Improving weight clipping in Wasserstein GANs

Estelle Massart
ICTEAM, Université catholique de Louvain,
Avenue Georges Lemaitre, 4/L4.05.01
1348 Louvain-la-Neuve, Belgium
Email: estelle.massart@uclouvain.be

***Abstract*—Weight clipping is a well-known strategy to keep the Lipschitz constant of the critic under control, in Wasserstein GAN training. After each training iteration, all parameters of the critic are clipped to a given box, impacting the progress made by the optimizer. In this work, we propose a new strategy for weight clipping in Wasserstein GANs. Instead of directly clipping the parameters, we first obtain an equivalent model that is closer to the clipping box, and only then clip the parameters. Our motivation is to decrease the impact of the clipping strategy on the objective, at each iteration. This equivalent model is obtained by following invariant curves in the critic loss landscape, whose existence is a consequence of the positive homogeneity of common activations: rescaling the input and output signals to each activation by inverse factors preserves the loss. We provide preliminary experiments showing that the proposed strategy speeds up training on Wasserstein GANs with simple feed-forward architectures.**

I. Introduction

Generative adversarial networks (GANs) have achieved impressive performance in a wide range of applications including realistic image generation [1], video [2], music [3] and text generation [4]. These models, initially proposed in [5], aim to generate points distributed according to an unknown target data distribution $\mathbb{P}_r$ supported on some unknown low-dimensional data manifold in $\mathbb{R}^N$, which makes alternative approaches such as variational sampling out of reach. GANs rely on a game between two players, the *discriminator* and the *generator*, that compete for conflicting objectives. The generator aims to mimic $\mathbb{P}_r$ by producing realistic samples, while the discriminator learns to distinguish the synthetic points returned by the generator from true data points sampled from $\mathbb{P}_r$, thereby enforcing the generator to better imitate the target data distribution.

In practice, the generator is often chosen as a neural network $G : \mathbb{R}^n \to \mathbb{R}^N$ that takes as input some low-dimensional noise vector with distribution $\mathbb{P}_z$ (e.g., $\mathbb{P}_z$ is a uniform or standard Gaussian distribution in $\mathbb{R}^n$, with $n \ll N$) and outputs points in $\mathbb{R}^N$ that should be approximately distributed along the target data distribution $\mathbb{P}_r$. The discriminator is a neural network $D : \mathbb{R}^N \to [0, 1]$ that takes as input some x and outputs a scalar specifying whether x is likely to be produced by the generator ($D(x)$ should be zero), or a real data point ($D(x)$ should be one). Mathematically, training a GAN amounts to solving the following minimax problem:

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

When the discriminator is trained to optimality previous to each update of the generator parameters, Goodfellow et al showed that solving (1) minimizes the Jensen-Shannon divergence between $\mathbb{P}_r$ and $\mathbb{P}_g$, where $\mathbb{P}_g$ is the distribution of $G(z)$ with $z \sim \mathbb{P}_z$ [5]. They suggest to use alternating stochastic gradient descent/ascent to train both models simultaneously, updating alternately the discriminator and the generator. This training strategy is subject to instabilities and convergence issues, such as vanishing gradients and mode collapse, an undesirable phenomenon in which the generator distribution $\mathbb{P}_g$ collapses to a few modes of the target distribution $\mathbb{P}_r$.

In breakthrough papers, Arjovsky et al provide theoretical explanations of convergence issues affecting GAN training algorithms, and propose a variant thereof, Wasserstein GANs (WGANs) [6], [7]. The training objective is modified to cope with discontinuities of the Jensen-Shannon divergence of classical GANs with respect to the parameters of the generator; the Jensen-Shannon divergence is replaced by the 1-Wasserstein distance, that is continuous everywhere and differentiable almost everywhere under weak assumptions on the generator and noise distribution. Since the resulting objective is untractable in its original form, Arjovsky et al rely on Kantorovich-Rubinstein duality to rewrite it in a simpler way, at the cost of imposing a Lipschitz constraint on the discriminator. To emphasize the fact that the discriminator's output does not necessarily lie in the interval $[0, 1]$, unlike in the classical GAN framework, the discriminator is referred to as a *critic*. Training WGANs amounts to compute an optimal critic $f : \mathbb{R}^N \to \mathbb{R}$ and generator $G : \mathbb{R}^n \to \mathbb{R}^N$ for the problem

$$\min_G \max_{\mathcal{L}(f)=1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[f(G(z))], \quad (2)$$

where $\mathcal{L}(f)$ is the Lipschitz constant of f. Arjovsky et al show empirically that WGANs are more robust to mode collapse and less subject to training instabilities than classical GANs [7].

