# ADVERSARIAL LIPSCHITZ REGULARIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Generative adversarial networks (GANs) are one of the most popular approaches when it comes to training generative models, among which variants of Wasserstein GANs are considered superior to the standard GAN formulation in terms of learning stability and sample quality. However, Wasserstein GANs require the critic to be 1-Lipschitz, which is often enforced implicitly by penalizing the norm of its gradient, or by globally restricting its Lipschitz constant via weight normalization techniques. Training with a regularization term penalizing the violation of the Lipschitz constraint explicitly, instead of through the norm of the gradient, was found to be practically infeasible in most situations. With a novel generalization of Virtual Adversarial Training, called Adversarial Lipschitz Regularization, we show that using an explicit Lipschitz penalty is indeed viable and leads to competitive performance when applied to Wasserstein GANs, highlighting an important connection between Lipschitz regularization and adversarial training.

## 1 INTRODUCTION

In recent years, Generative adversarial networks (GANs) (Goodfellow et al., 2014) have been becoming the state-of-the-art in several generative modeling tasks, ranging from image generation (Karras et al., 2018) to imitation learning (Ho and Ermon, 2016). They are based on an idea of a two-player game, in which a discriminator tries to distinguish between real and generated data samples, while a generator tries to fool the discriminator, learning to produce realistic samples on the long run. Wasserstein GAN (WGAN) was proposed as a solution to the issues present in the original GAN formulation. Replacing the discriminator, WGAN trains a critic to approximate the Wasserstein distance between the real and generated distributions. This introduced a new challenge, as WGAN requires the function space of the critic to only consist of 1-Lipschitz functions.

To enforce the Lipschitz constraint on the WGAN critic, Arjovsky et al. (2017) originally used weight clipping, which was soon replaced by the much more effective method of Gradient Penalty (GP) (Gulrajani et al., 2017), which consists of penalizing the deviation of the critic's gradient norm from 1 at certain input points. Since then, several variants of gradient norm penalization have been introduced (Petzka et al., 2018; Wei et al., 2018; Adler and Lunz, 2018; Zhou et al., 2019b). As an alternative, a weight normalization technique called Spectral Normalization (SN) (Miyato et al., 2018) is a very efficient and simple method for enforcing a Lipschitz constraint on a per-layer basis.

Virtual Adversarial Training (VAT) (Miyato et al., 2017) is a semi-supervised learning method for regularizing neural networks. It is applied to improve the network's robustness against local perturbations of the input. Using an iterative method based on power iteration, it approximates the adversarial direction corresponding to certain input points. Perturbing an input towards its adversarial direction changes the network's output the most.

We propose a method called Adversarial Lipschitz Regularization (ALR) that can be seen as a generalization of VAT, that enables the training of neural networks with regularization terms penalizing the violation of the Lipschitz constraint explicitly, instead of through the norm of the gradient. It provides means to generate a pair for each input point, for which the Lipschitz constraint is likely to be violated with high probability. In general, enforcing Lipschitz continuity of complex models can be useful for a lot of applications. In this work, we focus on applying ALR to Wasserstein GANs, as regularizing or constraining Lipschitz continuity has proven to have a high impact on training stability and reducing mode collapse.

Our contributions are as follows:

- We show that VAT can be seen as a Lipschitz regularization method.
- We propose ALR, and apply it to penalize the violation of the Lipschitz constraint directly, resulting in Adversarial Lipschitz Penalty (ALP).
- Applying ALP on the critic in WGAN (WGAN-ALP), we show state-of-the-art performance in terms of Inception Score and Fréchet Inception Distance when trained on CIFAR-10.
- To demonstrate that ALP works in the high dimensional setting, we apply it on the critic in Progressive Growing GAN and show competitive performance when trained on CelebA-HQ.

## 2 BACKGROUND

### 2.1 WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) provide generative modeling by a generator network $g$ that transforms samples of a low-dimensional latent space $Z$ into samples from the data space $X$, transporting mass from a fixed noise distribution $P_Z$ to the generated distribution $P_g$. The generator is trained simultaneously with another network $f$ called the discriminator, which is trained to distinguish between fake samples drawn from $P_g$ and real samples drawn from the real distribution $P_r$, which is often represented by a fixed dataset. This network provides the learning signal to the generator, which is trained to generate samples that the discriminator considers real. This iterative process implements the minimax game

$$\min_g \max_f \mathbb{E}_{x\sim P_r} \log(f(x)) + \mathbb{E}_{z\sim P_Z} \log(1 - f(g(z))) \tag{1}$$

played by the networks $f$ and $g$. This training procedure minimizes the approximate Jensen-Shannon divergence (JSD) between $P_r$ and $P_g$ (Goodfellow et al., 2014). However, during training these two distributions might differ strongly or even have non-overlapping supports, which might result in gradients received by the generator that are unstable or zero (Arjovsky and Bottou, 2017).

Wasserstein GAN (WGAN) (Arjovsky et al., 2017) was proposed as a solution to this instability. Originating from Optimal Transport theory (Villani, 2008), the Wasserstein metric provides a distance between probability distributions with much better theoretical and practical properties than the JSD. It provides a smooth optimizable distance even if the two distributions have non-overlapping supports, which is not the case for JSD. It raises a metric $d_X$ from the space $X$ of the supports of the probability distributions $P_1$ and $P_2$ to the space of the probability distributions itself. For these purposes, the Wasserstein-$p$ distance requires the probability distributions to be defined on a metric space and is defined as

$$W_p(P_1, P_2) = \left( \inf_{\pi\in\Pi(P_1,P_2)} \mathbb{E}_{(x_1,x_2)\sim\pi} d_X(x_1, x_2)^p \right)^{\frac{1}{p}}, \tag{2}$$

where $\Pi(P_1, P_2)$ is the set of distributions on the product space $X \times X$ whose marginals are $P_1$ and $P_2$, respectively. The optimal $\pi$ achieving the infimum in (2) is called the optimal coupling of $P_1$ and $P_2$, and is denoted by $\pi^*$. The case of $p = 1$ has an equivalent formulation

