April 6, 2021

Training GANs by Solving ODEs

https://arxiv.org/abs/2010.15040

Vikram Voleti

PhD student, Mila, University of Montreal

Supervisor: **Prof. Christopher Pal**

voletiv.github.io

@ (virtual) Mila, Montreal, Canada

Contributions

- We frame GAN training as solving ODEs.
- We design a regulariser on the gradients to improve numerical integration of the ODE.
- We show that higher-order ODE solvers lead to better convergence for GANs.
- Our algorithm (ODE-GAN) can train GANs to competitive levels without any adaptive optimiser (e.g. Adam [17]) and explicit functional constraints (Spectral Normalisation)

https://arxiv.org/abs/2010.15040

ODE-GAN

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{x \sim p(x)}[\log(D(x; \theta))] + \mathbb{E}_{x \sim p(x)}[\log(1 - D(G(z; \phi); \theta))]$$

$$\ell(\theta, \phi) = [\ell_D(\theta, \phi), \ell_G(\theta, \phi)] \quad (= [-\mathcal{J}(\theta, \phi), \mathcal{J}(\theta, \phi)])$$

$$\theta_{k+1} = \theta_k - \alpha \frac{\partial \ell_D}{\partial \theta}(\theta_k, \phi_k)\Delta t$$

$$\phi_{k+1} = \phi_k - \beta \frac{\partial \ell_G}{\partial \phi}(\theta_k, \phi_k)\Delta t$$

https://arxiv.org/abs/2010.15040

ODE-GAN

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{x \sim p(x)}[\log(D(x; \theta))] + \mathbb{E}_{x \sim p(x)}[\log(1 - D(G(z; \phi); \theta))]$$

$$\ell(\theta, \phi) = [\ell_D(\theta, \phi), \ell_G(\theta, \phi)] \quad (= [-\mathcal{J}(\theta, \phi), \mathcal{J}(\theta, \phi)])$$

$$\theta_{k+1} = \theta_k - \alpha \frac{\partial \ell_D}{\partial \theta}(\theta_k, \phi_k)\Delta t$$

$$\phi_{k+1} = \phi_k - \beta \frac{\partial \ell_G}{\partial \phi}(\theta_k, \phi_k)\Delta t$$

Infinitesimal gradient descent

$$\left[\frac{\partial \theta}{\partial t}, \frac{\partial \phi}{\partial t}\right] = -\left[\alpha \frac{\partial \ell_D}{\partial t}, \beta \frac{\partial \ell_G}{\partial t}\right]$$

https://arxiv.org/abs/2010.15040

Assuming we track the dynamical system exactly – and the **gradient vector field is bounded** – then in the vicinity of a differential Nash equilibrium, the parameters converge to it at a rate independent of the frequency of rotation with respect to the vector field.

$$\left[\frac{\partial \theta}{\partial t}, \frac{\partial \phi}{\partial t}\right] = -\left[\alpha \frac{\partial \ell_D}{\partial t}, \beta \frac{\partial \ell_G}{\partial t}\right]$$

https://arxiv.org/abs/2010.15040

ODE-GAN

$$\begin{pmatrix}\theta_{k+1}\\ \phi_{k+1}\end{pmatrix} = \texttt{ODEStep}(\theta_k, \phi_k, \mathbf{v}, h)$$

Regulariser: $$R(\theta) = \lambda \left\| \frac{\partial \ell_G}{\partial \phi} \right\|^2$$

Algorithm 1 Training an ODE-GAN

Require: Initial states (θ, ϕ), step size h, velocity function $\mathbf{v}(\theta, \phi) = -[\frac{\partial \ell_D}{\partial \theta}, \frac{\partial \ell_G}{\partial \phi}]$, regularization multiplier λ, and initial step counter $i = 0$, maximum iteration I

if $i < I$ **then**

$g_\theta \leftarrow \nabla_\theta \parallel \frac{\partial \ell_G}{\partial \phi}|_{(\theta,\phi)} \parallel^2$

