# Optimization with Optimistic Descent Algorithms : Review and Applications

**Pavan Chennagiri,Wenqing Zheng, Yiyue Chen, Zhenqiang Ying**
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712
pavancm, yiyuechen, w.zheng, zqying@utexas.edu

## Abstract

This work focuses on the review and applications of extra gradient methods popularly referred to as *optimistic* descent algorithms. First we introduce the concept optimistic mirror descent (OMD) and provide intuition as well as motivation behind this concept. Second we evaluate the performance of optimistic methods across a host of applications. In particular we compare the performance of OMD in convex, non-convex and saddle point problem settings. Lastly we observe that OMD has certain fundamental limitations and only outperforms the existing methods under certain specific considerations.

## 1 Introduction

Stochastic gradient descent (SGD) based methods have been receiving significant attention in the recent days especially due to their simplistic nature as well their success in deep learning based methods. Although SGD has comparatively slower convergence rate, various methods such as SGD with momentum, Adagrad [1], Adam [2] have been proposed to perform better than the naive SGD. In this work we focus on a particular class of algorithms which are based on extra gradient technique and improve on the exiting SGD based methods. Extra gradient methods differ from SGD by having an intermediate step for calculating additional gradient.

The method of employing additional gradient is generally referred to as *Optimism* and hence these methods are collectively referred to as optimistic descent methods. The *optimism* stems from the involvement of additional gradient in the hope that it will push the training along incumbent gradient. Optimistic methods have roots in mirror descent and mirror prox methods which is briefly reviewed in the next paragraph.

**Mirror Descent** For a strictly convex $\phi$, where $\phi$ is a mirror map, mirror descent is given by

$$\begin{aligned} \nabla\phi(y_{k+1}) &= \nabla\phi(x_k) - \eta\nabla f(x_k) \\ x_{k+1} &= \arg\min D_\phi(x, y_{k+1}) \\ &= \arg\min \eta\langle\nabla f(x_k), x\rangle + D_\phi(x, x_k) \end{aligned} \tag{1}$$

where $D_\phi(x, x_k) = \phi(x) - \phi(x_k) - \langle\nabla\phi(x_k), x - x_k\rangle$ is the Bregman divergence. Mirror descent can be thought as a generalization of standard gradient descent as eqn 1 with $\phi(x) = \frac{1}{2}||x||_2^2$ results in gradient descent update expression.

**Optimistic Mirror Descent** The optimistic version of mirror descent was first introduced in the context of online learning in [3] particularly to tackle saddle point problems, although it's applicability is not just restricted to saddle point objectives.

**OMD Algorithm.** *Let $\phi$ be a 1-strongly convex function and let $y_0 = \arg\min \phi(y)$. Suppose that at the beginning of every round t, we have access to $M_{t+1}$, a vector computable based on past information or side information, then OMD is given by*

$$y_{t+1} = \arg\min_x \eta_t \langle M_{t+1}, x \rangle + D_\phi(x, x_t),$$
$$x_{t+1} = \arg\min_x \eta_t \langle \nabla f(y_{t+1}), x \rangle + D_\phi(x, x_t)$$

Here $y$ is the intermediate sequence and $M_t$ is the prediction sequence which employs information from past gradients. For $M_{t+1} = \nabla f(x_t)$ OMD reduces to Mirror Prox algorithm. Other popular choices for $M_{t+1}$ include $M_{t+1} = \frac{1}{t}\sum_{k=0}^{t} \nabla f(x_k)$.

In this work we holistically evaluate and compare the performance of OMD algorithm across different class of problem settings and applications. The rest of the report is organized is as follows. In Section 2 and 3 we analyze OMD performance on convex and non-convex problem settings respectively followed by saddle point objectives in Section 4. We provide some concluding remarks in Section 5 followed by some pointers on future work in Section 6.

## 2 Convex Optimization Problem

In this section, we consider a specific convex optimization problem and experiment on the performance of mirror descent algorithm, optimistic mirror descent algorithm, its generalized version and other algorithms.

### 2.1 Problem Formulation

Consider the problem of robust regression. The problem can be formulated as a convex optimization problem:

$$\min_\beta : \quad ||\mathbf{X}\beta - \mathbf{y}||_1$$
$$s.t. : \quad \beta \in \mathcal{X}$$

Here the constraint set $\mathcal{X}$ is the simplex: $\mathcal{X} = \{\beta : \beta \geq 0, \sum \beta_i = 1\}$.

### 2.2 Extra Gradient Method in Convex Optimization and Its Generalization

Extragradient method was first proposed by Korpelevich in [4] and it has been used to solve variational inequality problems, especially in saddle point problems. After Korpelevich's work, other authors extended this extra-gradient method to other problems. For convex constrained optimization problems, [5] considered the performance of extra-gradient method under error bounds assumptions. When the objective function is the sum of two functions, the first being smooth and the second being convex(the results were later extended when both functions are convex), [6] showed sublinear convergence of extra-gradient method, as for gradient-based methods.

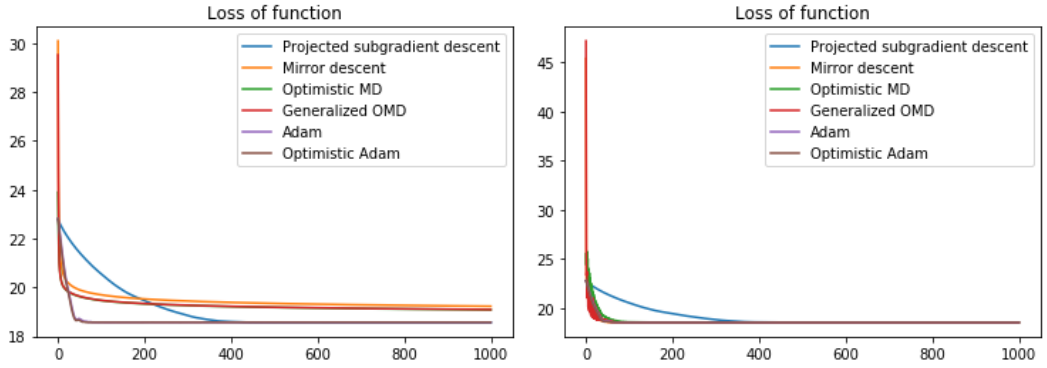
In extra-gradient method, the use of an additional projected gradient step can be seen as a guide during the optimization process. The introduction of mid-point can allow us to examine the geometry of the problem and curvature information near current point. In this example, we use the Mirror Prox algorithm(introduced in Section 1) as optimistic mirror descent.

The generalized version of the extra-gradient method was introduced in [7], where $(k-1)$ mid-points are inserted per iteration(in extra-gradient method, there is only one mid-point per iteration and $k = 2$). The generalized extra-gradient method can be written as:

$$X_{n+\frac{1}{k}} = \arg\min_{x' \in \mathcal{X}} \{\langle -\gamma_n \hat{g}_n, X_n - x' \rangle + D(x', X_n)\}$$
$$X_{n+\frac{k-s}{k}} = \arg\min_{x' \in \mathcal{X}} \{\langle -\gamma_n \hat{g}_{n+\frac{k-s-1}{k}}, X_n - x' \rangle + D(x', X_n)\}$$
$$X_{n+1} = \arg\min_{x' \in \mathcal{X}} \{\langle -\gamma_n \hat{g}_{n+\frac{k-1}{k}}, X_n - x' \rangle + D(x', X_n)\}$$

where $1 \leq s \leq k-2$.

The result in Theorem 5 in [7] shows the convergence and computational cost of generalized extra-gradient method in saddle point problem. However, to our knowledge, no result has been given in convex optimization problem. We therefore include this method in the experiments.