In practice, the critic and generator are again neural networks. The natural question is then: "How can we ensure that the function represented by a neural network has a given Lipschitz constant?" Several possibilities have been proposed in the literature. In [7], Arjovsky et al clip the weights of the model to a compact set (typically, some interval $[-l, l]$, with $l > 0$ sufficiently small). Other works have considered spectral norm regularization [8] and orthogonality of the model parameters [9] to keep the Lipschitz constant of the network

under control. Guljarani et al rely on optimal transport theory results characterizing the norm of the gradient of the optimal critic $\nabla f^*(x)$, minimizing (2) for a fixed generator G and latent space distribution $\mathbb{P}_z$; their proposed algorithm adds a regularizer to (2) that promotes the norm of $\nabla f^*(x)$ to be close to one [10]. Although their approach achieves impressive empirical results, it lacks theoretical support as it does not fully comply with optimal transport theory [11], [12]. Furthermore, Muller et al show experimentally that this strategy might be less robust to mode collapse, the undesirable situation in which some modes of the target distribution are over-represented by the generative model [9] . Let us finally mention the work of Mallasto et al, that proposes to bypass Lipschitz constraints by resorting to c-transformations; their numerical results do not show any improvement compared to existing approaches [13].

Beyond generative models, training a deep network under Lipschitz constraints is common in adversarial learning, as models with a small Lipschitz constant are more robust to adversarial attacks [14]. In that setting, Bungert et al propose using a regularizer that estimates the Lipschitz constant at each iteration [15]; this is strongly related to the one-sided regularizer considered in the appendix of the above-mentioned work of Guljarani et al [10]. Beyond not fully ensuring the Lipschitz property (replacing constraints by a regularizer), estimating the Lipschitz constant comes with additional costs.

Understanding WGANs, and developing efficient WGAN training algorithms, is now a very active area of research [16], [17]. In this work, we focus on the original WGAN approach, where the Lipschitz constraint is ensured through weight clipping. We propose a new clipping strategy that relies on the existence of invariances in the critic loss landscape.

Invariant curves resulting from positive homogeneity of activations have already been exploited in the framework of DNN training [18]–[21] and DNN quantization [22]. Neyshabur et al proposed Path-SGD, a training algorithm leading to a rescaling invariant optimization, coming with additional computational costs that are substantial in the small-batch (or no batch) setting [18]. Almost simultaneously, Badrinarayanan et al recasted training in parameter spaces with symmetries into optimization problems on Riemannian manifolds [19]. More recently, Meng et al improved on Path-SGD by proposing an alternative training method: they represent the set of paths in the network as a low-dimensional linear subspace (subspace generated by linear combination of a set of basis paths), and train the model directly in the subspace [20]. Finally, we mention the work [21], which also contains a training strategy rescaling the input/output weight vectors of neurons, though aiming to balance activations instead of individual weights in our case. To our knowledge, our approach is the first that addresses generative models and constrained training.

Finally, let us mention that the development of new GAN variants is a very timely research topic; we refer the reader to [23] and references therein for an overview of recent GAN architectures.

Contributions: We propose a new clipping strategy that relies on the existence of invariant curves in the critic loss landscape, a consequence of the positive homogeneity of common activations, including ReLU and variants thereof [24]. We then provide experiments on the MNIST dataset, indicating that the proposed strategy leads to a faster training of WGANs with feed-forward architectures.

II. An improved weight clipping algorithm for Wasserstein GANs

Let us consider a two-layer[1] critic $f : \mathbb{R}^N \to \mathbb{R}$:

$$f(x; W_1, W_2) = \sigma(W_2\phi(W_1 x)), \tag{3}$$

with weight matrices $W_1 \in \mathbb{R}^{d\times N}$, $W_2 \in \mathbb{R}^{1\times d}$, d the number of hidden units, $\phi : \mathbb{R} \to \mathbb{R}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ nonlinearities[2]. For simplicity, we write $W := (W_1, W_2) \in \mathcal{E} := \mathbb{R}^{d\times N} \times \mathbb{R}^{1\times d}$, and $f_W(x) := f(x; W_1, W_2)$. We assume that the nonlinearity ϕ satisfies the following assumption (satisfied by, e.g., ReLU-type activations [24]).

Assumption II.1. The activation function ϕ has the positive homogeneity property, namely, for all $v \in \mathbb{R}$ and for all $\alpha \geq 0$, there holds $\phi(\alpha v) = \alpha\phi(v)$.

Given real data $\{x_i\}_{i=1}^m$, $x_i \in \mathbb{R}^N$, latent variable samples $\{z_i\}_{i=1}^m$, with $z_i \in \mathbb{R}^n$, and a generator $G : \mathbb{R}^n \to \mathbb{R}^N$, Arjovsky et al [7] updates the critic parameter W according to the constrained minimization problem

$$\min_{W\in\mathcal{B}} \mathcal{L}(W), \tag{P}$$

with loss $\mathcal{L}(W) := \frac{1}{m}\sum_{i=1}^m (f_W(G(z_i)) - f_W(x_i))$ and domain

$$\mathcal{B} = \{W = (W_1, W_2) : \|W_1\|_\infty \leq l, \|W_2\|_\infty \leq l\}, \tag{4}$$

for some $l > 0$, with matrix norm

$$\|A\|_\infty := \max_{\substack{i=1,\dots,d_1\\ j=1,\dots,d_2}} |A(i,j)|, \qquad \forall A \in \mathbb{R}^{d_1\times d_2}.$$

