Using WGAN to generate logos of football clubs

Miloš Mladenović
Faculty of Technical Sciences
University of Novi Sad
Serbia
milosml@outlook.com

***Abstract* — Generative Adversarial Networks have been present in a machine learning community for a few years and gained huge attraction when they first appeared due to their innovativeness and better final results compared to other generative models, like autoencoders, Markov chains etc. However, they were not perfect and many problems existed with their training and convergence. Idea that tried to overcome those difficulties was using Wasserstein distance in so called WGANs, so that is the architecture that I used in this paper to try and generate logos of football clubs, from a relatively small dataset. I chose this particular type of images because of their distinctive features and thought that it would be interesting to see the results. Results of generative models could not be quantified, but images that were generated, of which some look promising are shown at the end of the paper.**

Keywords— deep learning, neural networks; GAN; WGAN; logos;

I. Introduction

Advancements in deep learning have brought new possibilities and research areas that were not present or even imagined before them. One of those areas is using deep learning for unsupervised learning and discovery of rich, hierarchical models [1] that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech and symbols in natural language corpora. [2] Using deep neural networks for unsupervised learning is the ultimate goal of all hopes and researches in the recent years, because most of the data present in the real world, that gets created in every second is unlabeled and not appropriate for supervised learning. Getting useful knowledge from all that data and putting it under some control, or transform it to some known form is the "holy grail" of all machine learning researchers.

So far, the most successful advancements in deep learning have come from the discriminative models, that map a high-dimensional, rich sensory input to a class label. [3] These successes are mostly based on backpropagation and dropout algorithms. Unsupervised models, including deep generative models have had less success because of the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies. They had less success also due to difficulty of leveraging the benefits of piecewise linear units in the generative context. [2]

Generative Adversarial Network first proposed in [2] generative model is set against an adversary – a discriminative model that learns to determine whether a sample is from the model distribution or data distribution. The generative model can be imagined as a counterfeiter trying to produce fake currency and use it without detection; while discriminator is can be analogous to detective that tries to detect the counterfeit currency. Competition in this "minimax" game drives both "players" to improve their methods until the counterfeits can not be distinguished from the genuine currency.

In this paper, I try to implement Generative Adversarial Network that generates logos of football clubs and uses a loss function based on Wasserstein distance metric described in the chapters **II** and **IV,** with architecture based on Deep Convolutional GANs, whose details will be described later. This type of images were chosen because they have distinct and special features that make them easily distinguishable from other types of images or logos – they usually have round or triangular shape, with club's name written on the circle's top or bottom with details about city or country of origin drawn in the middle. I was hoping that these features could be recognized by GAN and more easily generated.

In the remaining chapters, there will be more details about different aspects of the problem and it`s solution. In chapter **II** a brief overview of the existing implementations and papers regarding the GANs, WGANs and similar problems is given. Following this, chapter **III** contains details about the dataset, it's scraping from the web, characteristics of the images and data preprocessing process. In chapter **IV** concrete network architecture, implementation, training details and algorithm are given. The last – chapter **V** contains conclusion on this whole process and paper, results that were achieved with this architecture, future work plans and ideas.

II. Previous Work

What follows in this chapter are brief overviews of the three most important research papers that provided basis for the evolution of GANs and also this paper.

A. *Generative Adversarial Nets [2]*

In this paper, the authors propose a new framework for estimating generative models via an adversarial process, in which they simultaneously train two models: a generative model *G* that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than *G*. The training procedure, or rather the actual goal for G is to maximize the probability of D making a mistake. This framework actually represents a minimax two-player game, and it is the core of Generative Adversarial Networks.

A range of datasets were used to train these neural networks, including MNIST, TFD (Toronto Face Database) and CIFAR-10. The generator networks used a mixture of rectifier linear activations and sigmoid activations, while the discriminator net used maxout [4] activations. Besides this, dropout was applied in training the discriminator net. Details about this neural network architecture will be given in chapter **IV** where implementation details are mentioned.

Some advantages of this work compared to previous ones in this field are that Markov chains are not needed, only backpropagation is used to obtain gradients, no inference is needed during learning and a wide variety of functions can be incorporated into the model. [2] Disadvantages of this approach are the following: there is no explicit representation of $p_g(x)$; the discriminator must be synchronized well with the generator during training and there are no guarantees of convergence and meaningful training results and validation options.