(a) Loss with step size $0.01\frac{1}{t+1}$

(b) Loss with step size $0.1\frac{1}{t+1}$

Figure 1: Comparison of 2 step sizes with the results of six algorithms: the x-axis represents iteration and the y-axis represents loss.

### 2.3 Experiments and Results

#### 2.3.1 Setup

This is an extension of computational problems in homework 4 and we use the same data generated from the 20 newsgroups dataset. Here $\mathbf{X}$ is an $n \times m$ matrix, where $n$ is the number of unique words considered and $m$ is the number of topics, and $\mathbf{y}$ is a noisy word frequency. In the experiment, $n = 1000$ and $m = 20$. The goal is to recover the distribution over topics $\beta$.
The problem is solved using stochastic (sub)gradient descent(SGD), mirror descent(MD), optimistic mirror descent(OMD), generalized OMD, Adam and Optimistic Adam. Generalized OMD is introduced in [7] and here we plug in one more mid-point per iteration. The step size is $O(\frac{1}{t+1})$ at iteration $t$ . We vary in step size and plot the loss over iterations using these six methods, respectively.

#### 2.3.2 Results

First we compare the three mirror descent related algorithms. These algorithms decrease the loss rapidly for several steps at the beginning but we see a much slower convergence in the following iterations. Optimistic MD(extra-gradient method) and Generalized OMD(generalized extra-gradient method) perform similarly(not exactly the same) and outperform vanilla mirror descent. Larger step size gives faster convergence but these three algorithms still perform quite similarly. We cannot see an obvious improvement by using generalized extra-gradient method in this optimization problem.
Adam and Optimistic Adam outperform mirror descent methods after 30 iterations and give lower loss when the step size is $0.01\frac{1}{t+1}$. They also perform similarly but not exactly the same. When the step size is $0.1\frac{1}{t+1}$, there is no obvious difference among all algorithms except Projected subgradient descent. Projected subgradient descent converges slowly but reaches the same loss as Adam related methods finally.
Overall, in this robust regression problem, the introduction of optimistic methods and its generalized version does not show better performance.

## 3 Non-Convex Optimization Problem

In this section, we will show the performance on a non-convex optimization problem. Here, we choose image quality prediction as an example to show that training deep regression networks with optimism is both feasible and promising.

### 3.1 Background

**Image Quality Assessment (IQA).** We live in an increasingly visual digital world. However, images, as a common container of visual information, suffer from different degree of various distortions

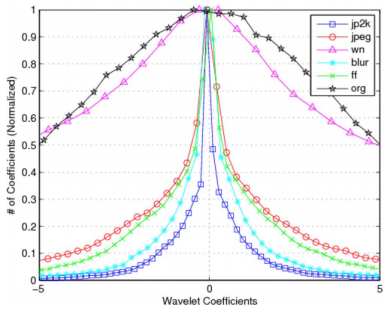



Figure 2: Subband statistics of an image for different distortions.

introduced during capturing, storing and/or transmission. Image quality assessment techniques are used to predict the visual quality of such distorted images, since using subjects to assess the visual quality is time-consuming, expensive, and unfeasible for real-time applications.

For IQA problem, the input is a color image and the output is a scalar score indicating the visual quality of the image. A higher number corresponds a higher image quality. The ground truth quality score, a.k.a mean opinion score (MOS), is computed by averaging the opinion scores given by a group of human viewers. We use deep neural networks to learn to predict MOS in an end-to-end manner. The problem can be formulated as:

$$\min_{\boldsymbol{\theta}} \ell(f(\boldsymbol{\theta}, \boldsymbol{I}_i), \boldsymbol{s}_i) \tag{2}$$

where $f$ is the deep neural network with trainable parameters $\boldsymbol{\theta}$, $\boldsymbol{I}_i$ is the $i$-th batch of the training data, $\ell(.,.)$ is the loss function, $\boldsymbol{s}_i$ is a vector of MOSs corresponding to each image in the $i$-th batch.

**Natural Scene Statistic (NSS).** Most existing IQAs are developed based on the statistics of natural images. Natural un-distorted images possess certain statistical properties that hold across different image contents. For example, it is well known that the power spectrum of natural scenes fall-off as (approximately) $1/f^{\gamma}$, where $f$ is frequency. Natural scene statistic (NSS) models seek to capture those statistical properties of natural scenes that hold across different contents. NSS-based IQA approaches assume that the presence of distortions in natural images alters the natural statistical properties of images, thereby rendering them (and consequently their statistics) unnatural, as shown in Fig. 2. The goal is to capture this "unnaturalness" in the distorted image and relate it to perceived quality.

### 3.2 Experiment Setup

**Optimizers.** Since the objective function is changing with different inputs, it can be viewed as an online learning problem where the players plays $\theta$ and the environment provides $\boldsymbol{I}_i$ and $\boldsymbol{s}_i$. Here, since training images are sampled from natural image database and the human opinion scores are highly correlated with the NSS features, we can assume that the gradient sequence is predictable. Therefore, we expect that introducing optimism to optimization algorithms can provide faster convergence rate.

**Networks.** In deep learning, the convolutional neural network (CNN) is commonly applied to analyzing visual imagery. We used a shallow CNN with 3 layers of kernel size 3, stride 1, and no padding. The channel numbers are 3, 16, and 16 sequentially. The output layer produces a scalar number supervised by the target image quality score (MOS). We used ReLU as activation function and didn't apply fancy self attention, batch normalization, or dropout. Mean Square Error between predicted scores and the ground truth scores are used as the loss function.

**Databases.** We use two image quality databases: CLIVE [8] and KonIQ [9]. The former contains 1162 images (175 ratings per image), and the latter contains 10,073 images annotated by 1467 crowd workers (120 ratings per image).

### 3.3 Results

We follow the common pratice in the field of IQA to use SROCC(Spearman's rank correlation coefficient) to measure the performance of the quality score prediction. A higher SORCC indicates

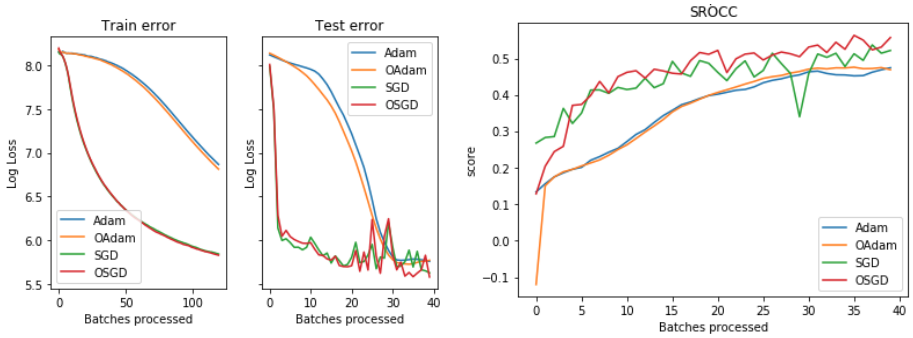



Figure 3: Results on CLIVE database.