$\theta, \phi \leftarrow \texttt{ODEStep}(\theta, \phi, \mathbf{v}, h)$

$\theta \leftarrow \theta - h\lambda g_\theta$

$i \leftarrow i + 1$

end if

return (θ, ϕ)

https://arxiv.org/abs/2010.15040

ODE-GAN

5.2.1 Different Orders of ODE Solvers

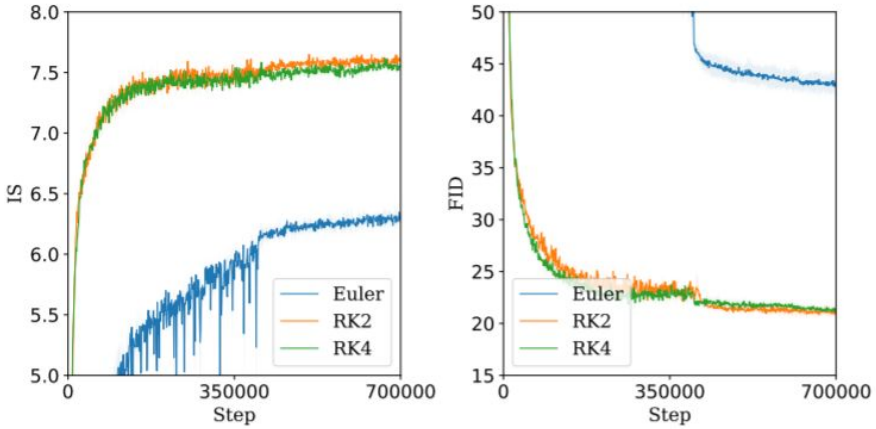

Figure 3: Comparison between different orders of integrators using $\lambda = 0.002$ and $h = 0.02$.

- Training improves with increase in order of integration
- Increasing order further gives diminishing returns
- Higher order methods allow for much larger step sizes:
  - Euler's becomes unstable with h ≥ 0.04, while Heun's method (RK2) and RK4 do not
- Increasing regularization weight reduces the performance gap between Euler and RK2

https://arxiv.org/abs/2010.15040

ODE-GAN

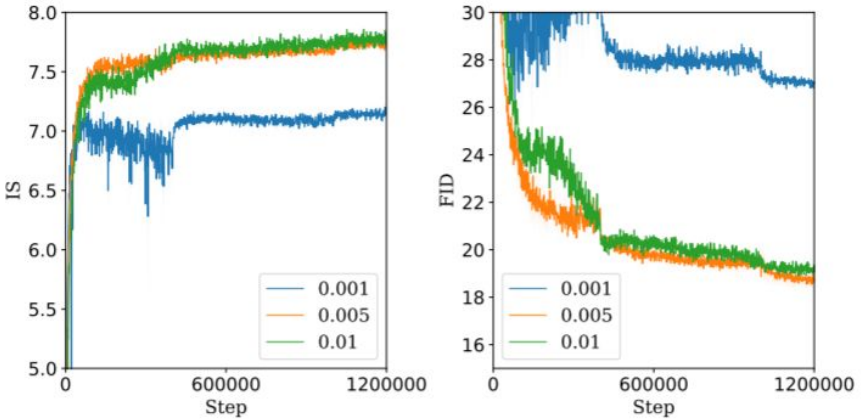

Figure 4: Comparison of runs with different regularisation weight λ shown in legend. The step size used is $h = 0.04$, all with RK4 integrator.

- The regulariser controls the truncation error by penalising large gradient magnitudes
- larger λ leads to smaller error

