FIGR : Few-shot Image Generation with Reptile

Sanghyuck Na

Sep, 27, 2019

Dongguk University

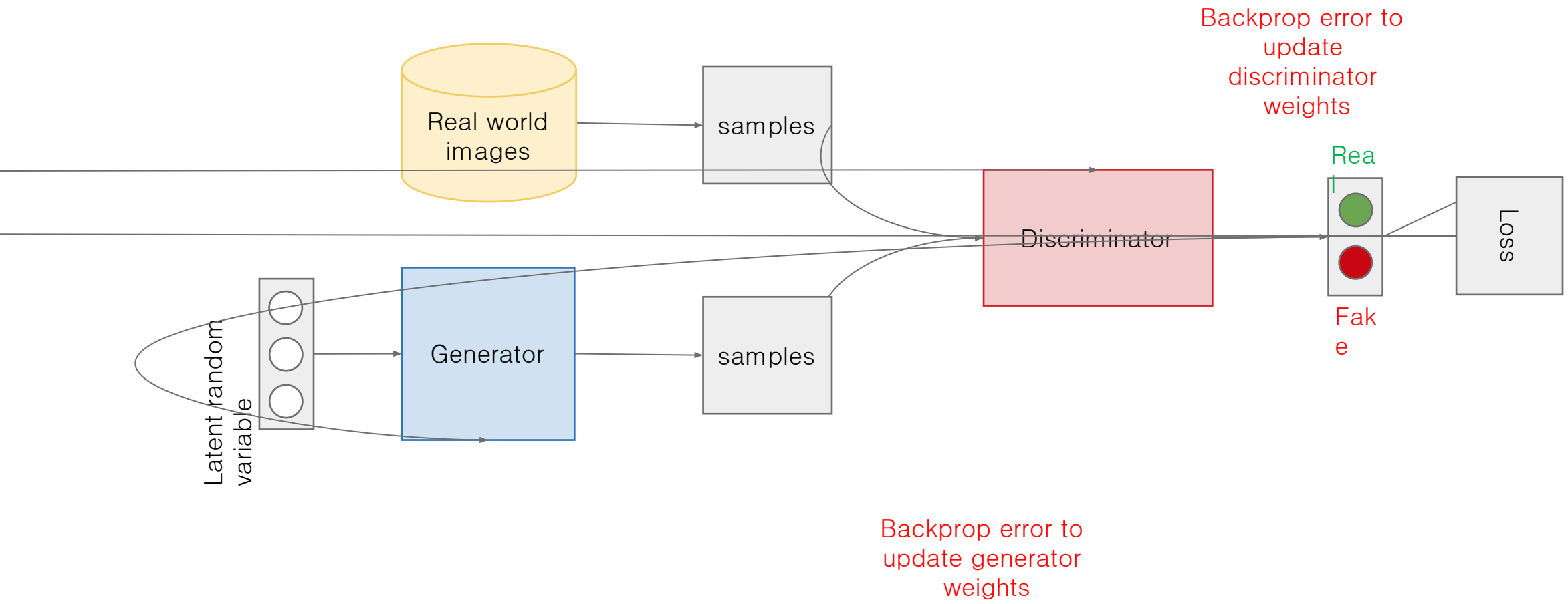
Artificial Intelligence Laboratory

shna@Dongguk.edu

0

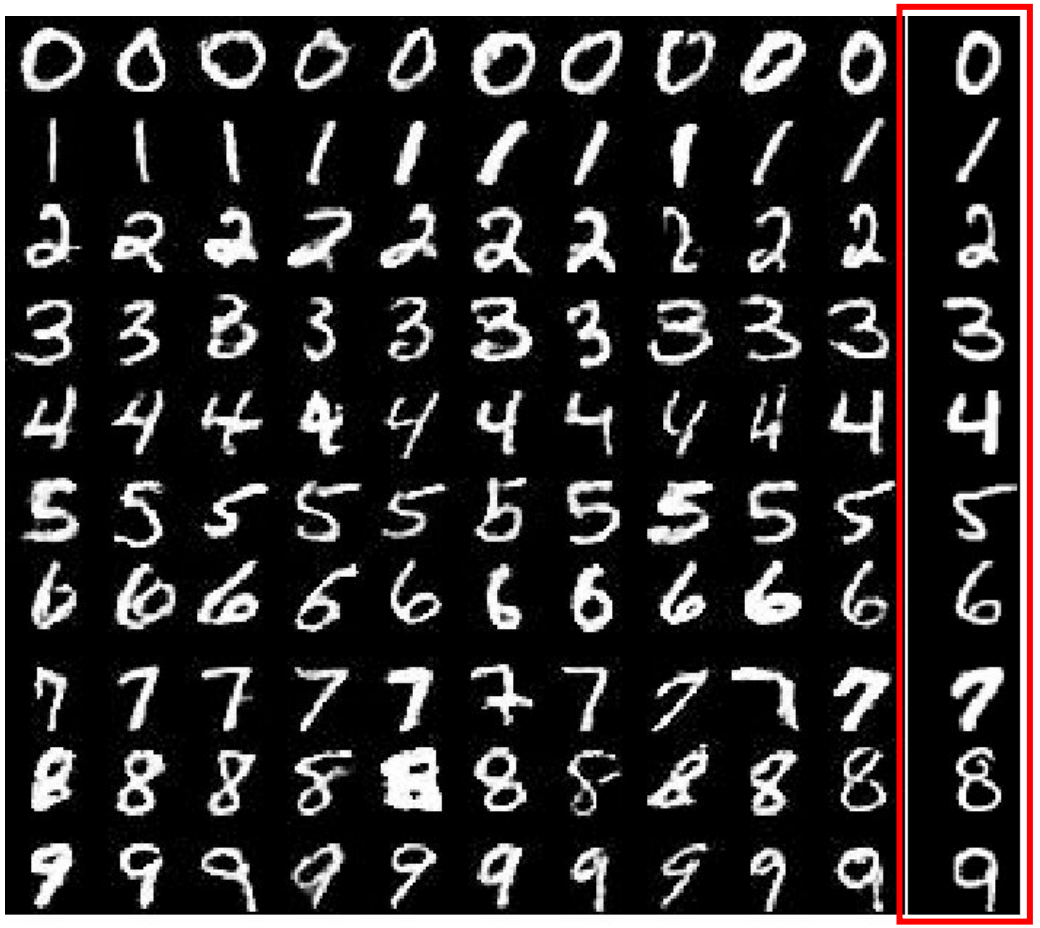
Contents

Vanilla Generative Adversarial Networks



1 Introduction

MNIST



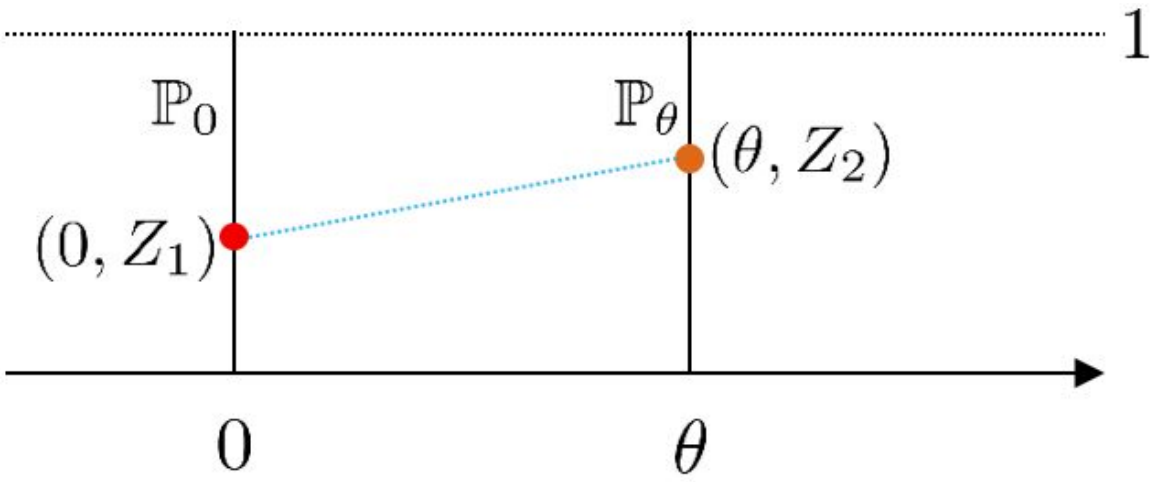
WGAN-GP

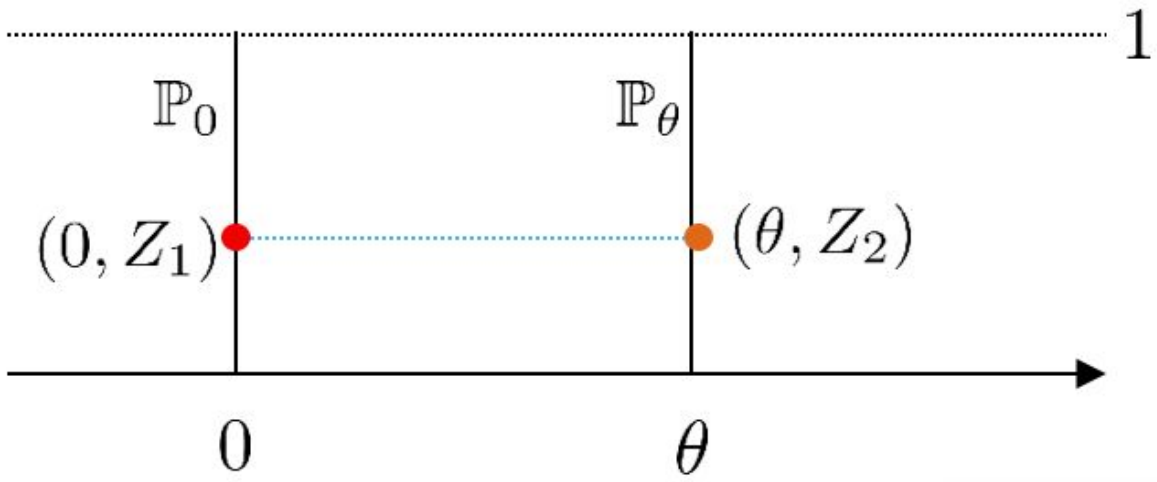
$\prod(\mathbb{P}_r, \mathbb{P}_\theta)$: *joint distribution*

γ : *one of the joint distribution*

Wasserstein-1 distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$





Wassertein $-$ 1 *distance minimum value* : $|\theta|$

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = |\theta|$$

WGAN-GP

Wasserstein-1 distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \epsilon \prod(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

$$Problem: \prod(\mathbb{P}_r, \mathbb{P}_\theta)\ have\ the\ large\ space$$

WGAN-GP

Wasserstein-1 distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

$$\Big\downarrow \; Kantorovich - Rubinstein\ Theorem$$

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$$\|f\|_L \leq 1 : f\ function\ is\ the\ 1 - Lipchitz\ function$$

The k lipschitz function does not increase the distance between two points by more than a certain ratio.

WGAN-GP

Wasserstein-1
distance

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$$\|f\|_L \leq 1 : f\ function\ is\ the\ 1 - Lipchitz\ function$$

$$\max_{w\in W} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$$

WGAN-GP

WGAN

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

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

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

f_w : discriminator
g_θ : generator

WGAN-GP

WGAN

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

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

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $\boxed{w \leftarrow \text{clip}(w, -c, c)}$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\boxed{\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)}$
end while