$$W_1(P_1, P_2) = \sup_{\|f\|_L\leq 1} \mathbb{E}_{x\sim P_1} f(x) - \mathbb{E}_{x\sim P_2} f(x), \tag{3}$$

called the Kantorovich-Rubinstein formula (Villani, 2008), where $f : X \to \mathbb{R}$ is called the potential function, $\|f\|_L \leq 1$ is the set of all functions that are 1-Lipschitz with respect to the ground metric $d_X$, and the Wasserstein-1 distance corresponds to the supremum over all 1-Lipschitz potential functions. The smallest Lipschitz constant for a real-valued function $f$ with the metric space $(X, d_X)$ as its domain is given by

$$\|f\|_L = \sup_{x,y\in X; x\neq y} \frac{|f(x) - f(y)|}{d_X(x, y)}. \tag{4}$$

Based on (3), the critic in WGAN (Arjovsky et al., 2017) implements an approximation of the Wasserstein-1 distance between $P_g$ and $P_r$. The minimax game played by the critic $f$ and the generator $g$ becomes

$$\min_g \max_{\|f\|_L\leq 1} \mathbb{E}_{z\sim P_Z} f(g(z)) - \mathbb{E}_{x\sim P_r} f(x), \tag{5}$$

a formulation that proved to be superior to the standard GAN in practice, with substantially more stable training behaviour and improved sample quality. The challenge became effectively restricting the

smallest Lipschitz constant of the critic $f$, sparking the birth of a plethora of Lipschitz regularization techniques for neural networks.

### 2.2 LIPSCHITZ FUNCTION APPROXIMATION

A general definition of the smallest Lipschitz constant of a function $f : X \to Y$ is

$$\|f\|_L = \sup_{x,y \in X; x \neq y} \frac{d_Y(f(x), f(y))}{d_X(x, y)}, \tag{6}$$

where the metric spaces $(X, d_X)$ and $(Y, d_Y)$ are the domain and codomain of the function $f$, respectively. The function $f$ is called Lipschitz continuous if there exists a real constant $K \geq 0$ for which $d_Y(f(x), f(y)) \leq K \cdot d_X(x, y)$ for any $x, y \in X$. Then, the function $f$ is also called K-Lipschitz. Theoretical properties of K-Lipschitz neural networks with low values of $K$ were explored in Oberman and Calder (2018), showing that training neural networks with Lipschitz constraints is good for generalization and convergence.

Learning mappings with Lipschitz constraints became prevalent in the field of deep learning with the introduction of WGAN (Arjovsky et al., 2017). Enforcing the Lipschitz property on the critic was first done by clipping the weights of the network. This approach achieved superior results compared to the standard GAN formulation, but still sometimes yielded poor quality samples or even failed to converge. While clipping the weights enforces a global Lipschitz constant, it also reduces the function space, which might not include the optimal critic any more.

Soon this method has been replaced by a softened one called Gradient Penalty (GP) (Gulrajani et al., 2017). Motivated by the fact that the optimal critic should have unit gradient norm on lines connecting the coupled points $(x_1, x_2) \sim \pi^*$ according to (2), they proposed a regularizer that enforces unit gradient norm along these lines, which not only enforces the Lipschitz constraint, but other properties of the optimal solution as well. However, $\pi^*$ is not known in practice, which is why (Gulrajani et al., 2017) proposed to apply GP on samples of the induced distribution $P_i$, by interpolating samples from the marginals $P_1$ and $P_2$. The critic in the WGAN-GP formulation is regularized with the loss

$$\lambda \mathbb{E}_{x \sim P_i} (\|\nabla_x f(x)\|_2 - 1)^2 \tag{7}$$

where $P_i$ denotes the distribution of samples obtained by interpolating pairs of samples drawn from $P_r$ and $P_g$, and $\lambda$ is a hyperparameter acting as a Lagrange multiplier.

Theoretical arguments against GP were pointed out by Petzka et al. (2018), arguing that unit gradient norm on samples of the distribution $P_i$ is not valid, as the pairs of samples being interpolated are generally not from the optimal coupling $\pi^*$, and thus do not necessarily need to match gradient norm 1. Furthermore, they point out that differentiability assumptions of the optimal critic are not met. Therefore, the regularizing effect of GP might be too strong. As a solution, they suggested using a loss penalizing the violation of the Lipschitz constraint either explicitly with

$$\lambda \mathbb{E}_{x,y \sim P_\tau} \left( \frac{|f(x) - f(y)|}{\|x - y\|_2} - 1 \right)_+^2 \tag{8}$$

or implicitly with

$$\lambda \mathbb{E}_{x \sim P_\tau} (\|\nabla_x f(x)\|_2 - 1)_+^2 \tag{9}$$

where in both cases $(a)_+$ denotes $\max(0, a)$. The first method has only proved viable when used on toy datasets, and led to considerably worse results on relatively more complex datasets like CIFAR-10, which is why Petzka et al. (2018) used the second one, which they termed Lipschitz Penalty (LP). Compared to GP, this term only penalizes the gradient norm when it exceeds 1. As $P_\tau$ they evaluated the interpolation method described above, and also sampling random local perturbations of real and generated samples, but found no significant improvement compared to $P_i$. Wei et al. (2018) proposed dropout in the critic as a way for creating perturbed input pairs to evaluate the explicit Lipschitz penalty (8), which led to improvements, but still relied on using GP simultaneously.

One of the strengths of the Wasserstein distance is that it can be defined with any metric $d_X$, a fact that Adler and Lunz (2018) built on by proposing Banach WGAN (BWGAN), which generalizes WGAN to separable Banach spaces. They resort to these spaces because in order to use GP, they need a tractable dual metric on the topological dual of $X$. This approach brought considerable improvements,

and Adler and Lunz (2018) emphasized the fact that through explicit Lipschitz penalties one could extend WGANs to general metric spaces as well.