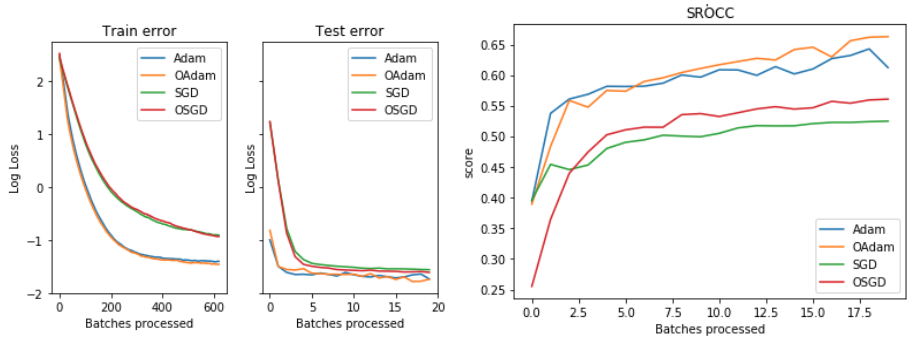



Figure 4: Results on KonIQ database.

| opt. func. | train data | SROCC on CLIVE | SROCC on KonIQ |
|---|---|---|---|
| Adam | KonIQ | 0.434014 | 0.612605 |
| OAdam | KonIQ | 0.471167 | 0.663117 |
| SGD | KonIQ | 0.351817 | 0.525014 |
| OSGD | KonIQ | 0.449329 | 0.560982 |
| Adam | CLIVE | 0.475049 | 0.488603 |
| OAdam | CLIVE | 0.469636 | 0.500744 |
| SGD | CLIVE | 0.522416 | 0.62065 |
| OSGD | CLIVE | 0.557891 | 0.613741 |

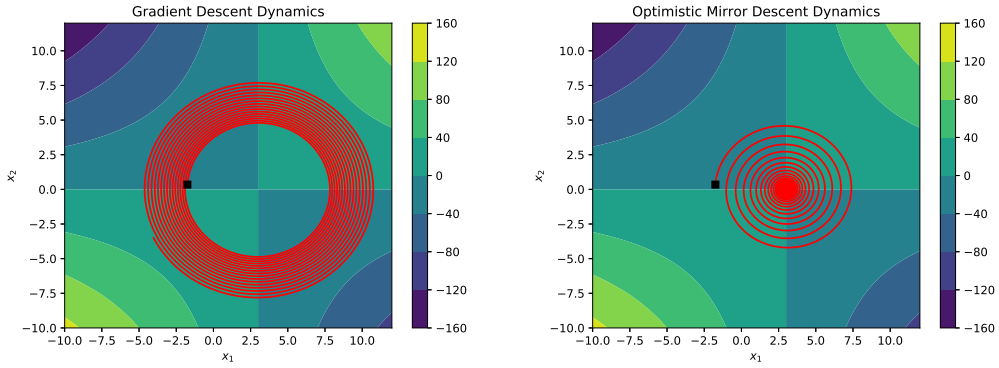
Table 1: Performance on two databases.

a better performance. In all cases, optimistic versions achieve better performance. As we can see from Fig. 3, when training on small databases, SGD/OSGD gives faster convergence but the curves of the testing error are more fluctuating and unstable compared with that of Adam/OAdam. This is because the statistical characteristics of the sampled data vary more when sampling a batch from a smaller data set. Adam and OAdam perform more stable because they introduce moving average and bias correction. When training on a large database, both Adam/OAdam and SGD/OSGD give stable convergence on the validation losses.

Table 1 shows the cross database validation results. Models trained on one dataset can generalize well on another database, since their performances are comparable with those trained and validated on the same database.

## 4 Saddle Point Objectives

Saddle point objectives involve a minimax optimization as shown in equation SP which is typically modeled as a two person zero sum game being played with two optimizers, where the aim of the first objective will be to achieve least possible loss while the other tries to obtain the highest possible




(a) Gradient Descent dynamics

(b) Optimistic Mirror Descent dynamics

Figure 5: Comparison of Gradient and Optimistic mirror descent dynamics. The objective employed is $\min_{x_1} \max_{x_2} \langle x_2, 3 - x_1 \rangle$ which has saddle point at $(3, 0)$. Black marker indicates the initialization point

reward simultaneously.

$$\min_{x_1 \in X_1} \max_{x_2 \in X_2} f(x_1, x_2) \tag{SP}$$

In case of convex-concave $f$, there exists a unique saddle point $(x_1^*, x_2^*)$ guaranteed by Sion's minimax theorem. However for non convex-concave problems it's possible to have multiple saddle points, many of which may not be the desirable for certain applications. Applying standard gradient descent on non convex-concave problems may not necessarily yield the optimal solution. Further it may often diverge for certain class of saddle point objectives as shown in Fig. 5a. Another challenge in using gradient descent is that it achieves convergence in the average of iterates, however it is often desirable to have last iterate convergence in many of the applications. Also averaging over all the iterates maybe a prohibitive approach in some applications such as Generative Adversarial Networks (GAN) etc.

In [10] the conditions under which the standard gradient descent converges was proved by employing the concept of coherence. Coherence is motivated from the monotonic property of saddle point problems which is stated as

$$\langle g(x), (x - x^*) \rangle \geq 0 \tag{MP}$$
$$\text{where} \quad g(x) = (g_1(x), g_2(x)) = (\nabla_{x_1} f(x_1, x_2), -\nabla_{x_2} f(x_1, x_2))$$

**Definition** We say that SP is coherent if:

1. There exists a solution $p$ of SP that satisfies MP.
2. Every solution $x^*$ of SP satisfies MP locally, i.e., $\forall\ x$ sufficiently close to $x^*$.

Intuitively coherence can be thought as a generalization of convex-concave condition. As an extension, two more conditions can be derived from coherence which are termed as strict and null coherence. Strict coherence refers to the problem which satisfies monotonic property MP in a strict fashion. One example for strict coherence is a function $f$ which is strictly convex-concave. Null coherence is the other end of the spectrum where the equation SP is satisfied with only equality. Bilinear objectives with interior solutions are example for null coherent objectives: for instance $f(x_1, x_2) = x_1 x_2$ has $\langle g(x), x \rangle = x_1 x_2 - x_2 x_1 = 0\ \forall\ x_1, x_2$. Lastly, neither strict nor null coherence imply any unique solution to SP. They are mainly constructs employed to determine the convergence of gradient and optimistic mirror descent methods.

Theorem 3.1 and it's corollaries in [10] show that mirror descent (which includes standard gradient descent as well) converges only when the function $f$ is strictly coherent and diverges for null coherent problems. This is clearly observed in Fig. 5a where the optimization involves a null coherent objective.

To overcome the drawbacks posed by mirror descent, the authors in [10] propose to use extra gradient analysis technique as *Optimistic Mirror Descent* (OMD). OMD was first proposed in the online learning literature by [3] and was shown to achieve last iterate convergence, a crucial component in some of the saddle point applications. Theorem 4.1 in [10] proves that OMD converges in last iterate for coherent problems (for both strictly as well as null coherent cases). OMD for saddle point is adapted from generic OMD in a slightly modified version with the updates given in equation 3 where minimization is performed over $x$ and maximization over $y$

$$\begin{aligned} x_{t+1} &= x_t - 2\eta\nabla_{x_t} f + \eta\nabla_{x_{t-1}} f \\ y_{t+1} &= y_t + 2\eta\nabla_{y_t} f - \eta\nabla_{y_{t-1}} f, \end{aligned} \tag{3}$$

The important thing to be noted here is that only for *coherent* problems satisfying the assumptions stated in the Theorem 4.1 in [10] using OMD converges to a saddle point. But the proof does not provide any guarantee that the rate of convergence is better than the standard gradient descent. Also applying OMD to problems with multiple saddle points may lead to convergence to one of them, which need not necessarily be the desired optimal solution. In such cases it is also not known apriori to which saddle point OMD converges.