https://arxiv.org/abs/2010.15040

ODE-GAN

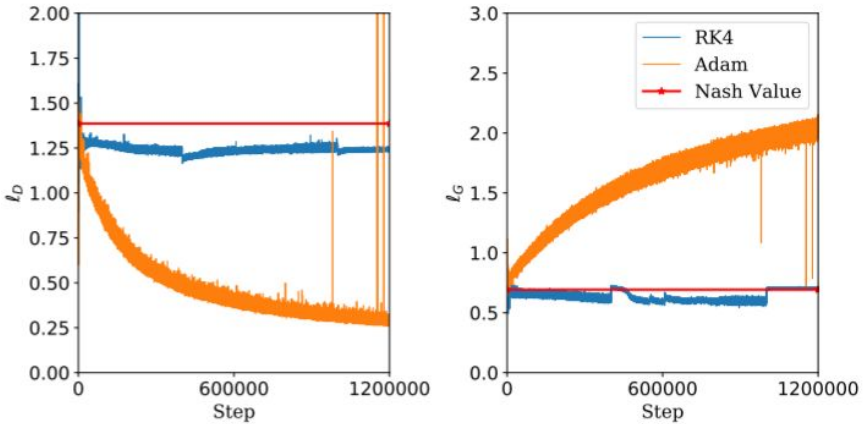

Figure 5: The evolution of loss values for discriminator (left) and generator (right) for ODE-GAN (RK4) versus training with Adam optimisation.

- Discriminator Loss and the Generator Loss stay very close to the values of a Nash equilibrium
- In contrast, the discriminator dominates the game when using the Adam optimiser : evidenced by a continuously decreasing D_loss, while the G_loss increases
    - This imbalance correlates with the well-known phenomenon of worsening FID and IS in late stages of training

https://arxiv.org/abs/2010.15040

ODE-GAN

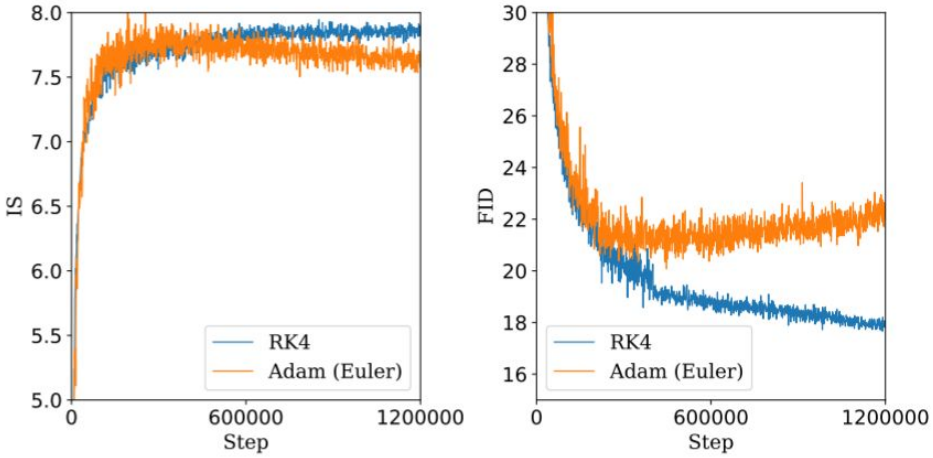

Figure 7: Here we compare using the convergence of RK4 to using Adam, $\lambda = 0.01$ for both.

- Our results challenge the widely-held view that adaptive optimisers are necessary for training GANs
- the often observed degrading performance towards the end of training disappears with improved integration
- this is the first time that competitive results for GAN training have been demonstrated for image generation without adaptive optimisers.