B. Unsupervised representation learning with Deep Convolutional Generative Adversarial Networks [5]

The authors of this paper, while planning to bridge the gap between the supervised and unsupervised learning, propose a new architecture and approach to generative models – using convolutional neural networks with certain architectural constraints for generative models – DCGANs. They demonstrated that deep convolutional networks can not only be used to train generative models, but also for learning a hierarchy of representations from object parts to scenes in both the generator and discriminator. After which they used learned features for some unusual tasks – demonstration of their applicability as general image representations.

DCGAN architecture and some guiding rules used in this paper were taken into account and used when I implemented my own Wasserstein DCGAN, described in details in chapter **IV**. DCGAN described in this paper was trained on various image datasets, including “Large-scale Scene Understanding” (LSUN), “Imagenet-1k” and “Faces” dataset. Some important guiding rules for stable DCGANs that I also followed, described in this paper are:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)
- Use batch normalization in both the generator and the discriminator
- Remove fully connected hidden layers for deeper architectures
- Use ReLU activation in generator for all layers except for the output, which uses *tanh*
- Use LeakyReLU activation in the discriminator for all layers

Besides pretty good results in generated images that were accomplished by using this approach, this paper also presented some other achievements that previous GANs and their authors were not able to accomplish. Those achievements include a way to empirically validate DCGAN’s capabilities and use it in supervised learning:

- Classifying CIFAR-10 using GANs as a feature extractor (*apply the GAN as feature extractor on supervised datasets and evaluate the performance of linear models fitted on top of these features*)
- Classifying SVHN (StreetView house numbers) [6] digits using GANs as a feature extractor
- Investigating and visualizing the internals of the networks

Advantages of this paper and implementation proposed in it are a more stable set of architectures for training generative adversarial networks and a proof that adversarial networks can learn good representations of images for supervised learning and generative modeling. One main disadvantage that still exists is that there are still some forms of model instability present – as models are trained longer they sometimes collapse a subset of filters to a single oscillating mode.

C. Wasserstein GAN [7]

The main subject of this paper is a proposed new method for calculating distance between two probability distributions – Wasserstein (Earth Mover’s) distance and implications that it bares in defining a loss function for GANs and training them. Also, they show the different ways to define a distance or divergence between different distributions and their comparison to Wasserstein distance.

Comprehensive theoretical analysis of how the Earth Mover (EM) distance behaves is given, along with a form of GAN called Wasserstein-GAN that minimizes a reasonable and efficient approximation of the EM distance and the corresponding optimization problem. Empirical proves that the WGANs cure the main training problems of GANs are also shown – in particular, that training WGANs does not require maintaining a careful balance in training of the discriminator and the generator, and does no require a careful design of the network architecture either, which leads to the reduction of the mode dropping phenomenon present in GANs. Details about the Wasserstein distance metric, along with loss function and network architecture are given in chapter **IV.**

Dataset that was used to train proposed WGAN model is LSUN–Bedrooms dataset, a collection of natural images of indoor bedrooms.

This paper’s main advantage is the new approach and distance metric that they offer, which gives much better results and training process on large datasets than most of the previous works. Main disadvantage, along with other GAN models is long and hard process of training and large dataset required for it.

III. Dataset and it's acquisition and properties; data preprocessing

Images that represent a dataset used for training in this paper were collected by scraping the web using a Python script and "beautiful soap" library. Most of the images that I found available on the internet and in appropriate format were from the Wikipedi. Football logos were chosen because of their distinct features and shape, which made me hope that GAN would be able to recognize this and create new examples that contain those basic attributes and shapes. Dataset that was collected and available on the net was relatively small – it includes only 4637 images. Some examples of the training data are given below:

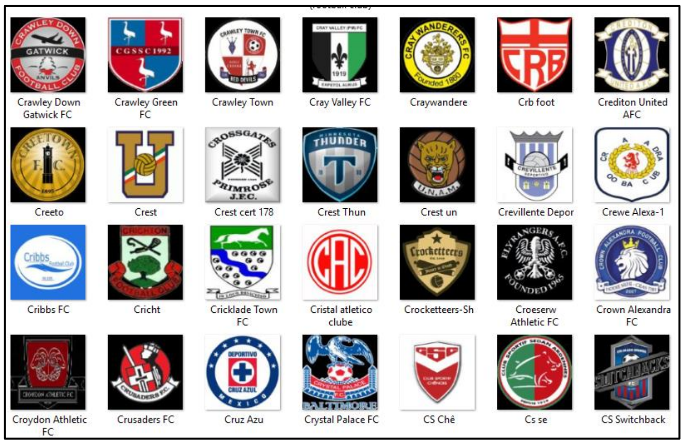


Figure 2: Training data examples

After scraping the images from the web, they were preprocessed so that they can be used in training with the following steps:

1. RGBA images were converted to RGB (alpha channel was removed), and they had black background
2. Images were resized to 128x128 pixels

That was the process done before all the training and preparation steps. Additional preparation steps – image processing, included:

- Changing image brightness to some random value
- Random image flipping left and right
- Changing image contrast to some random value
- Pixel values were normalized to values from 0 to 1
- A batch of 64 images was created before every training iteration

These steps were done in order to create some new examples that are different from all the collected images and that should allow the neural network to better generalize and come up with new ideas how generate new images.