### 4.1 Saddle Point Problems: Experimental Results

In this section the performance of OMD is evaluated on three different tasks, the first one involves learning mixture of Gaussians on a synthetic data and the second one is image generation task, both involving training GANs. The last one involves an indoor positioning estimation application. Before we discuss the experimental results a brief review of GANs is provided in the following paragraph.

**Generative Adversarial Networks** GANs are a class of unsupervised generative models which learn the data distribution $p_{data}$ using a large corpus of data by means of an adversarial loss. GANs were first proposed in [11] and they are built around two functions: the generator $G(z)$ generates a data sample from $p_{data}$ for a random $z$ sampled from a uniform distribution, ($z$ is also referred to as latent input) and the discriminator $D(x)$ provides inference whether the sample $x$ belongs to the data distribution $p_{data}$. From game theoretic standpoint GANs are viewed as a minimax zero sum game played between two players (the two players here are $G$ and $D$) and the learning occurs in a joint fashion where $D$ and $G$ train in an alternate manner. The minimax objective proposed in [11] is given by

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))], \tag{4}$$

where $p_{data}$ is the true data distribution and $p_z(z)$ is the distribution from which sample $z$ is drawn from. Since their introduction various GAN architectures have been proposed in the literature. Deep Convolutional GAN (DCGAN) [12] used Convolutional Neural Networks (CNN) both in $G$ and $D$. DCGAN along with the original GAN used Jenson-Shannon (JS) Divergence which is a symmetric version of Kullback Leibler (KL) Divergence in their objectives while training for measuring the distance between the model distribution $p_{model}$ and $p_{data}$. However JS divergence is known to suffer from vanishing gradient problem where gradients decay to zero resulting in no learning. Wasserstein GAN (WGAN) [13] addressed this problem by employing Wasserstein distance in place of JS Divergence. WGAN objective is given as

$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{x\sim p_{data(x)}}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))], \tag{5}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions. To enforce the Lipschitz constraint, [13] propose weight clipping to lie within a compact space. However it has been shown in [14] that this weight clipping often leads to undesired behavior while training such as gradient or explosion or vanishing gradients. Instead of weight clipping an alternative way to enforce Lipschitz constraint in the form of gradient penalty was proposed in [14]. The objective in 5 with gradient penalty transforms to

$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{x\sim p_{data(x)}}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\nabla_{\hat{x}}D(\hat{x})-1)^2], \tag{6}$$

where the final term corresponds to gradient penalty. It is derived from the fact that a differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. In eqn 6 $\lambda$ is a penalty coefficient which penalizes whenever the function violates Lipschitz continuity. As recommended by authors in [14] we use $\lambda = 10$ in all our experiments.

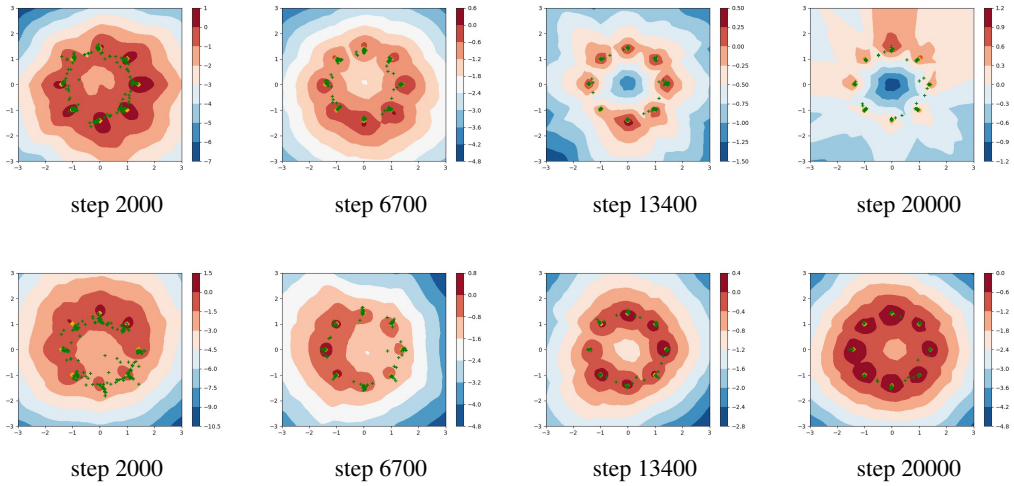



Figure 6: Adam vs Optimistic Adam (top and bottom respectively, learning rate $\eta = 5 \times 10^{-5}$ in both cases)

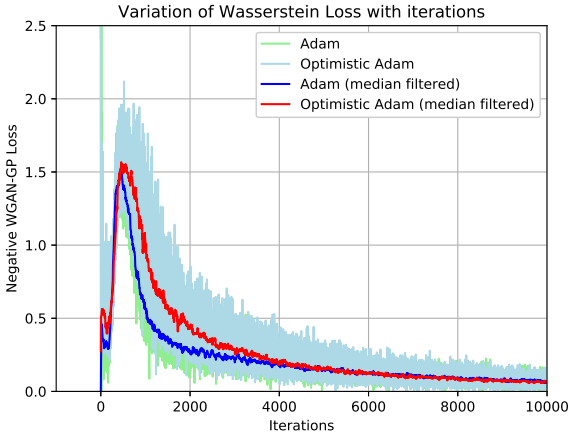
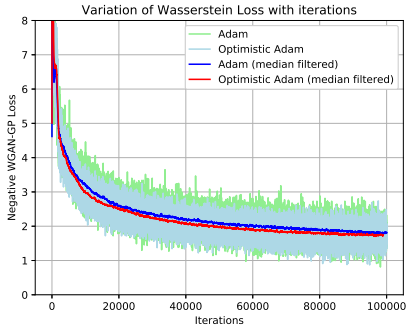



Figure 7: Variation of WGAN-GP loss with iterations

**Gaussian Mixture Models** In this experiment we consider data sampled from highly multi modal mixture of 8 Gaussians similar to the one used in [15]. The task is to learn the distribution of the mixture model by means of a GAN employing WGAN loss with gradient penalty (WGAN-GP) given in eqn 6. The generator and discriminator each have 3 fully connected layers with 512 neurons and Relu activations and generator generates two dimensional vectors. The networks were trained with Adam [2] and Optimistic Adam which was proposed in [16]. Explicit pseudo code for Optimistic Adam is provide in Appendix A for reference. The output after 2000, 6700, 13400, 20000 iterations is shown in Fig. 6. The variation of WGAN-GP loss is compared in Fig. 7. Although both algorithms appear to have similar convergence rate the output produced by Optimistic Adam appears to be better than that of Adam as shown in Fig. 6 indicating the convergence to saddle point nature of the Optimistic method. The models were trained for 20000 iterations with learning rate = $5 \times 10^{-5}$.

**Image Generation Task** In this section we compare the performance of optimistic methods on image generation task using GANs. The experiment is performed across two different variations.

