the following diagram.

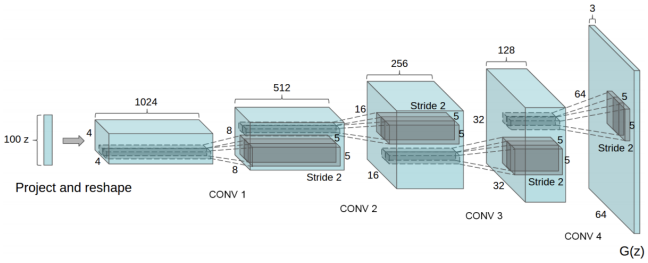

Fig. 2. General DCGAN architecture for Generator

For training this network they proposed using a batch size of 128, normalized images in the range $[-1, 1]$, initializing all weight from a zero centedred normal distribution with standard deviation 0.02, setting the paremeters of the leak to 0.2 and using an Adam optimizer with a learning rate of 0.0002 and a momentum term of $\beta_1 = 0.5$ as opposed to 0.9 as the original paper suggests [3].

C. DCGAN as a Feature Extractor

To test that the model was indeed useful it was trained first on imagenet 1k with the parameters described above and then the discriminators convolutional features where maxpooled and flattened to obtain a vector representation. The trained discriminator was then used to extract features from CIFAR 10 data that was later used to train an SVM. This approach achieved 0.828 accuraccy, which is still less that traditional CNNs but is more than using a K-means based approach. Results of the sample images produced can be found below:

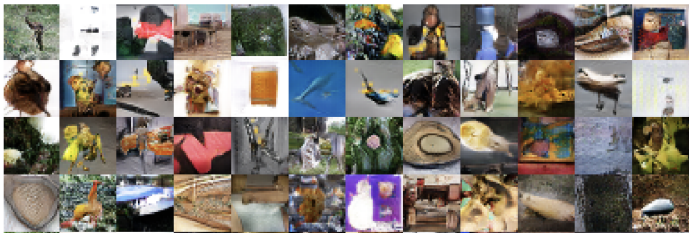

Fig. 3. Generated results from Imagenet

D. Vector arithmetic and other experiments

In addition to testing the capabilities of the neural network to extract features, there were also multiple tests conducted to examine the way in which the latent space was being mapped into the data space. In particular, the paper conducted experiments using a dataset of 3 million faces from 10 thousand people and LSUN bedroom dataset which consists of 3 million photos of bedrooms. The two experiments relevant to this project were the vector algebra and space walking experiment.

Fig. 4. Generated results from Faces dataset

The vector algebra experiment consisted in using simple algebraic operations on the vectors in the latent space and then seeing if the results where semantically meaninful. Instead of using raw vectors, the paper used averages between semantically similar images because this led to more stable results. Figure (4) contains images of the generated faces and figure (5) contains results from performing vector arithmetic on the faces.

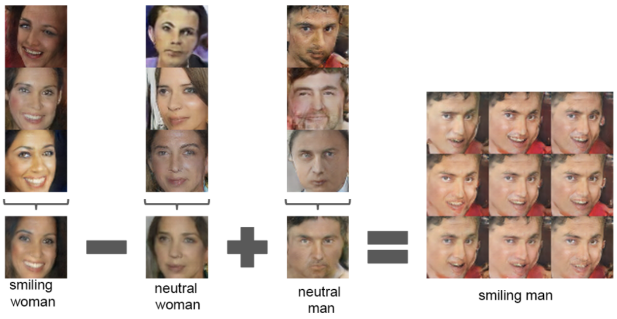

Fig. 5. Arithemetic results on faces

The latent space walking experiment consisted in interpolating between random points in the latent space and observing the pictures these points where mapped to. The objective was to show that the manifold learned was meaningful and there was no memorization. If memorization were present and the manifold was not meaningful then the interpolation woud show sharp changes rather than smooth transitions between both images. The paper performed these experiments with the LSUN dataset. Figure (6) contains some of the results in the paper.

III. Methodology of my Project

With the aim of exploring the results in the paper and understanding the limitations of the DCGAN model, this project replicated the results pertaining to the faces dataset and to the feature extraction capabilities of the model. In this section we formally describe the objectives, how these differ from the paper and the technical challanges these differences imply.