Problem (P) is typically addressed using projected stochastic gradient descent (P-SGD): iteration k, for $k = 1, 2, \dots$, is made of the two following steps:

$$\begin{cases} W_{\text{unc}}^{k+1} = W^k - \alpha^k \nabla\mathcal{L}^k(W^k), \\ W^{k+1} = \pi_{\mathcal{B}}(W_{\text{unc}}^{k+1}), \end{cases} \tag{P-SGD}$$

where $\mathcal{L}^k$ is the mini-batch approximation of the loss at iteration k, and where $\pi_{\mathcal{B}}(W)$ is the Euclidean projection of W on $\mathcal{B}$. Recalling that $W = (W_1, W_2)$, we consider the norm:

$$\|W\| = \sqrt{\sum_{i,j} W_1(i,j)^2 + \sum_{i,j} W_2(i,j)^2}.$$

The Euclidean projector $\pi_{\mathcal{B}}(W)$ is given by

$$\pi_{\mathcal{B}}(W) = ([W_1]_{-l}^{l}, [W_2]_{-l}^{l}), \tag{5}$$

[1]We first focus on a 2-layer critic for the clarity of the presentation, see Section III for a generalization to L layers.

[2]As usual, we write with a slight abuse of notation, for a vector $v = (v_1, \dots, v_d) \in \mathbb{R}^d$, $\phi(v) = (\phi(v_1), \dots, \phi(v_d)) \in \mathbb{R}^d$.

where $[A]_{-l}^{l}$ is obtained from the matrix A by truncating its entries to the interval $[-l, l]$.

When the critic or generator is non-smooth, P-SGD is replaced by its subgradient counterpart. Projection-based extensions of SGD variants, such as RMSProp and Adam, are also often used in practice.

Let $\mathcal{D}_+ \subset \mathbb{R}^{d\times d}$ be the set of diagonal matrices with strictly positive diagonal elements. For any $D \in \mathcal{D}_+$, we define the mapping:

$$\gamma_D : \mathcal{E} \to \mathcal{E} : (W_1, W_2) \mapsto (D^{-1}W_1, W_2 D). \tag{6}$$

Note that, by Assumption II.1, for all $W = (W_1, W_2) \in \mathcal{E}$ and for all $D \in \mathcal{D}_+$, $f_{\gamma_D(W)}(x) = f_W(x)$ so that $\mathcal{L}(\gamma_D(W)) = \mathcal{L}(W)$. We exploit these invariant curves to speed up and stabilize training. Instead of projecting, at each iteration, the unfeasible iterate onto $\mathcal{B}$, we project on $\mathcal{B}$ the closest equivalent model, according to the invariance described above. At iteration k, our proposed algorithm computes

$$\begin{cases} W_{\text{unc}}^{k+1} = W^k - \alpha^k \nabla \mathcal{L}^k(W^k), \\ W^{k+1} = \pi_{\mathcal{B}}(\gamma_{D^*}(W_{\text{unc}}^{k+1})), \end{cases} \tag{IP-SGD}$$

where D^* is the optimal scaling matrix, minimizing the distance to the box $\mathcal{B}$:

$$D^* = \operatorname{argmin}_{D\in\mathcal{D}_+} \|\gamma_D(W_{\text{unc}}^{k+1}) - \pi_{\mathcal{B}}(\gamma_D(W_{\text{unc}}^{k+1}))\|, \tag{7}$$

where $\|\cdot\|$ is the above-defined norm. An iteration of our proposed algorithm is illustrated on Figure 1 for the case $N = 1$ and $d = 1$, in which case $W_1, W_2 \in \mathbb{R}$ and $\mathcal{D}_+$ is the set of strictly positive real numbers.

Remark II.2. *In the example illustrated on Figure 1, the set of equivalent models $\{\gamma_D(W_{\text{unc}}^{k+1}) : D \in \mathcal{D}_+\}$ (i.e., the grey curve containing W_{unc}^{k+1}) has a non-empty intersection with the feasible domain $\mathcal{B}$, so that $\|\gamma_{D^*}(W_{\text{unc}}^{k+1}) - \pi_{\mathcal{B}}(\gamma_{D^*}(W_{\text{unc}}^{k+1}))\| = 0$. This means that the projection of the equivalent model $\gamma_{D^*}(W_{\text{unc}}^{k+1})$ on the domain $\mathcal{B}$ does not modify the objective: in this specific case, constraints can be imposed "for free" (at least, for this iteration). In general, one cannot expect this to be the case, and projecting on $\mathcal{B}$ the point $\gamma_{D^*}(W_{\text{unc}}^{k+1})$ instead of W_{unc}^{k+1} may even worsen the objective. Indeed, while our choice of the representative $\gamma_{D^*}(W_{\text{unc}}^{k+1})$ ensures proximity to $\mathcal{B}$, proximity of $\mathcal{L}(\gamma_{D^*}(W_{\text{unc}}^{k+1}))$ to $\mathcal{L}(\pi_{\mathcal{B}}(\gamma_{D^*}(W_{\text{unc}}^{k+1})))$ depends on the local variation of the objective. Our proposed methodology is motivated by the geometry of the invariances with respect to the feasible domain $\mathcal{B}$. As illustrated on Figure 1, (at least) in the 2-dimensional case, the loss has slower variations close to the corners of the box $\mathcal{B}$, i.e., in the part of the domain where we expect most of our iterates $\gamma_{D^*}(W_{\text{unc}}^{k})$ to lie and our projections to happen.*