IV. Implementation details, network architecture, training algorithm

This chapter contains detailed descriptions of neural network architecture, implementation details and improvements built upon original GAN, as advised in [7] and [5]. Also, details about Wasserstein distance metric, loss function, training algorithm and hyperparameter values used will be presented.

A. *Neural network architecture*

Architecture of network discussed in this paper is similar to the one described in [7] and it's general schema is shown on image below.

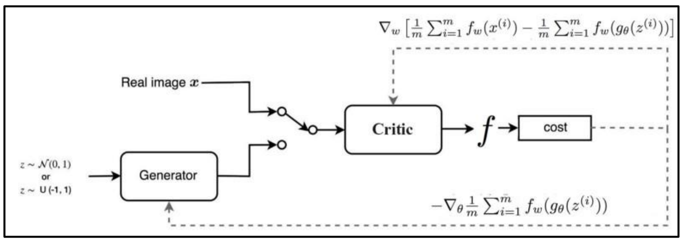

$\nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0,1)$ or $z \sim U(-1,1)$
Generator
$-\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

Figure 1: Generator-discriminator architecture

Discriminator is labeled as a critic and we can see that generator output and real image output come to discriminator, which determines whether the image was real or fake. Also, cost functions of WGAN network are given in the formulas on the image and they will be described next.

Generator neural network architecture is shown on the figure 3 below:

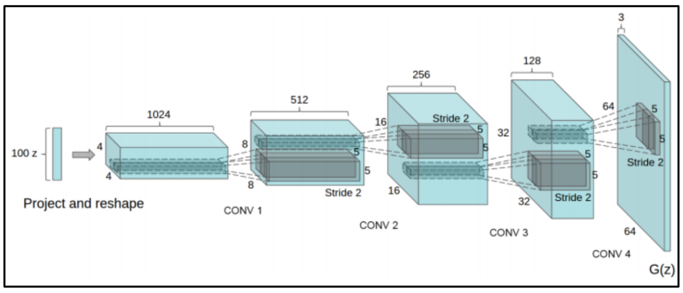


Figure 3: Generator architecture [5]

As can be seen – a 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of fractionally-strided convolutions then convert this high level representation into a 64x64 pixel image. [5] These fractionally-strided convolutional filters are wrongly called deconvolutional layers (layers) in some papers. All guiding rules for architecture described in chapter **II**, related to WGAN paper were followed.

B. *Wasserstein distance and loss function*

Original GAN paper [2] was using Jensen-Shannon divergence for measuring similarity between two probability distributions and appropriate loss function derived from it, but many problems existed with it, which made training GANs a very hard process and often impossible to complete. Some of them are: hard to achieve Nash equilibrium, bad low dimensional support, vanishing gradient [8] , mode collapse [7] and lack of a proper evaluation metric. Most of these problems were addressed in [7] by using a new distance metric between two probability distributions and a loss function derived from it. This distance metric called Wasserstein distance, or Earth Mover's distance, short for EM distance, because informally it can be interpreted as moving piles of dirt that follow one probability distribution at a minimum cost to follow the other distribution. The cost is quantified by the amount of dirt moved times the moving distance. When dealing with the continuous probability domain, the distance formula becomes:

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

In the formula above, $\Pi(p_r, p_g)$ is the set of all possible joint probability distributions between p_r and p_g. One joint distribution $\gamma \in \Pi(p_r, p_g)$ describes one dirt transport plan, but in the continuous probability space. Precisely $\gamma(x, y)$ states the percentage of dirt that should be transported from point x to y so as to make x follows the same probability distribution of y. That's why the marginal distribution over x adds up to p_g, $\sum_x \gamma(x, y) = p_g(y)$ (Once we finish moving the planned amount of dirt from every possible x to the target y, we end up with exactly what y has according to p_g.) and vice versa $\sum_y \gamma(x, y) = p_r(x)$. It is intractable to exhaust all the possible joint distributions in $\Pi(p_r, p_g)$,to compute $\inf_{\gamma \sim \Pi(p_r, p_g)}$, so the authors in [7] proposed a smart transformation of the formula based on the Kantorovich-Rubinstein duality to:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \le K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

Where by measuring supremum we want to measure the least upper bound or the maximum value. If we label this new function as f and it comes from a family of K- Lipschitz continuous functions $\{f_w\}_{w \in W}$, parametrized by w, the new discriminator is used to learn w to find a good f_w and the loss function is configured as measuring the Wasserstein distance between p_r and p_g.