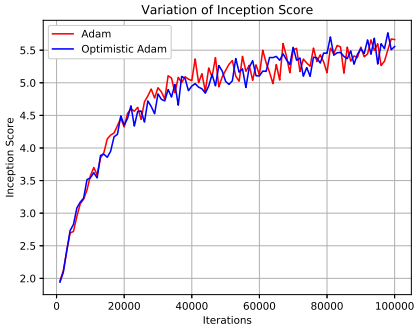
First experiment is to check the performance variation across two different datasets. For this purpose we use CIFAR-10 [17] and CelebA [18] datasets. CIFAR-10 consists of 60,000 images categorized



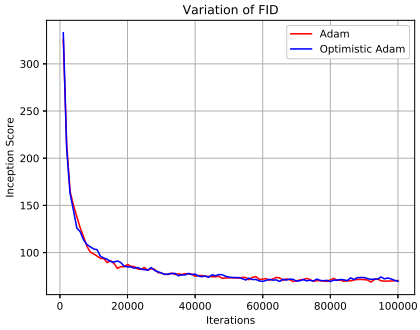

(a) CIFAR-10


(b) CelebA

Figure 8: WGAN-GP loss variation for CIFAR-10 (left) and CelebA(right)




(a) CIFAR-10 Inception Score




(b) CelebA FID

Figure 9: Quality variation of generated images

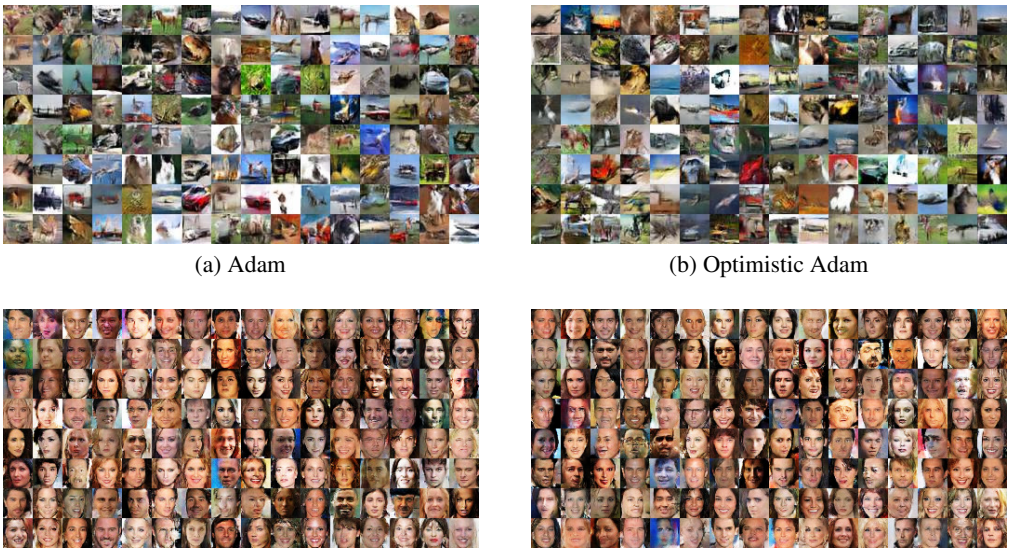
across 10 different classes. All images have a resolution $32 \times 32$. In case of CelebA, more than 200,000 images of consisting of celebrity faces are available.

Second experiment is evaluating the performance across different architectures of GAN. For this purpose we use DCGAN [12] with WGAN-GP loss and Boundary Equilibrium GAN (BEGAN) which uses a modified version Wasserstein loss. For evaluating the quality of generated images we use two metrics, Inception Score [19] and Fréchet Inception Distance (FID) [20]. In case of Inception Score, the generated images are applied to Inception model [21] which is pre-trained on ImageNet database to obtain the conditional label distribution. This along with true image distribution is used to obtain inception score. Note that Inception score is positively correlated with quality, where larger values imply better quality images.

FID is similar to Inception Score which uses inception model. The main difference arises on the way the output from inception model is employed. For FID features are obtained from a particular layer of the inception model. These features are empirically observed to follow a multivariate Gaussian distribution for a large class of images. The mean and covariance are calculated for both real data as well as generated data using these computed features and the distance between two Gaussians is measured by Fréchet distance [22] (also known as Wasserstein-2 distance) in order to quantify the quality of the generated samples.

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}), \tag{7}$$

where $(\mu_x, \Sigma_x)$ and $(\mu_g, \Sigma_g)$ are the mean and covariance matrix of features obtained from real and generated data respectively. FID scores are negatively correlated with visual quality where lower distance imply that the distribution of generated images is closer to that of the real data, thus higher quality.



(a) Adam (b) Optimistic Adam

(c) Adam (d) Optimistic Adam

Figure 10: Generated Images CIFAR-10 (top row) and CelebA (bottom row)

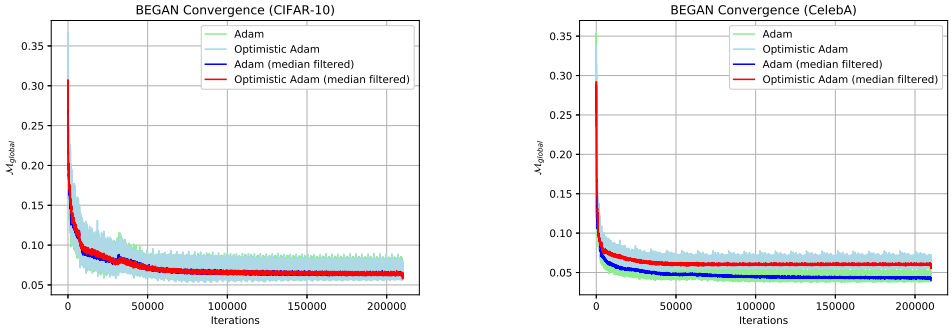
The following sections provide more details and results of the experiments we conduct for image generation task.

**DCGAN with WGAN-GP loss** We train a DCGAN with WGAN-GP loss for both CIFAR-10 and CelebA datasets, and compare their performance for Adam and Optimistic Adam optimizers. The architecture of DCGAN used in our experiments is detailed in Appendix B. The experiments were performed with a batch size of 64 and the resolution of input images for both CelebA and CIFAR-10 was kept constant at $32 \times 32$. The other hyperparameters were chosen as recommended in [14]. All the models were trained on Nvidia P100 GPU and took nearly 8 hours for 100,000 iterations. Fig. 8 shows the variation of WGAN-GP objective for 100,000 steps. Note that in both the cases the model hasn't sufficiently converged as the objective value is well above 1 indicating the slow convergence nature of WGAN. It can be observed that optimistic Adam has a slightly faster convergence than Adam. With regard to the quality of generated images, both Adam and Optimistic Adam achieve similar performance as illustrated in Fig. 9. The main takeaway from this experiment is that Adam and Optimistic Adam have negligible difference both with respect to convergence rate as well as quality of generated images. Note that we are restricting our analysis to 100,000 steps, it is quite possible that the performance improves/decreases if more iterations are used. Fig. 10 contains a subset of generated images (images can be zoomed for better quality). Visually the images from both optimization methods appear to be of similar quality.

**Boundary Equilibrium GAN (BEGAN)** BEGAN [23] is an extension of WGAN and motivated from Energy Based GAN (EBGAN) [24] where the descriminator $D$ is modeled as an energy function by means of an autoencoder. The objective of BEGAN is to match the loss distributions of real and generated images, which is in contrast to the traditional GANs which try to match the data distributions. If $x \in R^{N_x}$, then $D(x) : R^{N_x} \rightarrow R^{N_x}$, BEGAN aims to match auto-encoder loss distributions using Wasserstein (earth-mover) distance. The loss function for the auto-encoder is given by