Remark II.3. *While D^* defined in (7) is always unique (by convexity of $\mathcal{B}$), there may exist, depending on the value of W, an infinite number of matrices $D \in \mathcal{D}_+$ such that $\pi_{\mathcal{B}}(\gamma_D(W)) = \pi_{\mathcal{B}}(\gamma_{D^*}(W))$; see Figure 2. As any point on the blue curve yields the same projection on $\mathcal{B}$, hence the same next iterate, we do not necessarily need to solve (7) to a very high accuracy.*

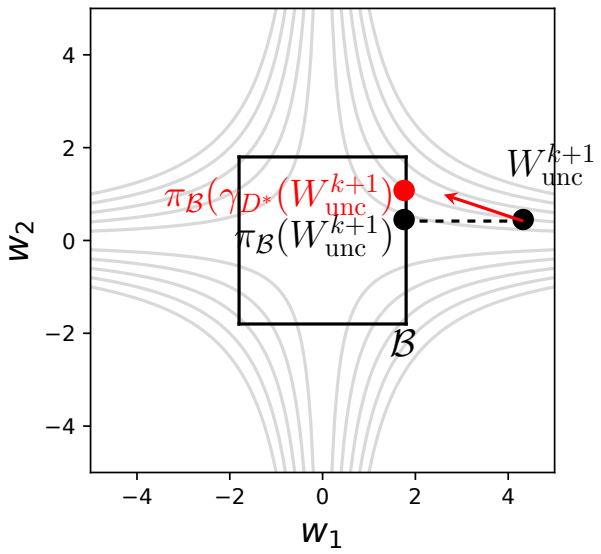

Fig. 1: Illustration of our proposed clipping technique, for $d = 1$ and $N = 1$. Grey curves are equivalent models (i.e., curves of constant loss). Instead of projecting the iterate W_{unc}^{k+1} on $\mathcal{B}$ directly, we first move along the invariant curve to get closer to $\mathcal{B}$, by computing an optimal rescaling matrix D^*, and then project this new model on $\mathcal{B}$. (In this specific example, the projection is the identity map.)

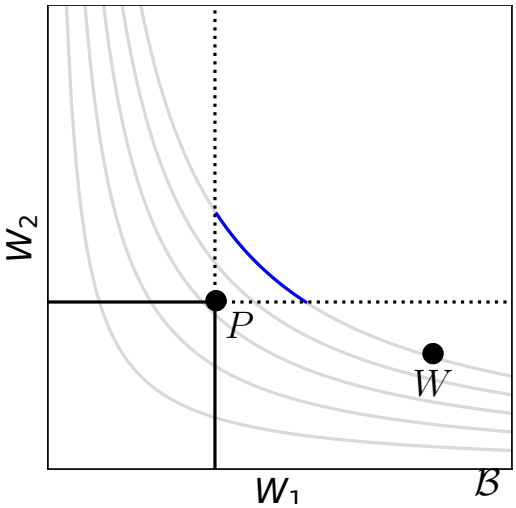

Fig. 2: Existence of an infinite number of matrices $D \in \mathcal{D}_+$ such that $\pi_{\mathcal{B}}(\gamma_D(W)) = P$, in the case $d = 1$ and $N = 1$. The range of values of D associated to the blue curve lead to the same point after projection on the feasible set $\mathcal{B}$.

A. Solving the projection problem

Let us now discuss the computation of the optimal rescaling matrix D^* defined in (7). Combining (5) and (6), the optimiza-

tion problem (7) may be written as follows:

$$\min_{D\in\mathcal{D}_+} \sum_{i,j}[D^{-1}(i,i)|W_1(i,j)|-l]_+^2+\sum_{i,j}[|W_2(i,j)|D(j,j)-l]_+^2. \tag{8}$$

We estimate D^* iteratively, as described in Algorithm 1. The diagonal elements of the rescaling matrix D are progressively increased/decreased to balance the norms of the constraints associated with each row of W_1 and column of W_2.