A second family of Lipschitz regularization methods is based on weight normalization, restricting the Lipschitz constant of a network globally instead of only at points of the input space. One such technique is called spectral normalization (SN) proposed in Miyato et al. (2018), which is a very efficient and simple method for enforcing a Lipschitz constraint with respect to the 2-norm on a per-layer basis, applicable to neural networks consisting of affine layers and K-Lipschitz activation functions. Gouk et al. (2018) proposed a similar approach, which can be used to enforce a Lipschitz constraint with respect to the 1-norm and $\infty$-norm in addition to the 2-norm, while also being compatible with batch normalization and dropout. Anil et al. (2018) argued that any Lipschitz-constrained neural network must preserve the norm of the gradient during backpropagation, and to this end proposed another weight normalization technique, showing that it compares favorably to SN, and an activation function based on sorting.

### 2.3 Virtual Adversarial Training

VAT (Miyato et al., 2017) is a semi-supervised learning method that is able to regularize networks to be robust to local adversarial perturbation. Virtual adversarial perturbation means perturbing input sample points in such a way that the change in the output of the network induced by the perturbation is maximal in terms of a distance between distributions. This defines a direction for each sample point called the virtual adversarial direction, in which the perturbation is performed. It is called virtual to make the distinction with the adversarial direction introduced in Goodfellow et al. (2015) clear, as VAT uses unlabeled data with virtual labels, assigned to the sample points by the network being trained. The regularization term of VAT is called Local Distributional Smoothness (LDS). It is defined as

$$L_{LDS} = D\left(p(y|x), p(y|x + r_{vadv})\right), \tag{10}$$

where $p$ is a conditional distribution implemented by a neural network, $D(p, p')$ is a divergence between two distributions $p$ and $p'$, for which Miyato et al. (2017) chose the Kullback-Leibler divergence (KLD), and

$$r_{vadv} = \underset{\|r\|_2 \leq \epsilon}{\arg\max}\, D\left(p(y|x), p(y|x + r)\right) \tag{11}$$

is the virtual adversarial perturbation, where $\epsilon$ is a hyperparameter. VAT is defined as a training method with the regularizer (10) applied to labeled and unlabeled examples. An important detail is that (10) is minimized by keeping $p(y|x)$ fixed and optimizing $p(y|x + r_{vadv})$ to be close to it.

The adversarial perturbation is approximated by the power iteration

$$r_{vadv} \approx \epsilon \frac{r_k}{\|r_k\|_2}, \tag{12}$$

where

$$r_{i+1} \approx \frac{\nabla_r D\left(p(y|x), p(y|x + r)\right)\Big|_{r=\xi r_i}}{\left\|\nabla_r D\left(p(y|x), p(y|x + r)\right)\Big|_{r=\xi r_i}\right\|_2}, \tag{13}$$

$r_0$ is a randomly sampled unit vector and $\xi$ is another hyperparameter. This iterative scheme is an approximation of the direction at $x$ that induces the greatest change in the output of $p$ in terms of the divergence $D$. Miyato et al. (2017) found that $k = 1$ iteration is enough in practical situations.

## 3 Virtual Adversarial Training as Lipschitz Regularization

VAT was defined by considering neural networks implementing conditional distributions $p(y|x)$, where the distribution over discrete labels $y$ was conditioned on the input image $x$ Miyato et al. (2017). To see why LDS (10), the regularization term of VAT, can be seen as special kind of Lipschitz continuity, we will use a different perspective. Consider a mapping $f : X \to Y$ with domain $X$ and codomain $Y$, where $X$ is the space of images and $Y$ is the space of distributions on the space of labels. Since a divergence (KLD in particular) is a premetric (prametric, quasi-distance) on the space of probability measures (Deza and Deza, 2009), $Y$ can be considered a premetric space $(Y, d_Y)$,

where $d_Y$ is the divergence $D$ from the VAT formulation. Let us metrize the space of images $X$ with the trivial metric

$$d_X(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

From this perspective, the network $f$ is a mapping from the metric space $(X, d_X)$ to the premetric space $(Y, d_Y)$. Although Lipschitz continuity is usually defined for mappings between metric spaces, nothing stops us from defining the Lipschitz constant (6) with $(Y, d_Y)$ being a premetric space. In fact, the only restriction is the possible division by zero if $d_X(x, y) = 0$ for some $x \neq y$, so we could consider premetrics for $d_X$ as well. Let us rewrite (6) with $y = x + r$ to get

$$\|f\|_L = \sup_{x, x+r \in X; 0 < d_X(x, x+r)} \frac{d_Y(f(x), f(x+r))}{d_X(x, x+r)}. \tag{15}$$

A given mapping $f$ is K-Lipschitz if and only if for any given $x \in X$, taking the supremum over $r$ in (15) results in a value $K$ or smaller. Assuming that this supremum is always achieved for some $r$, we can define a notion of adversarial perturbation with respect to the Lipschitz continuity as

$$r_{adv} = \arg\max_{x+r \in X; 0 < d_X(x, x+r)} \frac{d_Y(f(x), f(x+r))}{d_X(x, x+r)}, \tag{16}$$

and the corresponding maximal violation of the K-Lipschitz constraint at $x$ as

$$L_{ALP} = \left( \frac{d_Y(f(x), f(x + r_{adv}))}{d_X(x, x + r_{adv})} - K \right)_+. \tag{17}$$

In the setting currently being considered, $d_Y$ is D from VAT and $d_X$ is the trivial metric. Let us also assume that we aim at learning a mapping $f$ with the smallest possible $\|f\|_L$ by setting $K$ to 0, making (16) and (17) reduce to

$$r_{adv} = \arg\max_{x+r \in X; r \neq 0} D(f(x), f(x+r)) \tag{18}$$

and

$$L_{ALP} = D(f(x), f(x + r_{adv})), \tag{19}$$

respectively.

One can immediately see that (19) is exactly LDS (10), and (18) is quite similar to (11) as well. The difference is the constraint $\|r\|_2 \leq \epsilon$ in (11), but let us consider that as an implementation detail, as well as the question of keeping $f(x)$ fixed when minimizing (19). With these discrepancies aside, we have recovered VAT as a special case of Lipschitz regularization.