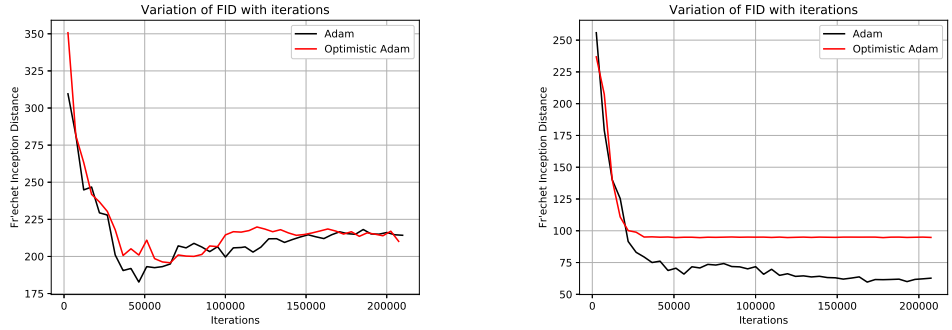
$$L(x) = |x - D(x)|^n; n = 1, 2, \tag{8}$$

where $L(x)$ is the loss of real image sample $x$ and $L(G(z))$ is the corresponding loss of generated sample $G(z)$. Since the aim is to match the loss functions with respect to Wasserstein distance, upon




(a) CIFAR-10 (b) CelebA

Figure 11: Variation of convergence for CIFAR-10 (left) and CelebA (right)




(a) CIFAR-10 FID (b) CelebA FID

Figure 12: Quality variation of generated images for BEGAN

modification as discussed in [23], the GAN objective reduces to

$$\begin{aligned} L_D &= L(x) - k_t L(G(z)) \\ L_G &= L(G(z)) \\ k_{t+1} &= k_t + \lambda_k(\gamma L(x) - L(G(z)), \end{aligned} \tag{9}$$

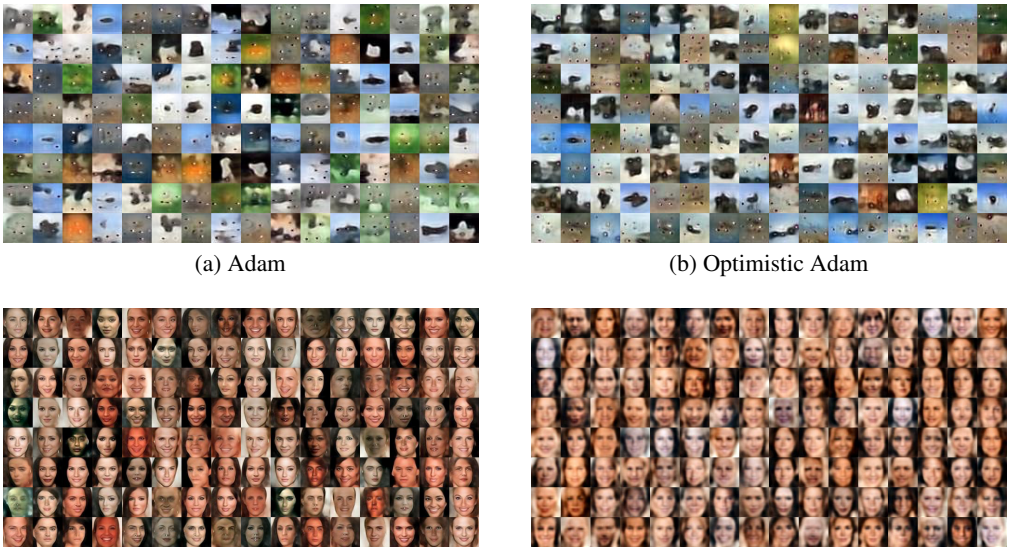
where $\gamma \in [0, 1]$ is a hyperparameter referred to as diversity ratio mathematically represented as

$$\gamma = \frac{\mathbb{E}[L(G(z))]}{L(x)}, \tag{10}$$

where $\gamma$ term balances between two goals: auto-encode real images and discriminate real from generated images. Lower values of $\gamma$ gives more prominence to auto-encoding while larger values result in more diversified image model. The term $k_t \in [0, 1]$ is a parameter at each iteration step in order to maintain the equilibrium $\mathbb{E}[L(G(z))] = \gamma\mathbb{E}[L(x)]$. The network architecture employed in BEGAN is provided in Appendix C. The authors of BEGAN also propose a global convergence using the equilibrium concept given by

$$\mathcal{M}_{global} = L(x) + |\gamma L(x) - L(G(z))| \tag{11}$$

This measure can be used to check if the network has converged or has suffered mode collapse where any latent input provides same output. BEGAN models were trained for 200,000 iterations which took nearly 16 hours on a Nvidia P100 GPU. In Fig. 11 the variation of convergence measure for both datasets is illustrated. While the rate of convergence is similar for both optimization methods for CIFAR-10, it differs for CelebA where Adam has a faster rate than Optimistic Adam. With regard to generated image for BEGAN, it can be observed from Fig. 12 that for CIFAR-10 both algorithms



(a) Adam (b) Optimistic Adam

(c) Adam (d) Optimistic Adam

Figure 13: Generated Images CIFAR-10 (top row) and CelebA (bottom row) with BEGAN

result in similar quality, though the FID values indicate image quality is much worse than those generated with WGAN-GP. This can be visually compared by observing Figs. 10 and 13.

In case of CelebA, FID values for optimistic Adam saturate after certain number of iterations implying convergence of the model to a specific saddle point. From Fig. 13 we can observe that the converged saddle point is not the desired optimal solution as the generated images are blurry in nature when compared to those generated using Adam. As mentioned previously, in case of optimistic methods though the convergence is guaranteed, it is quite possible in case of problems with multiple saddle points to converge to a non-optimal solution. Also the convergence is guaranteed for only coherent objectives, which is non-trivial to determine in case of GAN objectives due to their inherent non-linear nature.

#### 4.1.1 Indoor positioning task

**System Description** Indoor positioning is a key enabler for a wide range of applications, including navigation, smart factories and cities, surveillance, security, IoT, and sensor networks. Additionally, indoor positioning can be leveraged for improved beamforming and channel estimation in wireless communications. The object in this part is to design and train an algorithm that can determine the position of a user, based on estimated channel frequency responses between the user and an antenna array.

The final goal of training is to learn a network able to estimate position with high accuracy (low root-mean-square position error), averaged over all positions in the test dataset.

**About the Dataset** Channel vectors from a dataset created with the channel sounder are used. The dataset comprises channel responses and associated position ground truth information, The dataset are partitioned into training and probe sets as found appropriate for the algorithm development and training.

The dataset was acquired by the massive MIMO channel sounder described in [1]. Specifically, channel responses were measured between a moving transmitter and an $8 * 2$ antenna array. As transmitter, an SDR-equipped vacuum-cleaner robot was used, and it drove in a random path on a $4 * 2$ m table and transmitted uplink OFDM pilots with a bandwidth of 20 MHz and 1024 subcarriers at a carrier frequency of 1.25GHz. 10 % of the subcarriers were used as guard bands.

Figure 14 shows an example of ground truth coordinates, measured by a tachymeter with an accuracy below 1cm. Figure 15 shows a picture of the measurement setup.

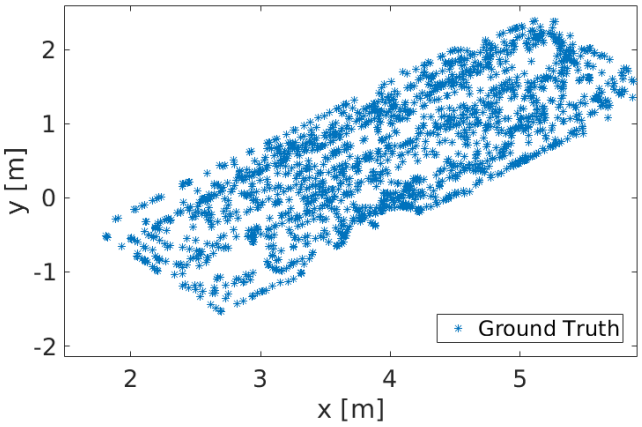



Figure 14: Indoor Dataset Ground Truth

Figure 15: Indoor Dataset Ground Truth

The available data is stored in two files: i) channel responses, ii) ground truth positions. The channel variables are mapped to the position in order of the first dimension of numpy array. For each measurement, the channel is a matrix with dimension: the number of antennas (16) times the number of used subcarriers (924), so the dimension of the channel database is 17486 (number of measurements) * 924 * 16. To leverage its real part and imaginary part, we concatenate the real and imaginary part in one observation together, and form a new matrix ready for training with size: 17486 * 924 * 32. The position database contains the coordinates x,y,z (in order), and the dimension of position database is 17486 * 3. The whole database is divided into training data-set of size 16000, and test data-set of size 1486.