Algorithm 1 Estimation of the rescaling matrix D^*

Let $\alpha > 1$ some initial rescaling factor, $D = \alpha \mathbf{I}_d$, and $k_{\max} > 0$.
% Define initial rescaling factor for each unit, based on constraints violation of weights entering/leaving the unit.
for $i = 1, \dots, d$ **do**
  Compute $n_{in}(i) := \sum_j [|W_1(i,j)| - l]_+^2$
  Compute $n_{out}(i) := \sum_j [|W_2(j,i)| - l]_+^2$
  if $n_{in}(i) < n_{out}(i)$ **then**
    % we should increase the weights entering the ith unit; by default we will decrease them by a factor $D(i,i)$.
    $D(i,i) := 1/D(i,i)$
  end if
end for
% Rescale the weights entering/leaving each unit, until constraints violations becomes comparable.
for $i = 1, \dots, d$ **do**
  Let $k = 0$
  while $k \leq k_{\max}$ **do**
    Compute $n_{in}^{new}(i) := \sum_j [D^{-1}(i,i)|W_1(i,j)| - l]_+^2$
    Compute $n_{out}^{new}(i) := \sum_j [D(i,i)|W_2(j,i)| - l]_+^2$
    if $n_{in}(i) > n_{out}(i)$ and $n_{in}^{new}(i) > n_{out}^{new}(i)$ **then**
      $D(i,i) = \alpha D(i,i)$
    else if $n_{out}(i) > n_{in}(i)$ and $n_{out}^{new}(i) > n_{in}^{new}(i)$ **then**
      $D(i,i) = D(i,i)/\alpha$
    else
      $k = k_{\max}$
    end if
    $k := k + 1$
  end while
end for

The whole training procedure is described in Algorithm 2; note that, for generality, we have replaced the SGD step in (IP-SGD) by an iteration of an arbitrary unconstrained optimization solver; typically, SGD, Adam or RMSProp.

III. Generalization to L layers and DCGANs

For critic networks made of more than two layers, Line 5 of Algorithm 2 is repeated to compute an optimal rescaling matrix D for each layer. Similarly, our algorithm can be extended to linear layers with biases; simply apply the rescaling operation to the biases entering and leaving each hidden unit.

We can also generalize our proposed algorithm to architectures involving convolutional layers. Let us consider two successive convolutional layers in the network architecture, with filters $W_{in} \in \mathbb{R}^{C_2\times C_1\times k_{in}\times k_{in}}$ and $W_{out} \in \mathbb{R}^{C_3\times C_2\times k_{out}\times k_{out}}$, where C_2 and C_3 are the number of output channels of W_{in} and W_{out}, respectively, C_1 and C_2 their numbers of input channels, and k_{in} and k_{out} their spatial filter dimension. We assume that the two convolutional layers are separated by a nonlinearity satisfying Assumption II.1, but no batch normalization. Let us write $W_{in}[i,:,:,:] \in \mathbb{R}^{1\times C1\times k_{in}\times k_{in}}$ the slice of the tensor W_{in} corresponding to the filters entering the ith hidden unit, and $W_{out}[:,i,:,:] \in \mathbb{R}^{C_3\times 1\times k_{out}\times k_{out}}$ the slice of W_{out} containing filters leaving the ith hidden unit. Similarly as for feed-forward neural networks, positive homogeneity of the activation ensures that we can multiply $W_{in}[i,:,:,:]$ and divide $W_{out}[:,i,:,:]$ by the same factor without changing the model. We can thus naturally apply Algorithm 1 and Algorithm 2 to convolutional layers, by simply rescaling slices of convolutional tensors instead of rows/columns of weight matrices.

Algorithm 2 Improved weight clipping for WGANs

1: Let $\{\alpha^k\}$ be a sequence of stepsizes. Set $k = 0$, and initialize $W^0 \in \mathcal{E}$.
2: **while** not converged **do**
3: % In the next line, we compute a (not necessarily feasible) iterate, using any optimization algorithm (e.g., SGD, Adam, RMSProp)
4: Let $W_{\text{unc}}^{k+1} := \text{optim}(W^k; \alpha^k)$
5: Compute $\hat{D}$, an estimator of D^* defined in (7), according to Algorithm 1.
6: Let $W^{k+1} = \pi_{\mathcal{B}}(\gamma_{\hat{D}}(W_{\text{unc}}^{k+1}))$
7: $k := k + 1$
8: **end while**

IV. Numerical results

We illustrate our algorithms on image generation trained over the MNIST dataset [25]. We consider two types of WGANs architectures a simple MLP-WGAN and a DCGAN (with no batch normalization in the critic).