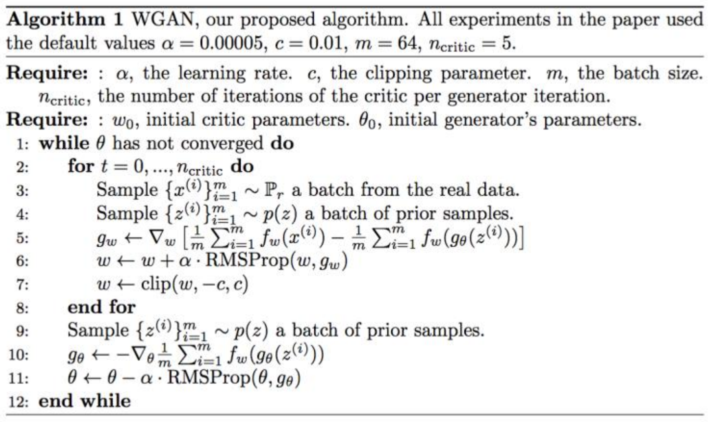
$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x\sim p_r}[f_w(x)] - \mathbb{E}_{z\sim p_r(z)}[f_w(g_\theta(z))]$$

Thus the "discriminator" is not a direct critic of telling the fake samples apart from the real ones anymore. Instead, it is trained to learn a K-Lipschitz continuous function to help compute Wasserstein distance. As the loss function decreases in the training, the Wasserstein distance gets smaller and the generator model's output grows closer to the real data distribution.

Only one issue arises with this – maintaining the K-Lipschitz continuity of f_w during training, in order to make everything work out. A simple trick is presented in the paper [7] and used in my implementation – after every gradient update, clamp the weights w to a small window, such as [-0.01, 0.01], which results in a compact parameter space W and thus f_w obtains its lower and upper bound to preserve the Lipschitz continuity.

C. *Training algorithm*

Algorithm that was used to train the WGAN in [7] was used in my implementation also. It's steps, taken from the mentioned paper are given on the figure below:



Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

Figure 4: WGAN training algorithm [7]

Compared to the original GAN algorithm, as previously mentioned, this one undertakes three main changes:

- After every gradient update on the critic function, clamp the weights to a small fixed range
- Use a new loss function derived from the Wasserstein distance, no logarithm anymore. The "discriminator" model does not play as a direct critic but a helper for estimating the Wasserstein metric between real and generated data distribution.
- It uses RMSProp optimizer instead of an Adam, which is known to cause instability in the model training. [7]

The only difference to the proposed hyperparameters above – I used learning rate of 0.0002, proposed in DCGAN paper. [5] In the following chapter, training results, along with some conclusions and future plans will be presented.

V. Training results and conclusions

When I first started thinking about generative models and how to try and use them for something new and creative, my first idea was to use Conditional GAN [9] to generate logos of football clubs conditioned on some labels such as club's city name, colors or country of origin. Because of their distinct shape and unique properties, my thoughts were that CGAN it would generate some interesting new examples. Unfortunately, lack of data available on the internet and finite number of footballing countries and football leagues and clubs in the world made that task pretty hard, so I tried to see what could WGAN generate with the available data.

WGAN described in this paper was trained on GPU - GeForce GTX 1060 with 6GB of DDR5 RAM and AMD Ryzen 5 1600 CPU. Training to 1600 epochs took around 3 days. Some examples after a few hundred training epochs are given on figures below.