Fig. 6. Walking on the latent space

A. Objectives

- Train a DCGAN on a Faces-like dataset to obtain results of a similar quality of those presented in the paper.
- Perform algebra on vectors in the latent space of the faces data set to verify that the learned manifold contains semantically meaningful representations.
- Verify that learned manifold is not memorizing by walking in the latent space and interpolating between differnt faces.
- Train a DCGAN on small dataset to analyze the performace of the model in a low data regime.
- Train a support vector machine on learned features of the low-data model to asses the quality of the learned representations.

B. Challenges

The challges of these project can be divided in two different categories. Those challanges that are inherent to the training of GANs, and those challenges that are caused due to the particular objectives described above. The challenges associated with GAN training are the following

- Due to the nature of the objective function, it is not possible to monitor progress of the GAN by looking at the loss. Instead training needs to be closesly babysitted by reviewing produced images at training time.
- Depending on the dataset used, the discriminator might become profficient at detecting real versus fake examples faster than the generator acquires the capability of producing good imitations.
- Mode collapse can happen at any point of training. This requires a constant examination of results produced by the network.
- Due to the nature of the loss function used for training it is not as simple to directly asses whether an improvement is better or not.

The challenges associated with the pariticular objectives of the project are described below.

- In a low data regime the model is more likely to be unstable because the diversity of the examples provided is so small that the discriminator can easily overfit.

- Low data regimes might exacerbate the mode collapse problem as the generator doesn't have a lot of examples to learn from.

Later on, when explaining in detail the architectures used we will describe steps that were taken to mitigate these issues.

C. Problem Formulation

To achieve these objectives we will follow the same problem formulation provided in the introduciton to the general GAN framework. The only changes from these descriptions are the architecture of the deep learning framework that we will use, and the latter feature extraction step.

Formally, using the notation establsihed in the introduction we will attempt to find a solution to the minimax game

$$\min_G \max_D E_{x\sim p_{\text{data}}}[\log(D(x))] + E_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

where we will assume that $p(z) \sim \mathcal{N}\left(\vec{0}, I_{100}\right)$ where I_n denotes the idenity matrix of dimension n , p_{data} is replaced by the pariticular data distribution at hand, and G and D are created following the architectural considerations stablished in the DCGAN paper. This is the formulation for the traditional GAN framework as expressed before.

The problem formulation for the feature extraction can then be described as follows. Recall that θ_d and θ_g parametrize the discriminator and generator repespectively (see introduction). Assume that $\mathcal{X}$ is the data space, images from these dataspace belong to l classes and F represents some feature space. Then our unsupervised learning forumation can be described as finding a transformation $T : \mathcal{X} \to F$ parametrized by θ_d such that the accuracy of some learner $\mathcal{A}$ increases when trained on data transormed by T as opposed to trained on $\mathcal{X}$. In our case, the learner A that we will use will be a support vector machine. The particular transformation T that we will be described later when considering network architectures.

Finally, to complete the problem formulation, we need to specify how we will train our algorithms. For solving the minimax problem we will use the training algorithm described in the introducion with an Adam optimizer, and for training the support vector machine we will use vanilla SGD.

D. Datasets

To train the models we used two datasets

- **Celeba**: The Celeba dataset [8] was used as a replacement for the faces dataset found in the original paper. It consists of 202,599 face images. Each image was rescaled to be 64×64 size and no augmentations were performed nor additional croppings. As opposed to the original paper, the background was visible and for some photos it is possible to see a small part of the body. All of the dataset was used for training. Below is a sample of the images used
- **CIFAR 10**: For the experiments related to the unsupervised feature extraction I decided to use CIFAR 10 [5] as opposed to Imagenet as was done in the paper. The dataset consists only of 60,000 images divided into ten categories. This dataset is significanlty smaller than the one used in the paper which contained over a million images. Nevertheless, using a small dataset allows us to check if using a large dataset is neccesary to obtain good results or if this is only a characteristic of the architecture. Sample images from this dataset are provided below. No cropping or data augmentation was performed to this dataset. The whole dataset was used for training the GAN but only 50,000 images where used to train the svm. The rest was used as a testing dataset.

Fig. 7. Celeba Dataset Sample Images

E. Design of Project

Having already explained exactly how we will proceed I provide below a flowchart of the process of how the experiments will be conducted. An important point to be made is that although the word hyper-parameter suggests that we will only be modifying things such as learning rate and similar parameters, due to the complexity of training a GAN, we will also be modifying more complex parameters such as the point where the gradient is calculated, degree of smoothness for labels, quantity of dropout and number of droput layers.