a) Experiment with MLP architectures: In this first experiment, the critic and generator are both feed-forward neural networks, with architecture given in Table I. The biases are fixed to zero. The latent variables follow a 100-dimensional standard Gaussian distribution, while the image space is 32×32 (we used 2-pixel padding). Note that the usual heuristic that selects the same clipping value for each parameter [6] may harm signal propagation through the layers; the same issue has been noted for model initialization, where distributions scaling with the number of input activations to a layer are often preferred, as in the He initialization [26]. To ensure signal propagation while keeping the Lipschitz constant under control, we therefore choose different clipping values for each weight matrix of the critic. Let us write d_l the dimension of the lth hidden unit (with $d_0 = N = 32 \times 32$ and $d_L = 1$ the input and output dimensions of the network). By analogy with the He initialization, the clipping value for $W_l \in \mathbb{R}^{d_{l+1}\times d_l}$ was selected around $1/\sqrt{d_l}$; we used for the critic architecture described in Table I the clipping values 0.035, 0.045, 0.0625

for the weight matrices $W_1 \in \mathbb{R}^{512\times1024}$, $W_2 \in \mathbb{R}^{256\times512}$ and $W_3 \in \mathbb{R}^{1\times256}$. As suggested in the literature, we let the optimizer update more regularly the critic than the generator, to improve the correspondance between the generator loss and the Wasserstein distance between the real and generated distributions. During the 100 first epochs, the generator was updated once per 100 critic updates. For the rest of the epochs, it was updated once every 5 critic updates.

We trained the WGAN model, either using traditional weight clipping for the critic, or using Algorithm 2, in both cases using the RMSProp optimization algorithm with learning rate $2 \cdot 10^{-4}$, all other parameters being chosen as default. In Algorithm 1, we use $\alpha = 1.1$ and $k_{\max} = 5$. We run the experiment with a batch-size of 128 on a workstation configured with Intel(R) Xeon(R) Silver 4108 CPU and GeForce RTX 2080 Ti graphical card, for 2000 epochs, and recorded both the generator loss, the generated pictures, and the running time at each epoch.

TABLE I: Architectures for the critic and generator.

Critic	**Generator**
Linear(32×32, 512)	Linear (latent dim., 128)
Leaky ReLU (param. 0.2)	ReLU
Linear(512, 256)	Linear (128, 256)
Leaky ReLU (param. 0.2)	Batchnorm (param. 0.8)
Linear(256, 1)	ReLU
	Linear (256, 512)
	Batchnorm (param. 0.8)
	ReLU
	Linear (512, 1024)
	Batchnorm (param. 0.8)
	ReLU
	Linear (1024, 28×28)
	Tanh

Figure 3 illustrates the images generated by the model at comparable training times, for Algorithm 2 (top image) and for projected RMSProp (bottom image), where the invariances of the model are not used. When using our algorithm, the model is already providing recognizable digits, while the digits generated by the standard approach are still way too blurry to be recognized. As the cost per iteration of Algorithm 2 is slightly higher than the baseline approach, we compare the pictures at different epochs corresponding to similar training times on our workstation (respectively, 4550 sec. for our proposed method and 4603 sec for the baseline).

b) Experiment using a DCGAN architecture: We apply here our proposed algorithm to WGAN training using convolutional neural networks as critic and generator. We rely on a DCGAN architecture [27], discarding batch normalization layers in the critic. The architecture is given in Table II. Similarly as in our previous experiment, the entries of a convolutional filter tensor $W \in \mathbb{R}^{C_{out}\times C_{in}\times k\times k}$ follow a uniform distribution supported on an interval $[-l, l]$, with l proportional to $1/\sqrt{C_{in} \times k^2}$ (similar to the He initialization [26]). We select the following values for l, in each layer: $l = 0.05, 0.008, 0.006, 0.004$. Accordingly, the clipping values of the convolutional weight filters were set to