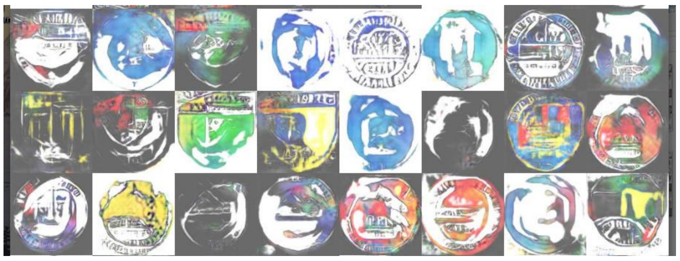


Figure 6: Some of the resulting logos after 250 epochs

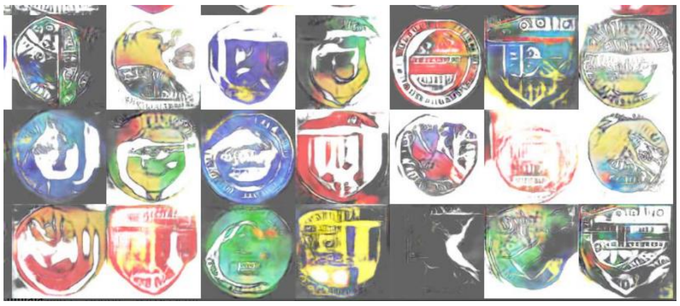


Figure 7: Resulting images after 350 epochs

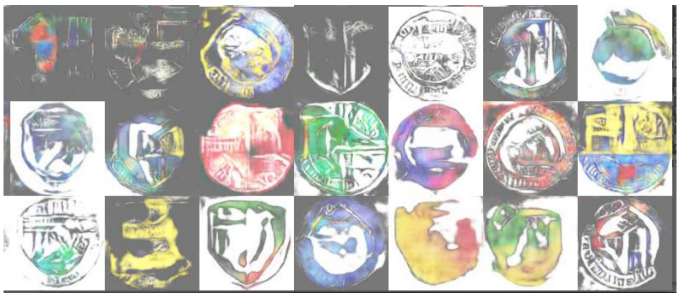


Figure 8: Results after 900 epochs

As can be seen from the images above, not much difference between generated images occurred between different epochs, probably because dataset was so small. Some sort of mode collapse probably happened, despite it being WGAN architecture, which can also be seen on images below, in later epochs.

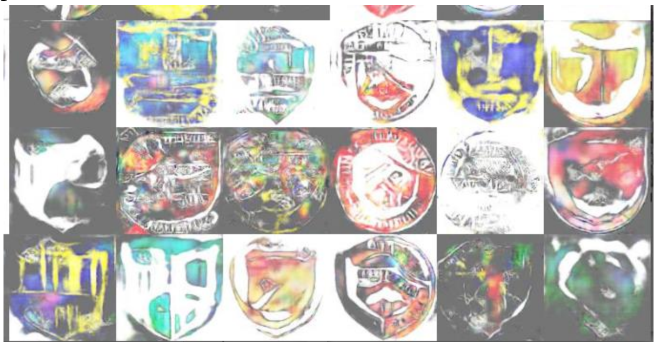


Figure 5:Figure 8: Results after 1400 epochs

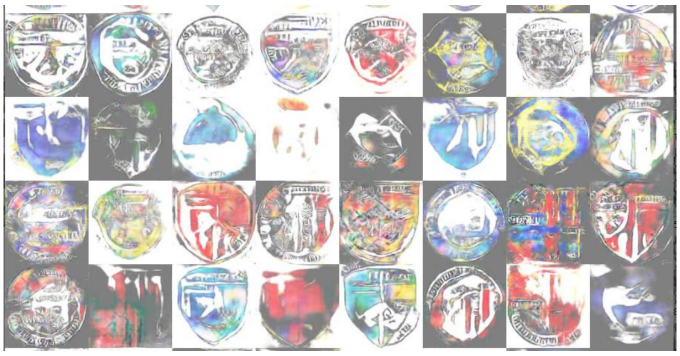


Figure 9: Some results after 1550 epochs

Sadly, as evident in my paper and implementation, along with the original paper, Wasserstein GAN is not perfect, and even authors of the original WGAN paper mentioned that "*weight clipping is a clearly terrible way to enforce a Lipschitz constraint*". [7] It still suffers from unstable training, as evidenced above, slow convergence after weight clipping and vanishing gradients. Some improvements, like replacing weight clipping with gradient penalty, suggested in [10] can maybe prove to be beneficial to WGAN training and results.

Also, as evidenced numerous times before and shown here, without a really large dataset for training, neural networks or any machine learning models cannot produce any meaningful results.

VI. References

[1] Y. Bengio, Learning deep architectures for AI, Now Publishers.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets," 2014.

[3] G. D. L. Hinton, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke and P. Nguyen, "Deep neural

networks for acoustic modeling in speech recognition," 2012.

[4] I. J. Goodfellow, D. Warde-Farley, M. Mirza and A. Courville, "Maxout networks," 2013.

[5] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with Deep Convolutional Generative Adversarial Networks," 2016.

[6] Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[7] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein GAN," 2017.

[8] M. Arjowsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017.

[9] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," 2014.

[10] I. Gulrajani, "Improved training of Wasserstein GANs," 2017.