IV. Implementation

The implemntation of my neural network followed very closely that of the paper. A careful attempt was made to replicate their architecture as much as possible when guidance was provided. However, specially for the second experiment, it was neccesary to deviate from the architecture that was proposed because 1. There was not enough guidance in the paper 2. The network was unstable and the discriminator got to good to quickly which led to poor results. Below I provide

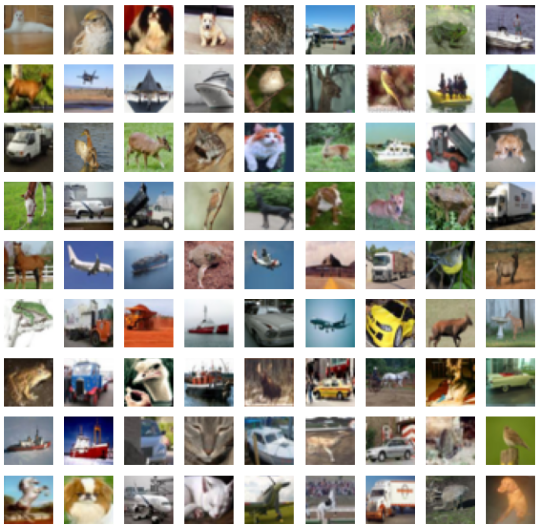

Fig. 8. Celeba Dataset Sample Images

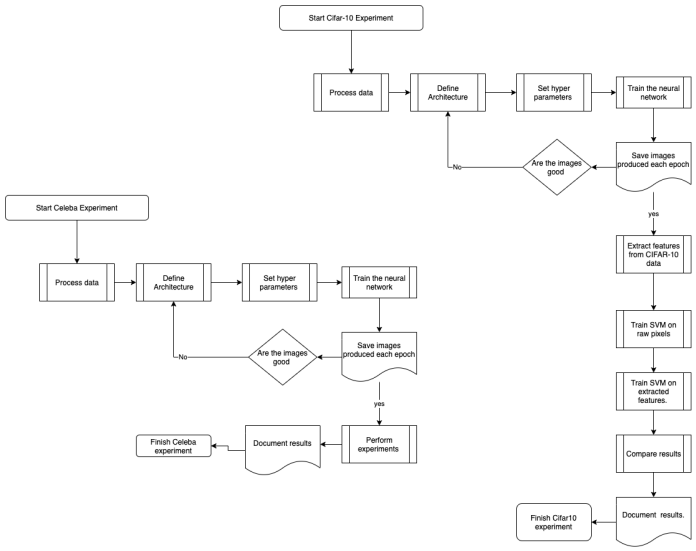

Fig. 9. Experiment flowchart

a careful explanation of the architectural desicions made and diagrams of the network used. Additionally, I will provide a careful description of the effots that were made to optimize the neural network alongside an explanataion of why this desicions were taken.

A. Celeba network and training

1) Network: The generator network for the Celeba experiments followed that of the paper. In the convolutional layers I added strides only so that the dimensions matched those provided in the paper. For the discriminator network I used a LeNet inspired architecture but getting of most dense layers and without using pooling operations. Block diagrams of the architectures are provided below. Some important notes are the following.

- The weight initialization scheme followed closely that of the paper. Every weight in both networks was initialized to a standard normal distribution with mean 0 and standard deviation 0.02.
- Instead of using ReLU activations for the generator I decided to use LeakyReLU as this proved to have a beneficial effect in terms of convergence of the algorithm.
- I used batch normalization after every convolution as described by the paper.
- In the last layer I use Tanh so that the activations match the normalized range of images.