https://arxiv.org/abs/2010.15040

ODE-GAN

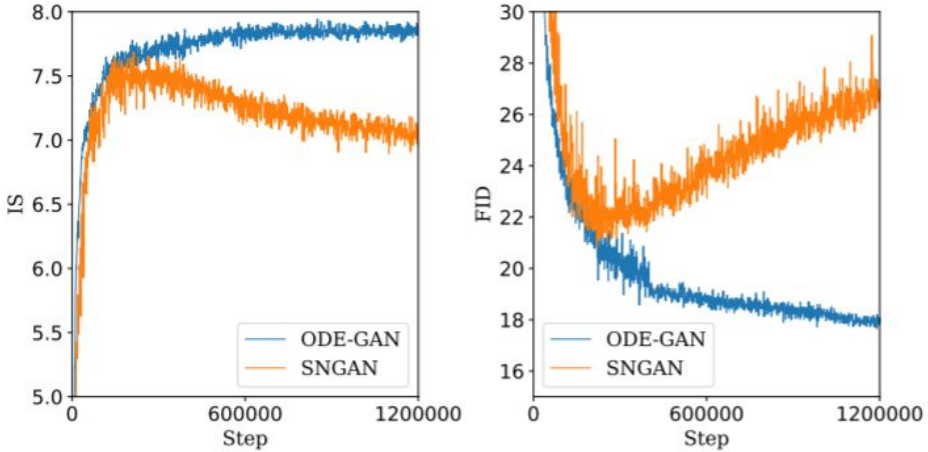

Figure 6: Here we compare ODE-GAN (with RK4 as `ODEStep` and $\lambda = 0.01$) to SN-GAN.

- ODE-GAN can improve significantly upon SN-GAN for both IS and FID

https://arxiv.org/abs/2010.15040

ODE-GAN

Table 1: Numbers taken from the literature are cited. ‡ denotes reproduction in our code. "Best" and "final" indicate the best scores and scores at the end of 1 million update steps respectively. The means and standard deviations (shown by ±) are computed from 3 runs with different random seeds. We use **bold face** for the best scores across each category incl. those within one standard deviation.

Method	FID (best) / FID(final)	IS (best) / IS(final)
CIFAR-10 Unconditional		
– DCGAN –		
ODE-GAN(RK4)	**17.66 ± 0.38 / 18.05 ± 0.53**	**7.97 ± 0.03 / 7.86 ± 0.09**
ODE-GAN(RK4+Adam)	**17.47 ± 0.30** / 23.20 ± 0.95	**8.00 ± 0.06** / 7.59 ± 0.14
SN-GAN‡	21.71 ± 0.61 / 26.16 ± 0.27	7.60 ± 0.06 / 7.02 ± 0.02
– ResNet –		
ODE-GAN (RK4)	**11.85 ± 0.21 / 12.50 ± 0.30**	**8.61 ± 0.06 / 8.51 ± 0.01**
ODE-GAN (RK4 + Adam)	12.11 ± 0.28 / 20.32 ± 1.17	8.23 ± 0.04 / 7.92 ± 0.03
SN-GAN‡	15.97 ± 0.22 / 23.98 ± 2.08	7.71 ± 0.05 / 7.20 ± 0.22
WGAN-ALP (ResNet) [30]	12.96 ± 0.35 / —	8.56 / —
– Evaluated with 5k samples –		
SN-GAN (DCGAN) [23]	29.3 / —	7.42 ± 0.08 / —
WGAN-GP (DCGAN) [14]	40.2 / —	6.68 ± 0.06 / —
SN-GAN (ResNet) [23]	21.7 ± 0.21 / —	8.22 ± 0.05 / —
ImageNet 128 × 128 **Conditional**		
– ResNet –		
ODE-GAN (RK4)	**26.16 ± 0.75 / 28.42 ± 1.46**	**38.71 ± 0.82 / 36.54 ± 1.53**
SN-GAN‡	37.05 ± 0.26 / 41.07 ± 0.46	31.52 ± 0.25 / 29.16 ± 0.20