## 4 Adversarial Lipschitz Regularization

We define Adversarial Lipschitz Regularization (ALR) as the method of adding a regularization term to the training objective that penalizes the violation of the Lipschitz constraint evaluated at sample pairs obtained by adversarial perturbation. We call this term Adversarial Lipschitz Penalty (ALP) and define it as

$$L_{ALP} := \left( \frac{d_Y(f(x), f(x + r_{adv}))}{d_X(x, x + r_{adv})} - K \right)_+ \tag{20}$$

where

$$r_{adv} = \arg\max_{x+r \in X; 0 < d_X(x, x+r)} \frac{d_Y(f(x), f(x+r))}{d_X(x, x+r)} \tag{21}$$

is the adversarial perturbation with respect to the Lipschitz constraint, $f : X \to Y$ is a neural network and $K$ is the Lipschitz constant that we'd like to enforce. $d_X$ and $d_Y$ are premetrics on the domain $X$ and codomain $Y$ of $f$, respectively.

To put it in words, ALP measures the deviation of $f$ from being K-Lipschitz evaluated at pairs of sample points where one is the adversarial perturbation of the other. If added to the training objective, it makes the learned mapping approximately K-Lipschitz around the sample points it is applied at. We found that it is best to minimize (20) without keeping $f(x)$ fixed. See Appendix A.3 for the semi-supervised case.

### 4.1 APPROXIMATION OF $r_{adv}$

Similarly to VAT, the adversarial perturbation (21) is approximated using the power iteration by

$$r_{vadv} \approx \epsilon \frac{r_k}{\|r_k\|_2}, \tag{22}$$

where

$$r_{i+1} \approx \frac{\nabla_r d_Y \left(f(x), f(x+r)\right)\Big|_{r=\xi r_i}}{\left\|\nabla_r d_Y \left(f(x), f(x+r)\right)\Big|_{r=\xi r_i}\right\|_2}, \tag{23}$$

is the approximated adversarial direction with $r_0$ being a randomly sampled unit vector. Unlike in VAT, we do not fix $\epsilon$, but draw it randomly from a predefined distribution $P_\epsilon$ over $\mathbb{R}_+$ to apply the penalty at different scales. The hyperparameters of the approximation scheme are therefore $k$, $\xi$ and those of $P_\epsilon$. The derivation of this formula is essentially the same as the one described in Miyato et al. (2017), but is included in Appendix A.1 for completeness.

### 4.2 COMPARISON WITH OTHER LIPSCHITZ REGULARIZATION TECHNIQUES

Theoretically, ALR can be used with all kinds of premetrics $d_X$ and $d_Y$, and any kind of model $f$, but the approximation of $r_{adv}$ described above imposes a practical restriction. It approximates the adversarial perturbation of $x$ as a translation with length $\epsilon$ with respect to the 2-norm in the adversarial direction, which is only a perfect approximation if the ratio in (20) is constant for any $\epsilon > 0$. This idealized setting is hardly ever the case, which is why we see the search for other approximation schemes as an important future direction. There is a large number of methods for generating adversarial examples besides the one proposed in VAT, which could possibly be combined with ALR either to improve the approximation or to make it possible with new kinds of metrics.

In terms of efficiency when applied to WGANs, ALR compares favorably to the implicit methods penalizing the gradient norm, and to weight normalization techniques as well, as demonstrated in the experiments section. Adler and Lunz (2018) argued that penalizing the norm of the gradient as in (9) is more effective than penalizing the Lipschitz quotient directly as in (8), as the former penalizes the slope of $f$ in all spatial directions around $x$, unlike the latter, which does so only along $(x - y)$. We hypothesize that using the explicit Lipschitz penalty in itself is insufficient because if one takes pairs of samples $x, y$ randomly from $P_r$, $P_g$ or $P_i$ (or just one sample and generates a pair for it with random perturbation), the violation of the Lipschitz penalty evaluated at these sample pairs will be far from its maximum, hence a more sophisticated strategy for sampling pairs is required. As we will show, a carefully chosen sampling strategy can in fact make the explicit penalty favorable over the implicit one.

To showcase the differences between weight normalization methods, implicit penalty methods and explicit penalty methods, represented by SN, LP and ALR, respectively, we devised the following toy example. Suppose that we want to approximate the following real-valued mapping on the 2-dimensional interval $[-4, 4]^2$:

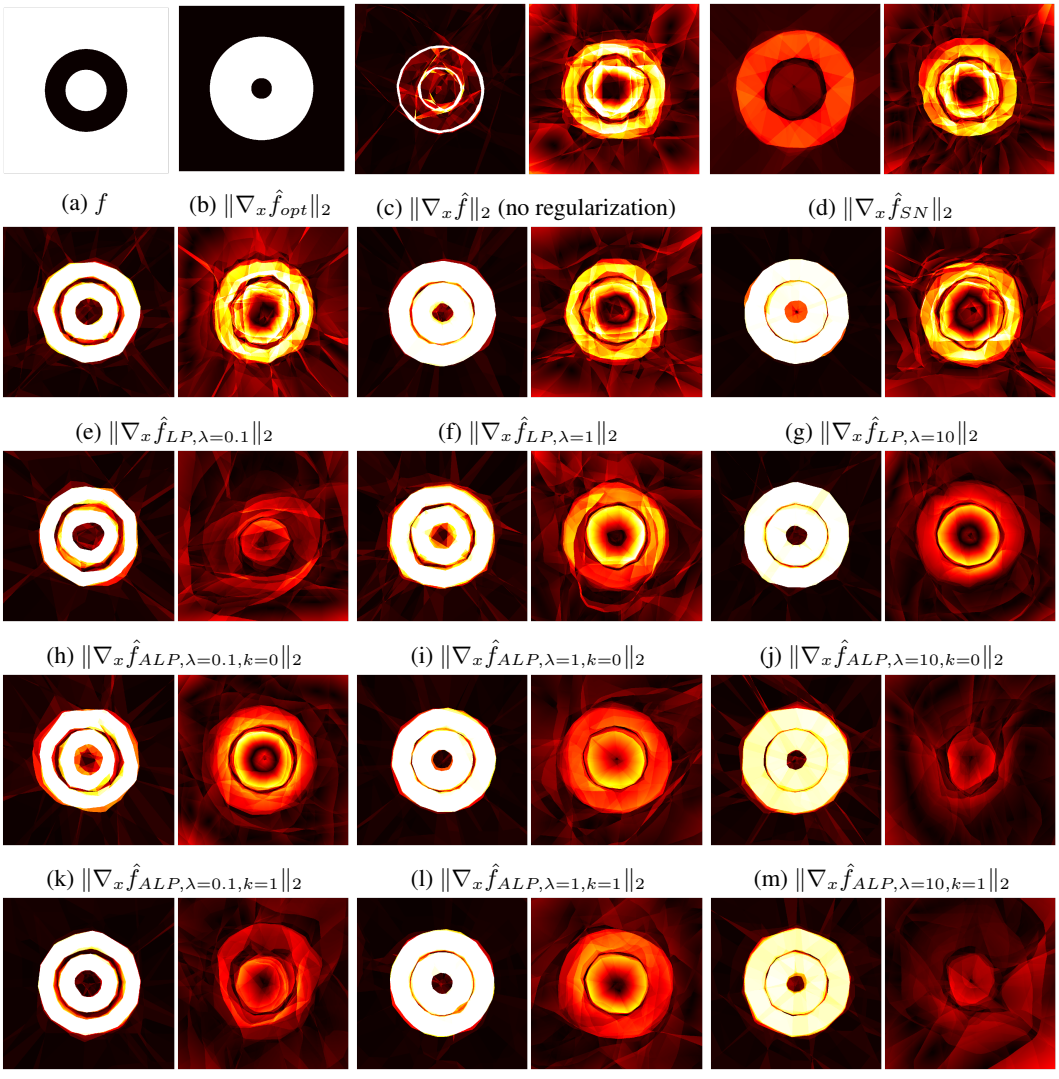
$$f(x, y) = \begin{cases} 0 & \text{if } 1 \leq \sqrt{x^2 + y^2} \leq 2, \\ 1 & \text{otherwise} \end{cases} \tag{24}$$