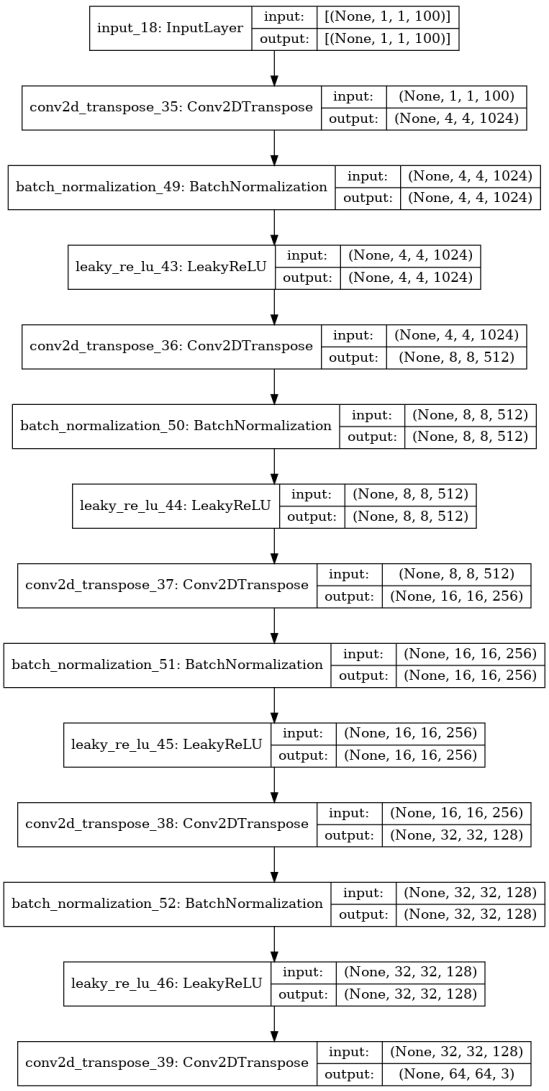

Fig. 10. Celeba generator

2) Training: For the training algorithm we follow the pseudocode in figure 1. This is the same algorithm generally

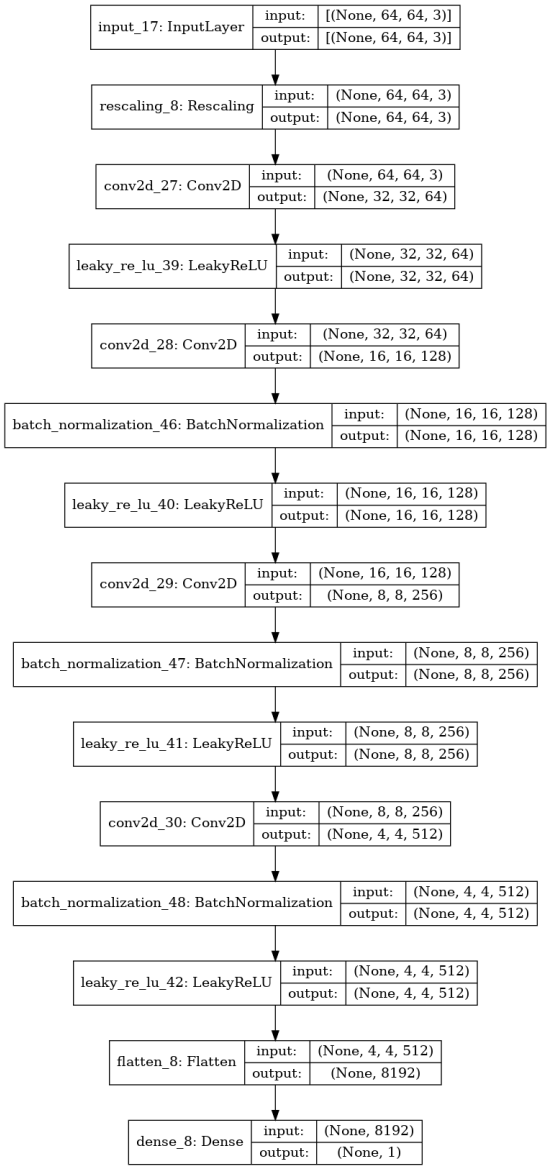

Fig. 11. Celeba discriminator

used for GANs. Some of the modifications we make are the following:

- I used $\log(D(G(z))$ instead of $\log(1 - D(G(z))$ as suggested by goodfellow. This means that in the last part of the algorithm we replace the descending step by an ascending step with the new loss.
- I used the hyper paramters provided by paper and an Adam optimizer as suggested by the paper. This included setting the learning rate to 0.0002 instead of 0.001 and I changed the momentum term to $\beta_1 = 0.5$ instead of 0.9 as is suggested by [3].
- I used a batch size of 128 as suggested by the paper.
- I trained for 25 epochs which took approximately 10 hours. I monitored the progress of the network closely by constantly checking the images provided by the network. I choose to train for this number of epochs because it allowed me to train the network architecture and try different optimization schemes. However, none seem to yield results better than the one described above.

The training algorithm and implementation can be found in this file

B. Cifar-10 network and training

In this section we provide a description of the neural network architecture and the training algorithm as used for the CIFAR 10 dataset. In addition, I report my efforts to optimize the neural network architecture.