https://arxiv.org/abs/2010.15040

H.1 Conditional ImageNet Generation

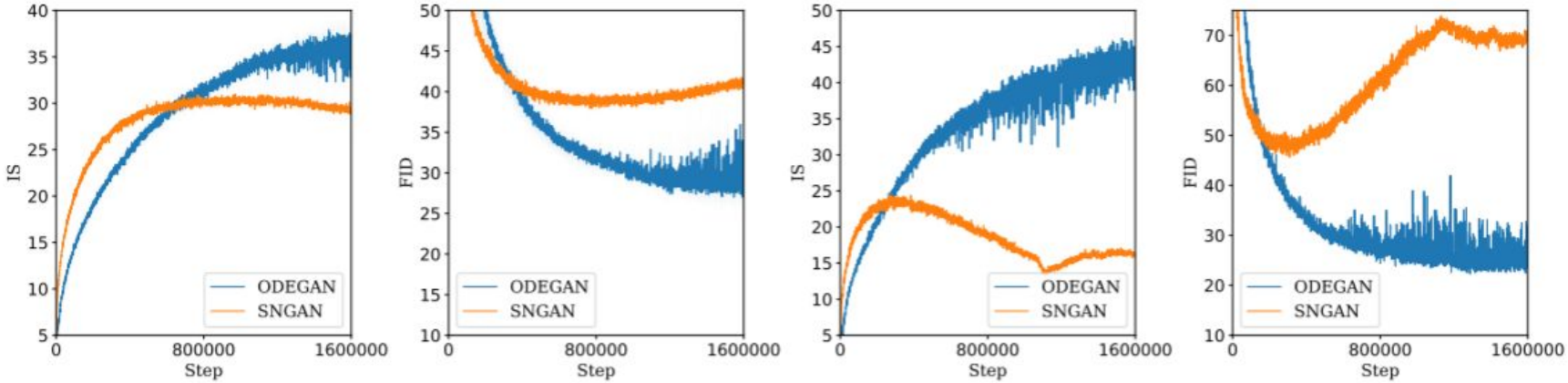

Figure 8: Comparison of IS and FID for ODE-GAN (RK4) vs. SNGAN trained on ImageNet 128×128. In the plot on the left we used a ResNet architecture similar to Miyato et al. [23] for ImageNet; on the right we trained with ResNet (large).

https://arxiv.org/abs/2010.15040

Our empirical results support the hypothesis that, at least locally,
the GAN game is not inherently unstable.

Rather, the **discretisation of GANs' continuous dynamics**,
yielding inaccurate time integration, **causes instability**

Adam and Spectral Norm may harm convergence, they are not necessary when higher-order ODE solvers are available.

https://arxiv.org/abs/2010.15040

ODE-GAN

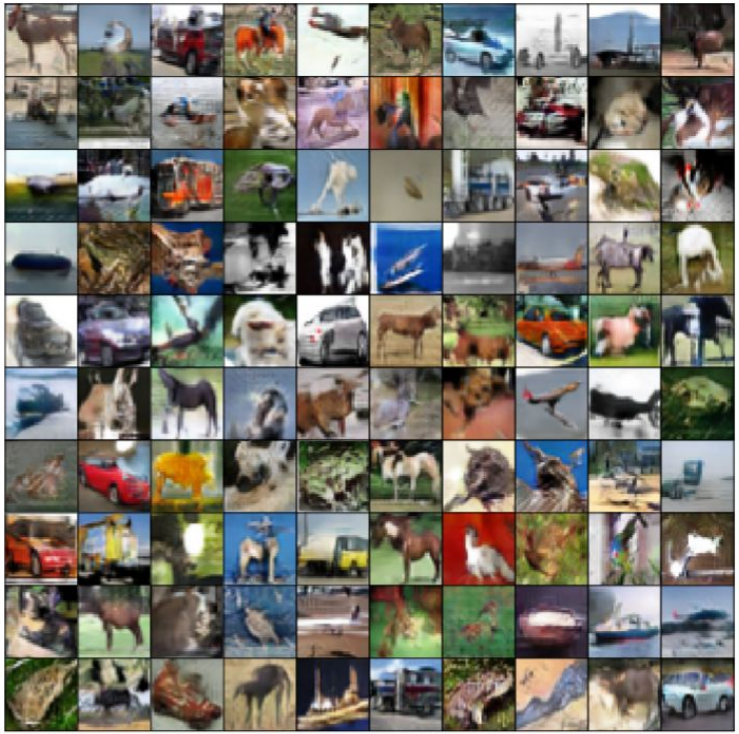

Figure 12: Architecture used DCGAN (CIFAR-10). Sample of images generated using ODE-GAN with $h = 0.04$, RK4 as the integrator and $\lambda = 0.01$.