for $-4 \leq x, y \leq 4$. In addition, we want the approximation to be 1-Lipschitz. It is easy to see that the optimal approximation is

$$\hat{f}_{opt}(x, y) = \begin{cases} 1 & \text{if } \sqrt{x^2 + y^2} \leq 0.5, \\ 1.5 - \sqrt{x^2 + y^2} & \text{if } 0.5 < \sqrt{x^2 + y^2} <= 1.5, \\ \sqrt{x^2 + y^2} - 1.5 & \text{if } 1.5 < \sqrt{x^2 + y^2} <= 2.5, \\ 1 & \text{otherwise.} \end{cases} \tag{25}$$

This example has connections to WGAN, as the optimal critic is 1-Lipschitz, and its approximation will provide the learning signal to the generator in the form of gradients. Therefore, it is important to closely approximate the gradient of the optimal critic, which is achieved indirectly by Lipschitz

regularization. In this example, we will see how closely the different Lipschitz regularization methods can match the optimal approximation $\hat{f}_{opt}$ and its gradient. We implemented the example in PyTorch. For the approximation $\hat{f}$, we use an MLP with 3 hidden layers containing 20, 40 and 20 neurons, respectively, with ReLU activations after the hidden layers, and a variant which also has batch normalization (BN) before the activations. We trained the networks for $2^{14}$ iterations, with batches consisting of an input, a corresponding output, and an additional input for regularization. The inputs are drawn uniformly at random from $[-4, 4]^2$ and the output is defined by (24). The minibatch size was $64$ for input-output pairs, and $1024$ for regularization inputs. We used heatmaps to visualize the target $f$ and the gradient norm surfaces of the optimal and learned mappings, with the color gradient going from black at $0$ to white at $1$, see Figure 1. This example is not intended to rank the competing Lipschitz regularization methods, as it always depends on the particular application which one is the best suited, but to show that they are fundamentally different and competent in their own way.



(a) $f$ (b) $\|\nabla_x \hat{f}_{opt}\|_2$ (c) $\|\nabla_x \hat{f}\|_2$ (no regularization) (d) $\|\nabla_x \hat{f}_{SN}\|_2$

(e) $\|\nabla_x \hat{f}_{LP,\lambda=0.1}\|_2$ (f) $\|\nabla_x \hat{f}_{LP,\lambda=1}\|_2$ (g) $\|\nabla_x \hat{f}_{LP,\lambda=10}\|_2$

(h) $\|\nabla_x \hat{f}_{ALP,\lambda=0.1,k=0}\|_2$ (i) $\|\nabla_x \hat{f}_{ALP,\lambda=1,k=0}\|_2$ (j) $\|\nabla_x \hat{f}_{ALP,\lambda=10,k=0}\|_2$

(k) $\|\nabla_x \hat{f}_{ALP,\lambda=0.1,k=1}\|_2$ (l) $\|\nabla_x \hat{f}_{ALP,\lambda=1,k=1}\|_2$ (m) $\|\nabla_x \hat{f}_{ALP,\lambda=10,k=1}\|_2$

(n) $\|\nabla_x \hat{f}_{ALP,\lambda=0.1,k=5}\|_2$ (o) $\|\nabla_x \hat{f}_{ALP,\lambda=1,k=5}\|_2$ (p) $\|\nabla_x \hat{f}_{ALP,\lambda=10,k=5}\|_2$

Figure 1: Target $f$, and gradient norm surfaces of optimal and learned approximations of $f$

Without any kind of regularization, the network learned to approximate the target function very well, but its gradients look nothing like that of $\hat{f}_{opt}$, although it is a better match with BN.

When we apply SN to the MLP layers, the result without BN will be a very smooth mapping with maximum gradient norm $\approx 0.59$. SN does not work very well with BN, the result being only slightly

better than the unregularized case. It can be seen that SN has a very strong regularization effect, which is because SN works by approximating the spectral norm of each layer and then normalizing the layers by dividing their weight matrices by the corresponding spectral norms, thereby resulting in overregularization if the approximation is greater than the actual spectral norm. A detail not visible here is that because SN is not a sample-based penalty, but is based on layer-wise weight normalization, it regularizes globally instead of around actual data samples. In this case, on the whole of $\mathbb{R}^2$ instead of just $[-4, 4]^2$. If the input space consists of $32 * 32$ RGB images with pixel values in $[-1, 1]$ (which is the case for WGANs trained on CIFAR-10), it means the trained mapping is regularized on $\mathbb{R}^{32*32*3}$ instead of just $[-1, 1]^{32*32*3}$, which can be unintended.