1) Network : The neural network architecture is very similar to the one detailed above. The differences can be categorized in three ways: efforts to regularize the discriminator, differences due to image sizes and differences to extract features. I will explain these categories below. In addition to these modifications I decided to use ReLU instead of Leaky ReLU because in tests it provided better images.

- **Image Sizes**: Due to varying image sizes I removed the last layer of the generator and adjusted activations accordingly so that images of size 32 x 32 would be produced.
- **Feature Extraction**: To extract features I added a global max pooling layer on top of every convolutional layer in the discriminator. This was done in parallel to the architecture described before. The original paper did not provide a clear guidance on how to extract features so I chose this approach as a combination of what was suggested in the paper and what was explored in class.
- **Regularization**: The main issue that I faced during training was that the generator quickly became too good and training quickly became unstable. This severely impacted the quality of training. To combat this issue I reduced the depth of the Discriminator to decrease its capacity and added dropout layers. I found that adding one dropout layer at the end of the network and one dropout layer inside of the network reduced the capacity enough to allow the generator to learn.
  In addition to my final solution. I also attempted to add one dropout layer, to add dropout to the input variables, and to provide no regularization. Figures (15) (16) (14) contain the results of these attempts after 10 epochs of training which justify the changes I made. The pictures correspond to no dropout, input dropout, last layer dropout respectively. Using only one layer of dropout worked well but after 30 epochs the results deteriorated very quickly. The training time for the CIFAR dataset was of about 5 hours for a 100 epochs which allowed for more iterations over the training process.

2) Training: As explained above, the main difficulty for training the network was that the generator quickly was able to differentiate very well between fake and real pictures. Althought the modification of the architecture detailed above heleped with this difficulty some additional measures were taken which slightly modified the training procedure used for