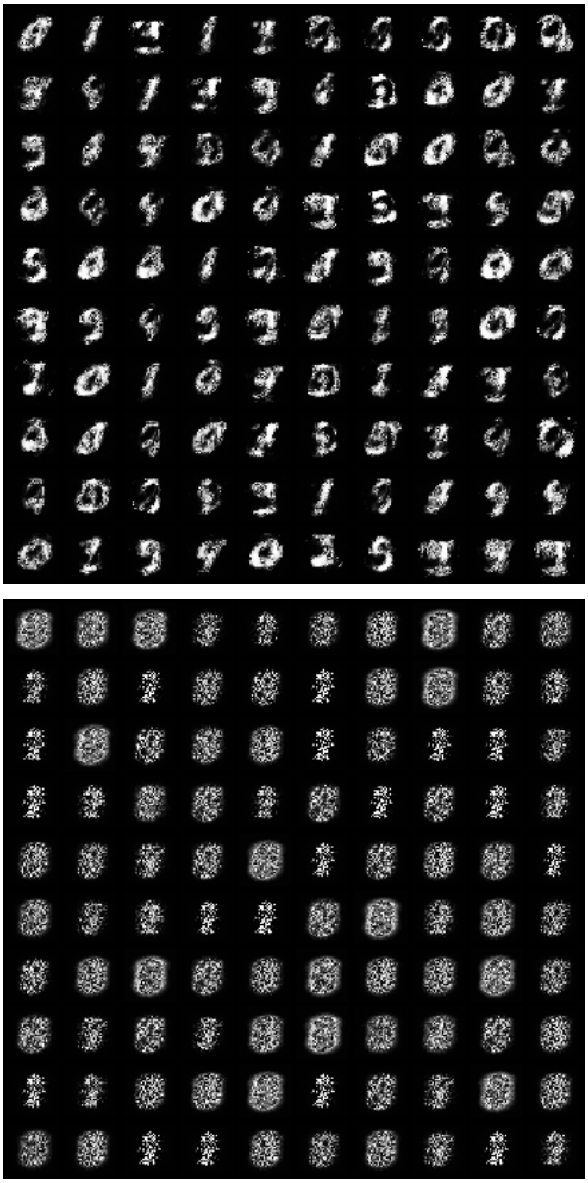

Fig. 3: Comparison of images generated by the model (for feed-forward architectures) when learned using Algorithm 2 (top) compared to default training, in which the iterates of RMSProp are projected on $\mathcal{B}$ without any use of model invariances (bottom); epochs number 210 (top) and 230 (bottom), corresponding to similar training time on our workstation: 4550 sec. (top) vs 4603 sec. (bottom)

$l = 0.05, 0.008, 0.006, 0.004$. The latent variables follow a 100-dimensional standard Gaussian distribution. The critic and generator are trained using RMSProp with learning rate $2{\cdot}10^{-4}$, and all other parameters set as default. In Algorithm 1, we use $\alpha = 1.1$ and $k_{\max} = 5$. Similarly as for our MLP experiment, we update the critic more often that the generator: for the 100 first epochs we update the critic 100 times per generator update, starting from epoch 101 the critic is updated 5 times per generator update.

TABLE II: Architecture of our DCGAN.

Critic	Generator
Conv2d(1, 64, 4, 2, 1)	ConvTranspose2d(100, 512, 4, 1, 0)
Leaky ReLU (param. 0.2)	BatchNorm2d(512)
Conv2d(64, 128, 4, 2, 1)	ReLU
Leaky ReLU (param. 0.2)	ConvTranspose2d(512, 256, 4, 2, 1)
Conv2d(128, 256, 4, 2, 1)	BatchNorm2d(256)
Leaky ReLU (param. 0.2)	ReLU
Conv2d(256, 1, 4, 1, 0)	ConvTranspose2d(256, 128, 4, 2, 1)
	BatchNorm2d(128)
	ReLU
	ConvTranspose2d(128, 64, 4, 2, 1)
	Tanh

We illustrate in Figure 4 the images generated by the discriminator *at the same epoch*, when the critic is trained using Algorithm 2, or using the standard approach, projecting iterates on $\mathcal{B}$ at each iteration without using invariances of the loss. Though, per epoch, our proposed algorithm yields more accurate results than the baseline, this comes with higher training costs. We argue that our proposed methodology is also promising for GANs with convolutional layers (faster training per epoch), but is balanced by a larger training time. To alleviate the training cost, one may, for example, explore intermediate approaches projecting the iterates (either using Algorithm 2 or standard orthogonal projection) every k iterations (with $k > 1$), allowing for unfeasible iterates in-between.

V. Conclusions

We have proposed a new weight clipping strategy that exploits invariant curves in the critic loss landscape, whose existence is a consequence of the positive homogeneity of common activations. In the case of a feed-forward critic and generator, we have illustrated numerically that our proposed method outperforms the standard approach (where the iterate is simply projected on the feasible domain at each iteration): for a similar training time, images generated by the model trained with our proposed clipping strategy are substantially more accurate that those generated by the baseline. Finally, we have also extended our proposed framework to convolutional layer architectures. Future research will address extensions of our proposed clipping strategy to batch normalization layers.