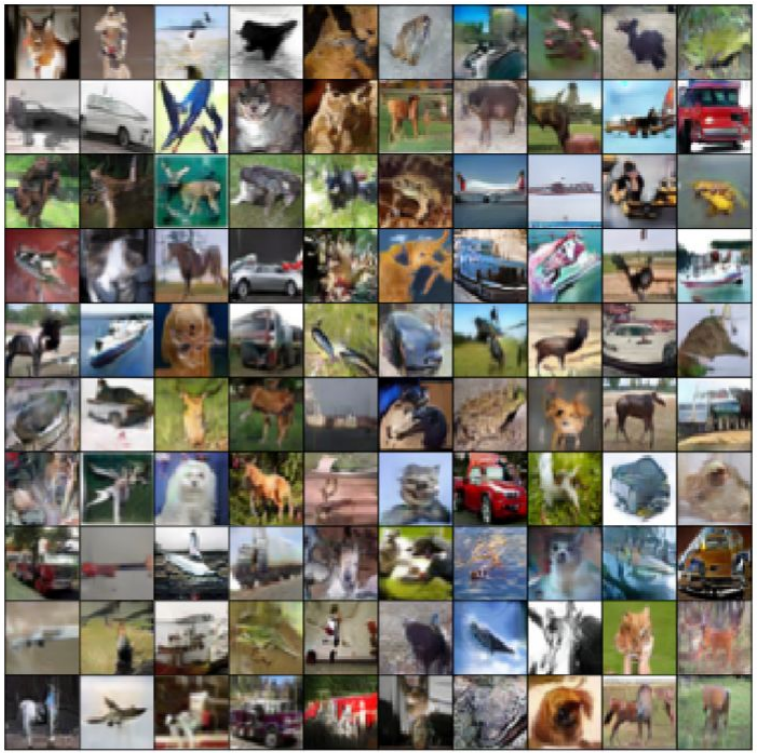

Figure 13: Architecture used ResNet (CIFAR-10). Sample of images generated using ODE-GAN (RK4) with $h = 0.01$, RK4 as the integrator and $\lambda = 0.01$.

https://arxiv.org/abs/2010.15040

ODE-GAN

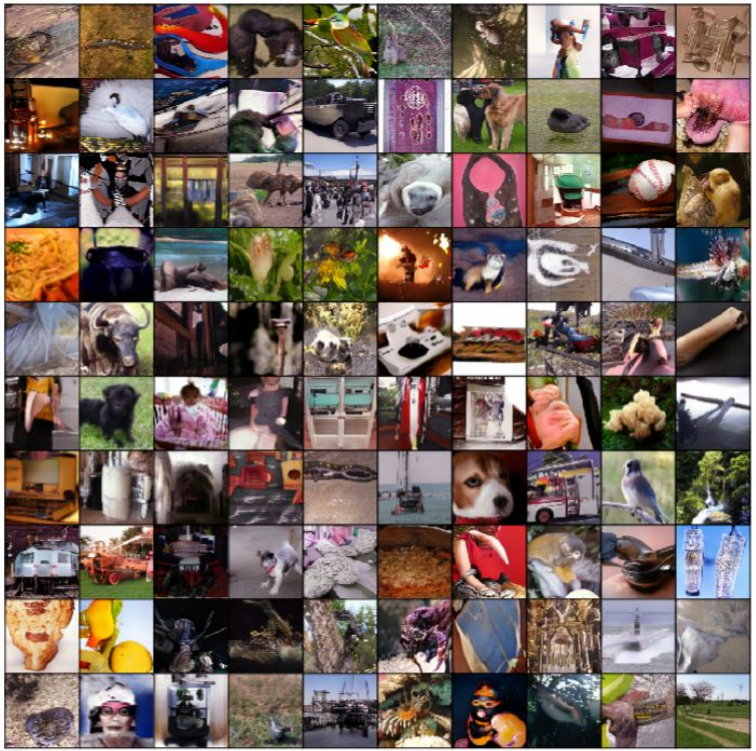

Figure 14: Architecture used ResNet (ImageNet conditional). Sample of images generated using ODE-GAN (RK4) with $h = 0.02$, RK4 as the integrator and $\lambda = 0.00002$.

https://arxiv.org/abs/2010.15040

Fine print

- Up to 2x slower than original GAN training
- our algorithm seems to be more prone to landing in NaNs during training for conditional models...

https://arxiv.org/abs/2010.15040

Thank you!

voletiv.github.io