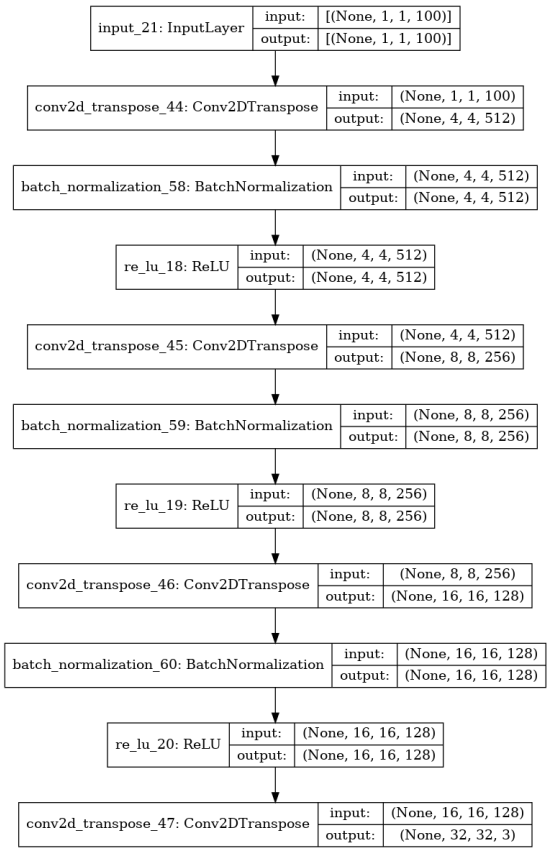

Fig. 12. Cifar 10 generator

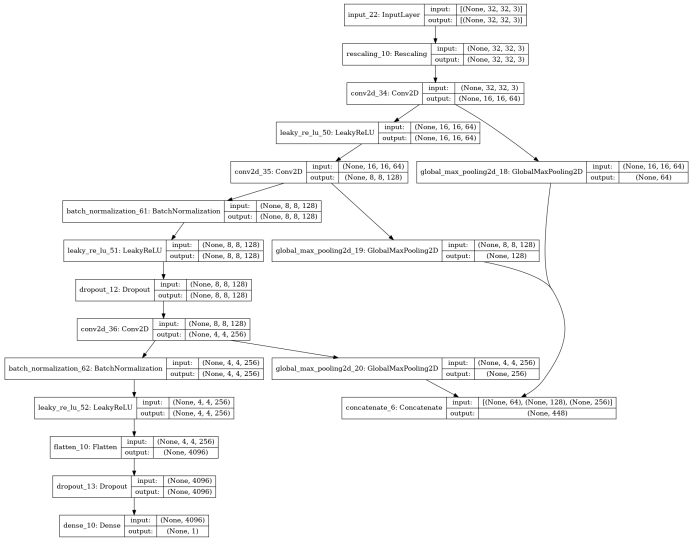

Fig. 13. Cifar 10 discriminator

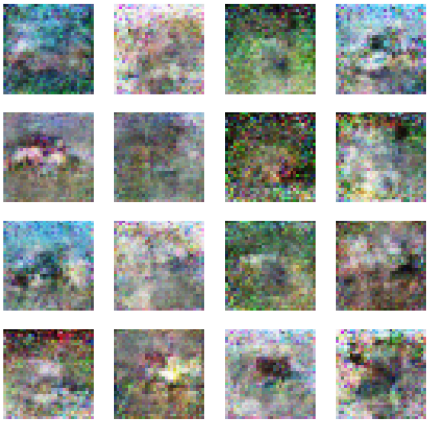

Fig. 14. No dropout

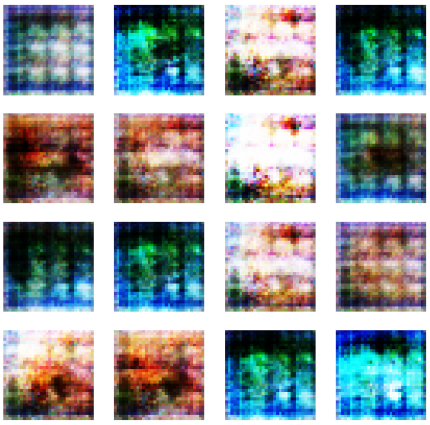

Fig. 15. Dropout at the input

the Celeba dataset. I will explain what these modifications were. Besides these changes the training algorithm was the same:

- Label Smoothing: I used label smoothing for the samples provided to the discriminator. This means that rather than providing a label of 1 or 0 for real and fake examples I used a random value between $0.7 - 1.2$ and $0 - 0.3$. Different values were tried and there didn't seem to be any significant difference between changing this range much. However, using Label Smoothing improved the quality of the images significantly.
- I calculated the gradients for the left and right term of the loss function independently as this was suggested as a way of gaining stability in the Gan hacks document referenced before. This helped only slightly with the

Fig. 16. Dropout only at the end

training. I attempted to also update only at the end of training but this decreased the quality of the updates.

- I attempted to use different optimizers for the generator and the discriminator. In particular, I attempted to use SGD with momentum for the discriminator and to use Adam for the generator. The rationale was that doing so would slow the improvement of the discriminator so that the generator could catch up. Unfortunately this wasn't the case so I didn't add this modification.

Besides these changes everything was done in the same way as in the Celeba dataset. Because of the reduced size of the dataset I was able to train for up to 100 epochs instead of 20. However the results didn't improve for more than 50 epochs so I present those below. After 50 epochs there was a drastic mode collapse. Additionally, it is important to note that the outputs for the feature extractino were not used in any way during the training process. They were simply ignored and therefore didn't affect the calculation of the gradients. The code for the implementation can be found in this file.

C. Software design

Given that the project didn't require the interaction of many elements, the software design of my project was simple. It consists only of the models, plus custom training algorithms that have been described above. With the aim of being conscise I don't repeat these descriptions here. Additionally, links to the relevant code in Github can be found above. Everything was done using Tensorflow version 2.4 (the change in version was simply due a clash in my machine), keras and scikit-learn for the SVMs. Given that there were various modifications to the training loop that needed to be implemented, I used the direct autograd capabilities of Tensorflow rather than the premade keras fit methods.

V. Results

A. Celeba Image Generation

As can be observed from the in figure (17), the results are comparable to those obtained by the authors. An inportant difference is that because the images that we provided had, hair, background and body the generator learned to produce this as well. Similarly, because the genrator has to focus on more things, sometimes it learns to produce images of comparable but slightly lower quality

Fig. 17. My celeb a generated images

B. Algebra in Latent Space

The images below, show that the generator for the Celeba dataset is able to learn semantically meaningful representations. These are only some of the experiments I conducted. In figure (18) we see for example that (brown hair woman)-(brown hair man) = (blonde woman). I attempted to reproduce the results in the paper exactly but unfortunately because the representations now contain more information that just the face it was harder to get a one to one comparison. For example, now the representations need to encode things such as, length of hair, background, type of shirt (etc). This entanglement of the representations makes it difficult to perform the same type of algebra calculations. The interpretations are still possible as the experiment shows.