When the network is regularized using LP (9), the regularization strength can be controlled by tuning the value of $\lambda$. We trained with $\lambda = 0.1, 1$ and 10. Without BN, the highest of these values seems to work the best, although the middle part has relatively high gradient norms. With BN, the resulting mapping is visibly highly irregular.

With ALR, in addition to $\lambda$, we have additional control over the regularization by the hyperparameters of the approximation scheme of $r_adv$. After some experimentation, we have found the best $P_\epsilon$ for this case was drawing a sample from a 0-centered Gaussian with standard deviation $10^{-6}$, taking its absolute value incremented by $10^{-6}$, and then setting all values above $10^{-5}$ to $10^{-5}$, thereby getting values from $[10^{-6}, 10^{-5}]$ with the density increasing towards the lower end. We trained with $\lambda = 0.1, 1$ and 10, and $k = 0, 1$ and 5 power iterations. Arguably, both with and without BN the $\lambda = 1$ and $k = 5$ case seems like the best choice. Without BN, the results are quite similar to the LP case, but when BN is introduced, the resulting mappings are smoother than the ones obtained with LP.

The performance of ALR mostly depends on the speed of the approximation of $r_{adv}$. The current method requires 1 step of backpropagation for each power iteration step, which means that running time will be similar to that of LP and GP with $k = 1$. SN is much cheaper computationally than each penalty method, although we believe ALR has the potential to become relatively cheap as well by adopting new techniques for obtaining adversarial examples (Shafahi et al., 2019).

## 5 WGAN-ALP

We specialize the ALP formula (20) with $f$ being the critic, $d_X(x, y) = \|x-y\|_2$, $d_Y(x, y) = |x-y|$ and $K = 1$, and apply it to the WGAN objective to arrive at a version with the explicit penalty, which uses adversarial perturbations as a sampling strategy. It is formulated as

$$\mathbb{E}_{z\sim P_Z} f(g(z)) - \mathbb{E}_{x\sim P_r} f(x) + \lambda \mathbb{E}_{x\sim P_{r,g}} \left( \frac{|f(x) - f(x + r_{adv})|}{\|r_{adv}\|_2} - 1 \right)_+^2, \tag{26}$$

where $P_{r,g}$ is a combination of the real and generated distributions (meaning that a sample $x$ can come from both), $\lambda$ is the Lagrange multiplier, and the adversarial perturbation is defined as

$$r_{adv} = \underset{r;0<\|r\|_2}{\arg\max} \frac{|f(x) - f(x + r)|}{\|r\|_2}. \tag{27}$$

This formulation of WGAN results in a stable explicit Lipschitz penalty, overcoming the difficulties experienced when one tries to apply it to random sample pairs as shown in Petzka et al. (2018).

To evaluate the performance of WGAN-ALP, we trained one on CIFAR-10, consisting of $32 * 32$ RGB images, using the residual architecture from Gulrajani et al. (2017), implemented in TensorFlow. Closely following Gulrajani et al. (2017), we used the Adam optimizer (Kingma and Ba, 2015) with parameters $\beta_1 = 0$, $\beta_2 = 0.9$ and an initial learning rate of $2 \cdot 10^{-4}$ decaying linearly to 0 over 100000 iterations, training the critic for 5 steps and the generator for 1 per iteration with minibatches of size 64 (doubled for the generator). We used (26) as a loss function to optimize the critic. $K = 1$ was an obvious choice, and we found $\lambda = 100$ to be optimal (the training diverged for $\lambda = 0.1$, and was stable but performed worse for $\lambda = 10$ and 1000). The hyperparameters of the approximation of $r_{adv}$ were set to $\xi = 10$, $P_\epsilon$ being the uniform distribution over $[0.1, 10]$, and $k = 1$ power iteration. Both batches from $P_r$ and $P_g$ were used for regularization.

We monitored the Inception Score and FID during training using 10000 samples every 1000 iteration, and evaluated them at the end of training using 50000 samples. To measure the performance of

WGAN-ALP, we ran the training setting described above 5 times with different random seeds, and calculated the mean, standard deviation and maximum of the final Inception Scores and FIDs, which we report for WGAN-ALP and other relevant GANs (Gulrajani et al., 2017; Petzka et al., 2018; Zhou et al., 2019a; Wei et al., 2018; Miyato et al., 2018; Adler and Lunz, 2018; Karras et al., 2018) in Table 1. Competing variants reported either or both of the average and the best Inception Scores in the corresponding papers, which is why we chose to report both. Both the Inception Score and FID achieved is state of the art for non-progressive growing methods. We show some generated samples in Figure 2.

Table 1: Inception Scores and FIDs on CIFAR-10

| | Inception Score | | |
|---|---|---|---|
| Method | Average | Best | FID |
| WGAN-GP | $7.86 \pm .07$ | | $18.86 \pm .13$ |
| WGAN-LP | $8.02 \pm .07$ | | |
| LGAN | $8.03 \pm .03$ | | $15.64 \pm .07$ |
| CT-GAN | $8.12 \pm .12$ | | |
| SN-GAN | $8.22 \pm .05$ | | $21.70 \pm .21$ |
| BWGAN | $8.31 \pm .07$ | | $16.43$ |
| Progressive GAN | $8.56 \pm .06$ | $8.80$ | |
| **WGAN-ALP (ours)** | $\mathbf{8.37 \pm .04}$ | $\mathbf{8.59}$ | $\mathbf{13.01 \pm .37}$ |

Figure 2: Generated CIFAR-10 samples

To show that ALR works in a high-dimensional setting as well, we trained a Progressive GAN on the CELEBA-HQ dataset (Karras et al., 2018), consisting of $1024 * 1024$ RGB images. The best FID seen during training with the original GP version was $8.69$, while for the modified ALP version it was $14.65$. The example shows that while ALP did not beat GP in this case (although it possibly could, given more engineering hours), it does work in the high-dimensional setting as well. For details and generated samples see Appendix A.2.