**Model Description** To compare the performance of Adam and Optimistic Adam, we make this problem a saddle point problem in the first trail, by adopting the structure of GAN to learn the distribution transformation between the channel matrix and the position vectors. The model flowchart is shown in Fig. 16, the generator use the CNN to convert channel matrix to estimated position, while the discriminator takes into both the channel and estimated position to see if the joint distribution obeys the real channel-position pair. The objective function for posGAN is then:

$$V(D) = \max \underset{(H,p)\sim DB}{\mathbb{E}} \log(D(H,p)) - \underset{H\sim DB}{\mathbb{E}} \log(D(H,G(H))) - \underset{(H,p)\sim\mathcal{N}(0,\Sigma)}{\mathbb{E}} \log(D(H,p)) \tag{12}$$

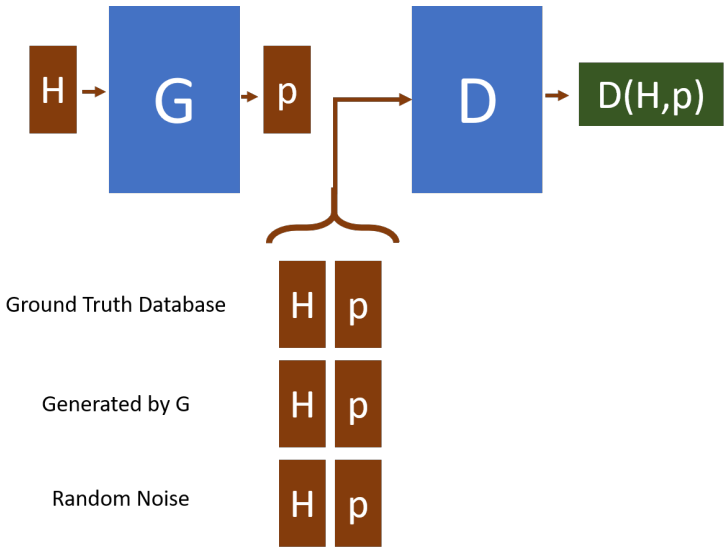



Figure 16: posGAN structure demonstration

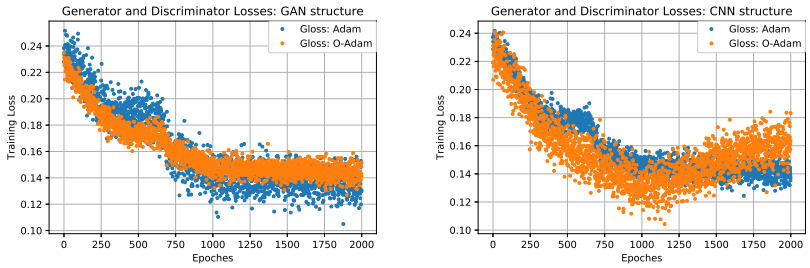



Figure 17: Training Losses for two trails

$$V(G) = \min \mathop{\mathbb{E}}_{H \sim DB} - \log(D(H, G(H))) \tag{13}$$

During each epoch, the mean square estimation error is calculated via $MSE = \mathbb{E}(||\hat{p} - p||_2)$, where the $\hat{p}$ is the position from the generator, and $p$ is the ground-truth position mapped to the real database. We first compared the results of training the GAN with optimistic Adam, Adam, Adagrad, optimistic Adagrad, and SGD, then changed the network structure and compared the performance on a smaller network.

In the first trial, to subtract the position information, we use the GAN structure illustrated in 16. In the second trail, we removed the Discriminator sub-network and directly train the Generator, with the altered objective function being:

$$V(G) = \min \mathop{\mathbb{E}}_{H \sim DB} ||p - G(H)||_2 \tag{14}$$

We take the convolutional neural network with stride 2. In this case, it is not a saddle point problem, and the comparison of performances of OAdam and Adam is also compared.

**Experiment results** The comparison among optimistic Adam and Adam are shown in Fig. 17,Fig. 18, and Fig. 19.

It can be seen from Fig 17 that in this experiment setting and this assumption, with saddle point problem (GAN structure in trail one) the optimism contributed to the speed-up of convergence at the initial stage of training, but in this realization, the Adam start to outperform OAdam after 600 iterations, and the final training loss of Adam is slightly small than Adam. In the second trail, the

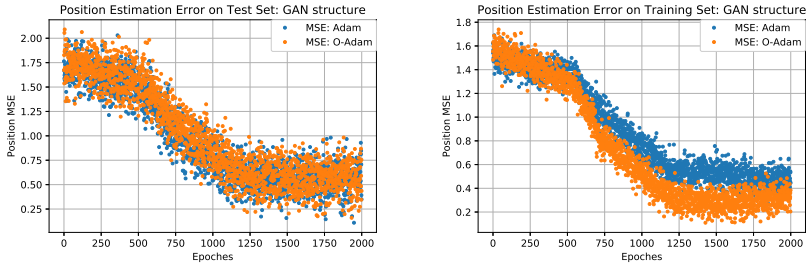



Figure 18: Train and Test MSE for trail 1

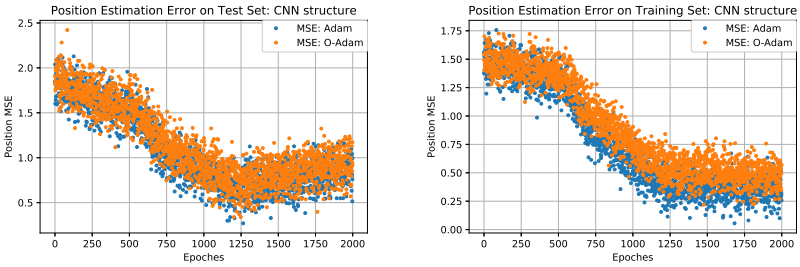



Figure 19: Train and Test MSE for trail 2

optimism only offers minor advances during first 900 iterations, which could be easily submerged below noise level; and after 900 iterations, the optimistic Adam start to increase its loss.

It can be seen from Fig. 18 that in this saddle point problem, the OAdam offers slightly smaller MSE on the training dataset, but performs similar on the test set. However, the optimism lost its advantage in the non-saddle point problem as demonstrated in Fig. 19. In this trail, the test set MSE for both algorithms are still non-distinguishable, but the training MSE for Adam is better then OAdam.

## 5 Conclusion

Though optimistic methods appear to be promising, at the outset they suffer from some elementary limitations. The major advantages as well as the drawbacks for every application is summarized as follows.

**Convex Optimization Problem** In this robust regression problem, we do not see an obvious improvement using optimistic algorithms. Further the introduction of more mid-points in generalized extra-gradient method does not show improvement over vanilla methods. We also see slower convergence in extra-gradient methods than projected subgradient method and Adam algorithms after 200 iterations.

**Non-Convex Optimization Problem.** We use image quality assessment as an example for testing the performance when introducing optimism to training non-convex problem. The experimental results show that the optimistic versions of two decent algorithms give better performances for both big and small databases.