An important note is that to improve stability we use averages over various latent space images.

Fig. 18. Arithmetic example in latent space

C. Smooth transitions in Latent Space

The pictures below show the transition from a between two points in the latent space. We see that the transition between these two points is coherent and smooth. This means that every vector in the transition represents a valid person. Moreover, we can see that there are no abrupt changes. Instead the color and background of the woman's hair changes seamelesly from one to the next. This further supports the hypothesis posited above above the vectors encoding much more than just facial features.

Fig. 19. Interpolation between vectors in the latent space

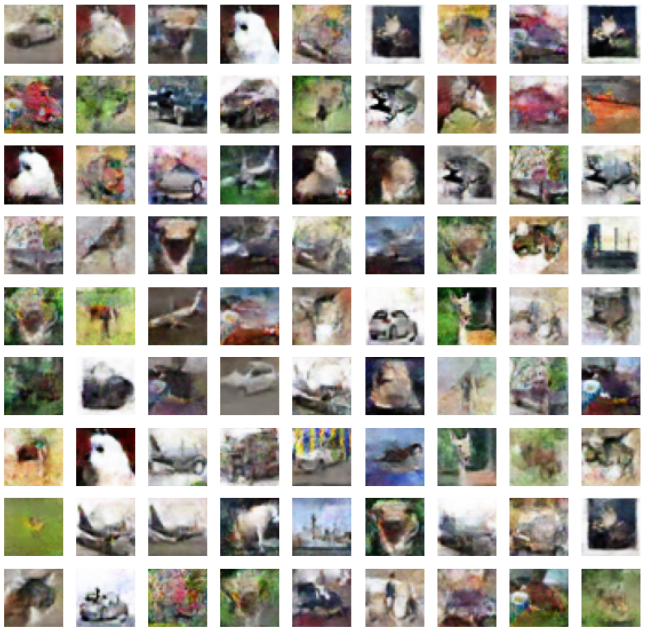

Fig. 20. Cifar 10 generated images by my model

D. Cifar-10 Images generated

Below I show the results of the images from the Cifar 10 dataset. As we can see there are some pictures that are distinguishable. Among them it is possible to recognize dogs, cars, boats, planes and horses. However, the quality is not as good as those produced by the paper. These results were after several rounds of improvements and modifications to the neural network architecture. These changes are the ones described above. It is very likely that the low quality is simply due to the small amount of images compared to those in Imagenet.

Additionally, it is evident that some of the images are very similar to each other which indicates some level of mode collapse.

E. Cifar-10 Feature Extraction

After extracting the features we trained an SVM classifier on the CIFAR-10 datset. The classifier was trained using SGD as other training methods were too slow. The results however were signficantly worse than the paper. Although I managed to construct a classifier that improved performance over raw pixel data by approximately 30 percent, it still was not close to the accuracy reported in the paper. The values I obtained were an accuracy of 0.49222 for the classifier trained on learned features and an accuracy of 0.386 for the classifier trained on the raw pixels. Figures (21) and (22) contain plots of the confusion matrices for both classifiers. An effort was made to optimize these results using cross validation but no values yielded a significant improvement.

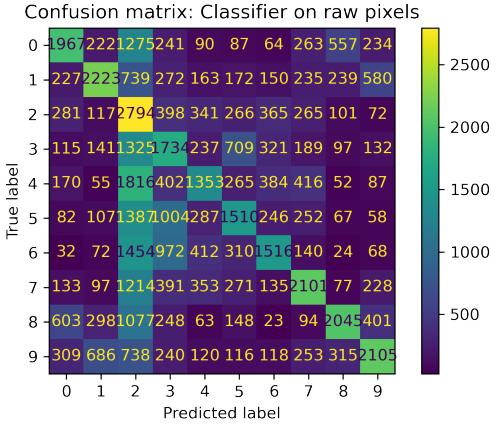

Fig. 21. Confusion matrix of SVM on trained on raw pixels

VI. Discussion of Results: Comparison to the paper and Insights gained

From the figures presented above it is evident that when the paper was closely followed the results are significantly better than those that were obtained when it wasn't. This is probably due to the fact that in the low data regime the generator is not able to learn features that are as meaningful as those in a high data regime. Additionally, it is also probable that reducing the capacity of the network as was neccesary in this case further