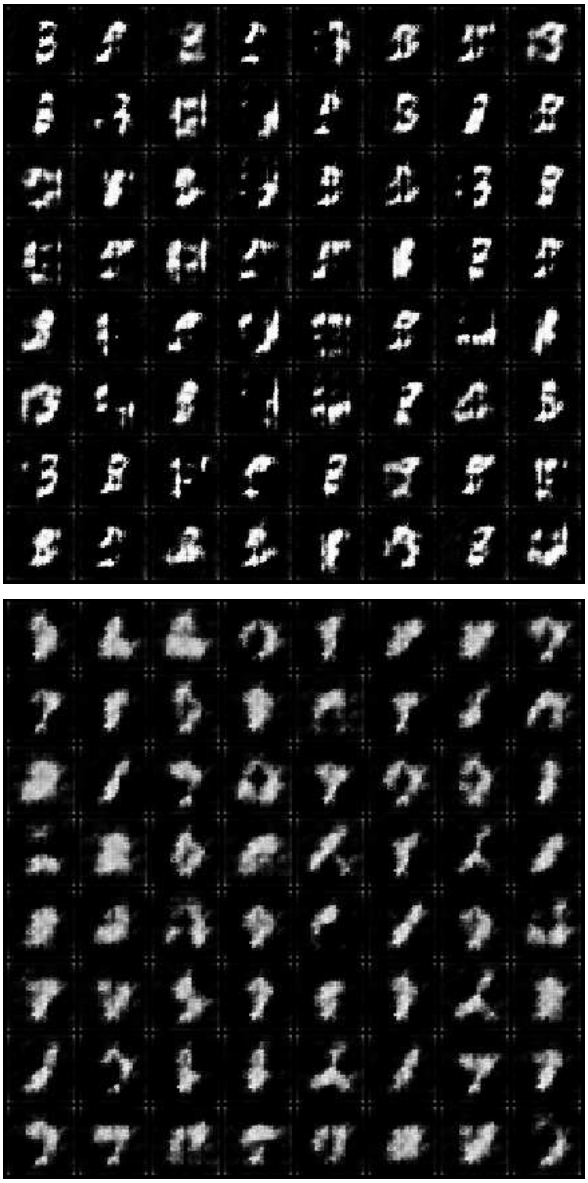

Fig. 4: Comparison of images generated using a DCGAN-type architecture (no batch normalization in the critic) when learned using Algorithm 2 (top) compared to default training, in which the iterates of RMSProp are projected on $\mathcal{B}$ without any use of model invariances (bottom); epoch number 102, training times: 4876 sec. (top) vs 4272 sec. (bottom)

Acknowledgment

Most of this work was done when the author was with the University of Oxford and the National Physical Laboratory (Teddington, UK); the author was then funded by the National Physical Laboratory. The author is now funded by the FNRS, Belgium.

References

[1] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks." in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019.

[2] D. Acharya, Z. Huang, D. Paudel, and L. V. Gool, "Towards high resolution video generation with progressive growing of sliced Wasserstein GANs," Master Thesis, ETH Zürich, available at: arXiv.1810.02419, 2018.

[3] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," in *Constructive Machine Learning Workshop, 30th Conference on Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, 2016.

[4] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *31st AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, 2017.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in

28th Conference on Neural Information Processing Systems (NeurIPS), Montréal, Canada, 2014.

[6] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *34th International Conference on Machine Learning (ICML)*, Sidney, Australia, 2017.

[8] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *6th International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, 2018.

[9] J. Müller, R. Klein, and M. Weinmann, "Orthogonal Wasserstein GANs," arxiv: 1911.13060, 2019.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *31st Conference on Neural Information Processing Systems (Neurips)*, Long Beach, CA, USA, 2017.

[11] M. Gemici, Z. Akata, and M. Welling, "Primal-Dual Wasserstein GAN," arxiv preprint: 1805.09575, 2018.

[12] H. Petzka, A. Fischer, and D. Lukovnikov, "On the regularization of Wasserstein GANs," in *6th International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, 2018.

[13] A. Mallasto, G. Montúfar, and A. Gerolin, "How well do WGANs estimate the Wasserstein metric?" arxiv preprint: 1910.03875, 2019.

[14] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural network," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018.

[15] L. Bungert, R. Raab, T. Roith, L. Schwinn, and D. Tenbrinck, "CLIP: Cheap lipschitz training of neural networks," in *Lecture Notes in Computer Science*, ser. Scale Space and Variational Methods in Computer Vision (SSVM). Springer International Publishing, 2021, pp. 307–319.

[16] G. Biau, M. Sangnier, and U. Tanielian, "Some Theoretical Insights into Wasserstein GANs," *Journal of Machine Learning Research*, vol. 22, pp. 1–45, 2021.

[17] J. Stanczuk, C. Etmann, L. M. Kreusser, and C.-B. Schonlieb, "Wasserstein GANs Work Because They Fail (to Approximate the Wasserstein Distance)," arxiv preprint: 2103.01678, 2021.

[18] B. Neyshabur, R. R. Salakhutdinov, and N. Srebro, "Path-SGD: Path-Normalized Optimization in Deep Neural Networks," in *29th Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2015.

[19] V. Badrinarayanan, B. Mishra, and R. Cipolla, "Symmetry-invariant optimization in deep networks," arxiv preprint: 1511.01754, 2015.

[20] Q. Meng, S. Zheng, H. Zhang, W. Chen, Q. Ye, Z.-M. Ma, N. Yu, and T.-Y. Liu, "G-SGD: optimizing ReLU neural networks in its positive scale-invariant space," in *7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[21] Q. Yuan and N. Xiao, "Scaling-based weight normalization for deep neural networks," *IEEE Access*, vol. 7, pp. 7286–7295, 2019.

[22] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, 2019.

[23] H. D. Meulemeester, J. Schreurs, M. Fanuel, B. D. Moor, and J. A. K. Suykens, "The Bures metric for generative adversarial networks," in *Machine Learning and Knowledge Discovery in Databases. Research Track*. Springer International Publishing, 2021, pp. 52–66.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *27th International Conference on Machine Learning (ICML)*, Haifa, Israël, 2010.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[27] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.