## 6 Conclusions

Derived as a generalization of VAT, we have shown that ALR is an efficient and powerful method for learning Lipschitz constrained mappings implemented by neural networks. Resulting in competitive performance when applied to the training of WGANs, ALR is a generally applicable regularization method. It draws an important parallel between Lipschitz regularization and adversarial training, which we believe can prove to be a fruitful line of future research.

## REFERENCES

J. Adler and S. Lunz. Banach wasserstein gan. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6754–6763. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7909-banach-wasserstein-gan.pdf.

C. Anil, J. Lucas, and R. Grosse. Sorting out lipschitz function approximation. *CoRR*, abs/1811.05381, 2018. URL http://arxiv.org/abs/1811.05381.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=Hk4_qw5xe.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

M. Deza and E. Deza. *Encyclopedia of Distances*. Encyclopedia of Distances. Springer Berlin Heidelberg, 2009. ISBN 9783642002342.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *CoRR*, abs/1804.04368, 2018. URL http://arxiv.org/abs/1804.04368.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, pages 529–536, 2004.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5769–5779, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295327.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *NIPS*, pages 4565–4573, 2016.

A. Householder. *The Theory of Matrices in Numerical Analysis*. A Blaisdell book in pure and applied sciences : introduction to higher mathematics. Blaisdell Publishing Company, 1964.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

T. Miyato, S. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *CoRR*, abs/1704.03976, 2017. URL http://arxiv.org/abs/1704.03976.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

A. M. Oberman and J. Calder. Lipschitz regularized deep neural networks converge and generalize. *CoRR*, abs/1808.09540, 2018. URL http://arxiv.org/abs/1808.09540.

H. Petzka, A. Fischer, and D. Lukovnikov. On the regularization of wasserstein gans. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=B1hYRMbCW.

A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *CoRR*, abs/1904.12843, 2019. URL http://arxiv.org/abs/1904.12843.

C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. ISBN 9783540710509.

X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=SJx9GQb0-.

Z. Zhou, J. Liang, Y. Song, L. Yu, H. Wang, W. Zhang, Y. Yu, and Z. Zhang. Lipschitz generative adversarial nets. *CoRR*, abs/1902.05687, 2019a. URL http://arxiv.org/abs/1902.05687.

Z. Zhou, J. Shen, Y. Song, W. Zhang, and Y. Yu. Towards efficient and unbiased implementation of lipschitz continuity in gans. *CoRR*, abs/1904.01184, 2019b. URL http://arxiv.org/abs/1904.01184.

# A Appendix

## A.1 Derivation of the approximation of $r_{adv}$

We assume that $f$ and $d_Y$ are both twice differentiable with respect to their arguments almost everywhere, the latter specifically at $x = y$. Note that one can easily find a $d_Y$ for which the last assumption does not hold, for example the $l1$ distance. If $d_Y$ is translation invariant, meaning that $d_Y(x, y) = d_Y(x + u, y + u)$ for each $u \in Y$, then its subderivatives at $x = y$ will be independent of $x$, hence the method described below will still work. Otherwise, one can resort to using a proxy metric in place of $d_Y$ for the approximation, for example the $l2$ distance.

We denote $d_Y(f(x), f(x + r))$ by $d(r, x)$ for simplicity. Because $d(r, x) \geq 0$ and $d(0, x) = 0$, it is easy to see that

$$\nabla_r d(r, x)\big|_{r=0} = 0, \tag{28}$$

so that the second-order Taylor approximation of $d(r, x)$ is $d(r, x) \approx \frac{1}{2} r^T H(x) r$, where $H(x) = \nabla\nabla_r d(r, x)\big|_{r=0}$ is the Hessian matrix. The eigenvector $u$ of $H(x)$ corresponding to its eigenvalue with the greatest absolute value is the direction of greatest curvature, which is approximately the adversarial direction that we are looking for. The power iteration (Householder, 1964) defined by

$$r_{i+1} := \frac{H(x) r_i}{\|H(x) r_i\|_2}, \tag{29}$$

where $r_0$ is a randomly sampled unit vector, converges to $u$ if $u$ and $r_0$ are not perpendicular. Calculating $H(x)$ is computationally heavy, which is why $H(x)r_i$ is approximated using the finite differences method as

$$H(x) r_i \approx \frac{\nabla_r d(r, x)\big|_{r=\xi r_i} - \nabla_r d(r, x)\big|_{r=0}}{\xi} = \frac{\nabla_r d(r, x)\big|_{r=\xi r_i}}{\xi} \tag{30}$$

where the equality follows from (28). The hyperparameter $\xi \neq 0$ is introduced here. In summary, the adversarial direction is approximated by the iterative scheme

$$r_{i+1} := \frac{\nabla_r d(r, x)\big|_{r=\xi r_i}}{\left\|\nabla_r d(r, x)\big|_{r=\xi r_i}\right\|_2}, \tag{31}$$

of which one iteration is found to be sufficient and necessary in practice.

### A.2 PROGRESSIVE GAN

We took the official TensorFlow implementation and modified the loss function of the critic, which originally used GP, with a version of ALP. The same setup used on CIFAR-10 was stable until the last stage of progressive growing, but to make it work on the highest resolution, we had to replace it with

$$\begin{aligned}&\mathbb{E}_{z\sim P_Z} f(g(z)) - \mathbb{E}_{x\sim P_r} f(x) \\ &\quad + \lambda \mathbb{E}_{x\sim P_{r,g}} \left( \left( \frac{|f(x) - f(x + r_{adv})|}{\|r_{adv}\|_2} - 1 \right)_+^2 + \left( \frac{|f(x) - f(x + r_{adv})|}{\|r_{adv}\|_2} - 1 \right)_+ \right),\end{aligned} \tag{32}$$

meaning that we used the sum of the absolute and squared values of the Lipschitz constraint violation as the penalty. The optimal hyperparameters were $\lambda = 0.1$, $P_\epsilon$ being the uniform distribution over $[0.1, 100]$, $\xi = 10$ and $k = 1$ step of power iteration. The best FID seen during training with the original GP version was $8.69$, while for the modified ALP version it was $14.65$. The example shows that while ALP did not beat GP in this case (although it possibly could given more engineering hours), it does work in the high-dimensional setting as well. Some generated samples can be seen in Figure 3 for ALR and Figure 4 for the original GP version.