**Saddle Point Objectives** The experimental results does not guarantee a better performance of optimism when applied to saddle point problems. When hyper-parameters are set to be the same, one optimizer could perform better or worse than its optimistic version, and even when the optimistic version performs better, this accuracy gain lost when the problem is modified back to the non-saddle point problem. Though Theorem 4.3 from [10] proves convergence for optimistic methods, it has two fundamental limitations. Firstly it is limited to coherent problems, but determining coherency is in general non-trivial in nature. Secondly for problems with multiple saddle points it may converge to a non-optimal saddle point which was observed in one of the applications discussed in this work.

Therefore applying OMD to any saddle point problem might not necessarily converge, in some cases it may even converge to a non desirable solution. Lastly saddle point problems especially GANs are highly sensitive to the data used for training as well as hyper-parameter variations. Thus convergence of a model to the desired solution is not solely dependent on the choice of the optimization method.

## 6 Future Work

Even though the extragradient method has been used and proved convergence in many convex as well as saddle point optimization problems, there is still no general result for the convergence rate, especially for the generalized version. The selection of step size may also affect the performance. More systematic and unified work could be done to better understand these algorithms in general convex optimization problems.
As for non-convex optimization problems, we only show the results on deep models trained from scratch. A common practice in deep learning is to start with a model pretrained on a large related dataset and finetune its parameter for tasks that only has a small dataset available. Therefore, the performances of optimistic versions of optimization algorithm when finetuning a deep network is also worth-exploring.

## References


[1] Duchi, J., E. Hazan, Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[2] Kingma, D. P., J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Rakhlin, S., K. Sridharan. Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems*, pages 3066–3074. 2013.

[4] Korpelevich, G. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[5] Luo, Z.-Q., P. Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

[6] Nguyen, T. P., E. Pauwels, E. Richard, et al. Extragradient method in optimization: Convergence and complexity. *Journal of Optimization Theory and Applications*, 176(1):137–162, 2018.

[7] Mokhtari, A., A. Ozdaglar, S. Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. *arXiv preprint arXiv:1901.08511*, 2019.

[8] Ghadiyaram, D., A. C. Bovik. Massive online crowdsourced study of subjective and objective picture quality. *IEEE Transactions on Image Processing*, 25(1):372–387, 2016.

[9] Lin, H., V. Hosu, D. Saupe. Koniq-10k: Towards an ecologically valid and large-scale iqa database. *arXiv preprint arXiv:1803.08489*, 2018.

[10] Mertikopoulos, P., H. Zenati, B. Lecouat, et al. Mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*, 2018.

[11] Goodfellow, I., J. Pouget-Abadie, M. Mirza, et al. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. 2014.

[12] Radford, A., L. Metz, S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] Arjovsky, M., S. Chintala, L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. 2017.

[14] Gulrajani, I., F. Ahmed, M. Arjovsky, et al. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777. 2017.

[15] Metz, L., B. Poole, D. Pfau, et al. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[16] Daskalakis, C., A. Ilyas, V. Syrgkanis, et al. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[17] Krizhevsky, A., G. Hinton. Learning multiple layers of features from tiny images. Tech. rep., Citeseer, 2009.

[18] Liu, Z., P. Luo, X. Wang, et al. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738. 2015.

[19] Salimans, T., I. Goodfellow, W. Zaremba, et al. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242. 2016.

[20] Heusel, M., H. Ramsauer, T. Unterthiner, et al. 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.

[21] Szegedy, C., V. Vanhoucke, S. Ioffe, et al. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826. 2016.

[22] Fréchet, M. Sur la distance de deux lois de probabilité. In *C. R. Acad. Sci. Paris*, pages 244:689–692. 1957.

[23] Berthelot, D., T. Schumm, L. Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[24] Zhao, J., M. Mathieu, Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

# A Optimistic Adam

**Algorithm 1** Optimistic Adam

**procedure** OPTIMISTIC ADAM OPTIMIZER
Parameters: learning rate $\eta$, decay rates for moments $\beta_1, \beta_2 \in [0, 1)$, loss function $f$, initial parameters $x_0$
**for** each iteration $t \in 1, ..., T$ **do**
Compute stochastic gradient: $\nabla_{x,t} = \nabla_x f_t(x)$
First moment: $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_{x,t}$
Second moment: $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\nabla^2_{x,t}$
First moment correction: $\hat{m_t} = \frac{m_t}{1-\beta_1^t}$
Second moment correction: $\hat{v_t} = \frac{v_t}{1-\beta_2^t}$
*Optimistic* gradient descent: $x_t = x_{t-1} - 2\eta\frac{\hat{m_t}}{\sqrt{\hat{v_t}}+\epsilon} + \eta\frac{\hat{m_{t-1}}}{\sqrt{\hat{v_{t-1}}}+\epsilon}$
Return $x_T$

# B DCGAN Architecture

Layers with corresponding output shapes are provided here for both Generator and Discriminator. Relu and Batch normalization is applied to every layer except for the final layer. Hyperbolic tangent non-linearity is applied to final layer of generator. Note that Generator and Discriminator are mirror images of each other.

### Generator

- `Input` = (batch size, 128) - *Noise from a Normal distribution*
- `Fully Connected Layer` = (batch size, 128, 4096)
- `Reshape` = (batch size, 256, 4, 4)
- `Transposed Convolution Layer` = (batch size, 128, 8, 8)
- `Transposed Convolution Layer` = (batch size, 64, 16, 16)
- `Transposed Convolution Layer` = (batch size, 3, 32, 32)

**Discriminator**

- `Input` = (batch size, 3, 32, 32)
- `Convolution Layer` = (batch size, 64, 16, 16)
- `Convolution Layer` = (batch size, 128, 8, 8)
- `Convolution Layer` = (batch size, 256, 4, 4)
- `Reshape` = (batch size, 4096)
- `Fully Connected Layer` = (batch size, 1)

## C BEGAN Architecture

The BEGAN architecture is illustrated in Fig. 20. The variable $n$ corresponds to the number of filters. In case of CelebA dataset the input images were of dimension $64 \times 64$ for which $n$ was made equal to 128, while for CIFAR-10 data $n = 64$ was used as the images were of dimension $32 \times 32$.

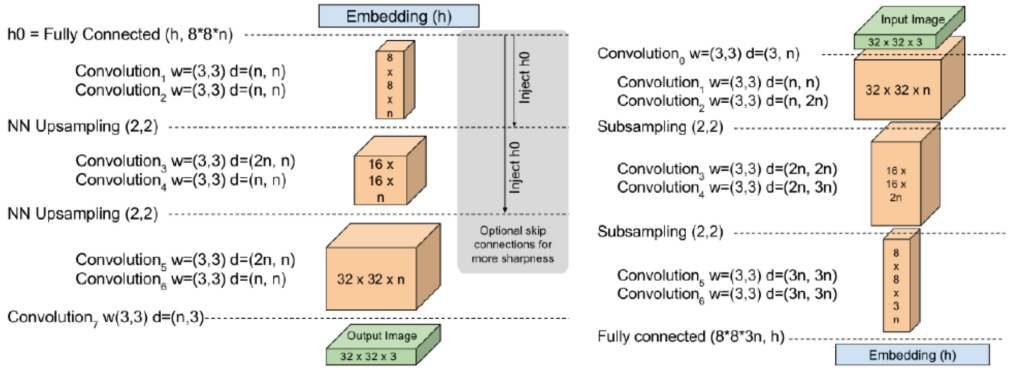



Figure 20: Network architecture of generator and discriminator in BEGAN.