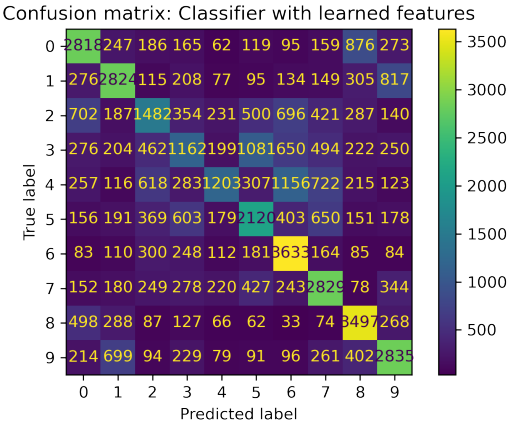

Fig. 22. Confusion matrix of SVM on trained on learned features

aggravates the situation.

In the case of the Celeba dataset then it is clear that all of the experiments were succesful in the sense that the results from my model matched closely those of the paper. However, in the case of the cifar 10 experiment I believe that the results are only partially sucessful as they show that it is possible to learn a better feature representation using the DCGAN model, but this representaion is not superior to the traditional methods explained in the paper.

A. Insights from Celeba experiments

Although we were able to show that it is possible to reproduce the results in paper an important insight that was not originally in the latter is that representations get tangled as more information gets added. In other words, when we performed the interpolation exercises it is clear the transition included not only features such as facial shape and hair color, but also background, lighting and texture. This makes it harder to perform algebraic manipulations because each picture doesn't have such a clear cut interpretation. It is likely that the paper omitted algebraic manipulations with the other datasets precisely because of this issue.

B. Insights from Cifar experiments

The cifar experiment didn't perform nearly as well as in the paper but this is probably because the training conditions were very different. In the paper, a larger network and more images were used. In my project, given that I decided to train on a smaller number of images, I had to reduce the number capacity of the discriminator to avoid serious instability. An SVM trained in this way yieled an accuracy of 0.49 while the paper managed to achieve an accuracy of of 0.82. This shows that large quantities of data are required.

Additionally, although the results were unsucessful in terms of the learned features they were succesful in terms of generated images. However, various changes needed to be made to the model. In particular, it was neccesary to add dropout, reduce the capacity of the discriminator, add smooth labels, and alter the traing strategy to compute the gradients in two steps. This shows that GAN training is still unstable even if a DCGAN is used. As a possible future direcition, I believe it would be interesting to attempt the same experiments with auxiliary labels and feature matching [7]. In the paper where these techinques are proposed they managed to obtain better image generation. Attempting this experiment would be good because we would able to confirm that the low quality of the features is related to the quality of the image produced.

VII. Conclusions

DCGANS provide a useful and important alterantive to traditional GANs by producing higher quality images. The manifold learned provides useful and meaningful representations that can be used to understand the generation process. Additionally, the discriminator can be used as a method for feature extraction in unsupervised settings. However, the quality of the representations becomes entangled the more information we add to the image and the quality of the features extracted diminishes as the amount of data is reduced. Although DCGANS are in many ways an improvement over traidional GANs, in various situations it is still very important to use "tricks" that further help with stability during training.

Workload division

All of the work in this project was done by Nicolas Beltran.

Acknowledgment

I would like to thank the creators of Tensorflow for providing the software used for the class, to Juan David Sandoval Valencia for giving me encouragement to try this paper and to Mora for being artistically interested in the pictures the model was producing.

References

[1] A. Radford, L. Metz, and S.Chintala, "Unusupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv:1511.06434U [cs.LG], Jan. 2016

[2] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images", 2009.

[3] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980 [cs.LG], Dec 2014.

[4] I. J. Goodfellow et al., "Generative Adversarial Netowrks," arXiv:1406.2661 [stat.ML], Jun. 2014.

[5] R. C. Çalik and M. F. Demirci, "Cifar-10 Image Classification with Convolutional Neural Networks for Embedded Systems," 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), Aqaba, Jordan, 2018, pp. 1-2, doi: 10.1109/AICCSA.2018.8612873.

[6] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167 [cs.LG], Feb 2015.

[7] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen "Improved Techniques for Training GANs," arXiv:1606.03498 [cs.LG], Jun. 2016.

[8] Z. Liu, P. Luo, X. Wang and X. Tang, Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015