### A.3 SEMI-SUPERVISED LEARNING

Since VAT is a semi-supervised learning method, it is important to see how ALR fares in that regime. To show this, we have replicated one of the experiments from Miyato et al. (2017). We trained the ConvLarge architecture to classify images from CIFAR-10 with the same setting as described in Miyato et al. (2017), except that we did not decay the learning rate, but kept it fixed at $3e - 4$. We split the $50000$ training examples into $4000$ samples for the classification loss, $45000$ samples for regularization and $1000$ for validation, with equally distributed classes. Test performance was evaluated on the $10000$ test examples. We have found that unlike in the unsupervised setting, here it was important to assume $f(x)$ fixed when minimizing the regularization loss, and also to complement the smoothing effect with entropy minimization (Grandvalet and Bengio, 2004). The baseline VAT method was ALR specialized with $K = 0$, $d_X$ the trivial metric (14), $d_Y$ being the KL divergence, fixed $\epsilon = 8$ and $\lambda = 1$. This setting achieved maximal validation performance of $84.2\%$ and test performance $82.46\%$. After some experimentation, the best performing choice was $K = 0$, $d_X$ being the $l_2$ metric, $d_Y$ the mean squared difference over the logit space (which parametrize the categorical output distribution over which the KL divergence is computed in the case of VAT), $P_\epsilon$ being the uniform distribution over $[1, 10]$ and $\lambda = 1$. This way the maximal validation performance was $85.3\%$ and test performance $83.54\%$. Although this $\approx 1\%$ is improvement is not very significant, it shows that ALR can be a competitive choice as a semi-supervised learning method as well.

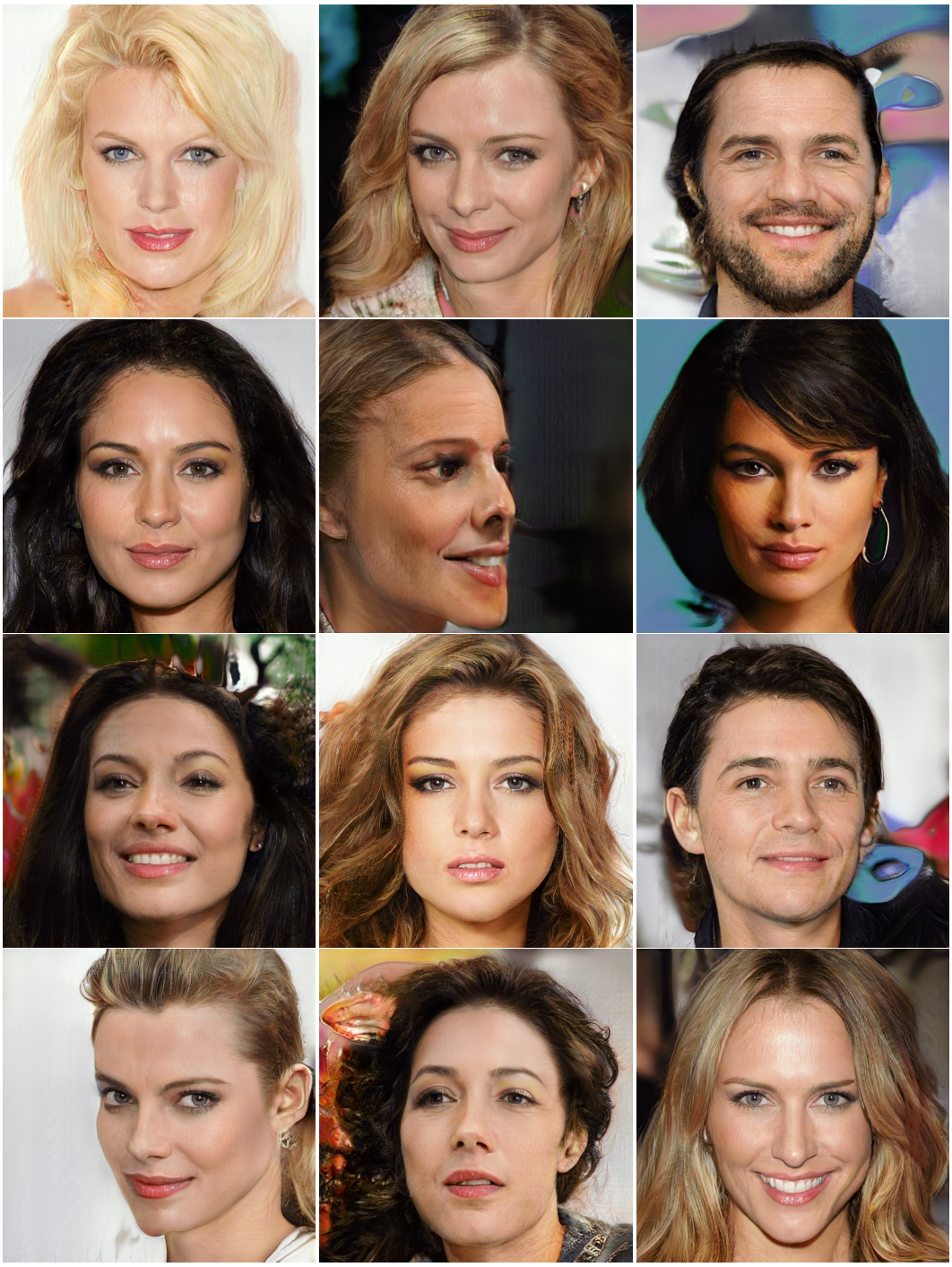


Figure 3: Images generated using Progressive GAN trained with ALR

Figure 4: Images generated using Progressive GAN trained with GP