f_w : discriminator
g_θ : generator

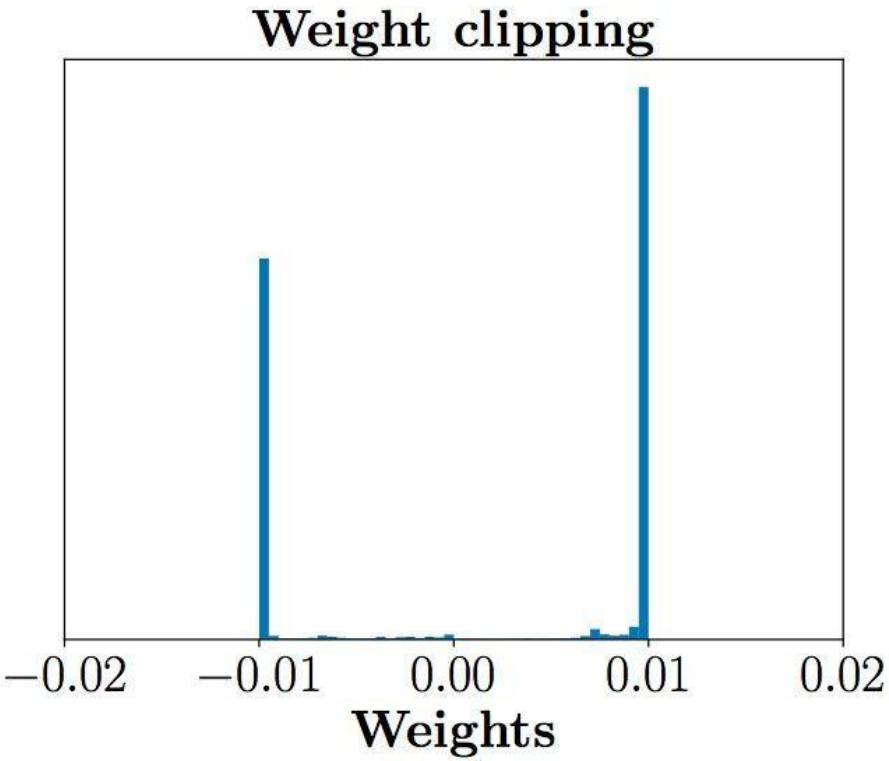
Weight clipping

$$W = [-0.01, 0.01]^d$$

WGAN-GP

WGAN

$$\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$



WGAN-GP

WGAN-GP (Wasserstein GAN with gradient penalty)

$$L = \underbrace{\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)]}_{\text{Original critic}} + \underbrace{\lambda \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Gradient}}$$

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

```
while θ has not converged do
    for t = 1, ..., n_critic do
        for i = 1, ..., m do
            Sample real data x ~ P_r, latent variable z ~ p(z), a random number ε ~ U[0, 1].
            x̃ ← G_θ(z)
            x̂ ← εx + (1 − ε)x̃
            L^(i) ← D_w(x̃) − D_w(x) + λ(‖∇_x̂ D_w(x̂)‖_2 − 1)^2
        end for
        w ← Adam(∇_w (1/m) Σ_{i=1}^m L^(i), w, α, β_1, β_2)
    end for
    Sample a batch of latent variables {z^(i)}_{i=1}^m ~ p(z).
    θ ← Adam(∇_θ (1/m) Σ_{i=1}^m −D_w(G_θ(z)), θ, α, β_1, β_2)
end while
```

$\mathbb{P}_g$: *generator distribution*
$\mathbb{P}_r$: *discriminator distribution*
$\mathbb{P}_{\hat{x}}$: *sampling along straight lines*

WGAN-GP

WGAN-GP (Wasserstein GAN with gradient penalty)

$$L = \underset{\tilde{x}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{x})] - \underset{x\sim\mathbb{P}_r}{\mathbb{E}}[D(x)] + \lambda \underset{\hat{x}\sim\mathbb{P}_{\hat{x}}}{\mathbb{E}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Original critic loss

Gradient penalty

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \boxed{\lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2}$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

Gradient penalty

$\mathbb{P}_g$: *generator distribution*
$\mathbb{P}_r$: *discriminator distribution*
$\mathbb{P}_{\hat{x}}$: *sampling along straight lines*

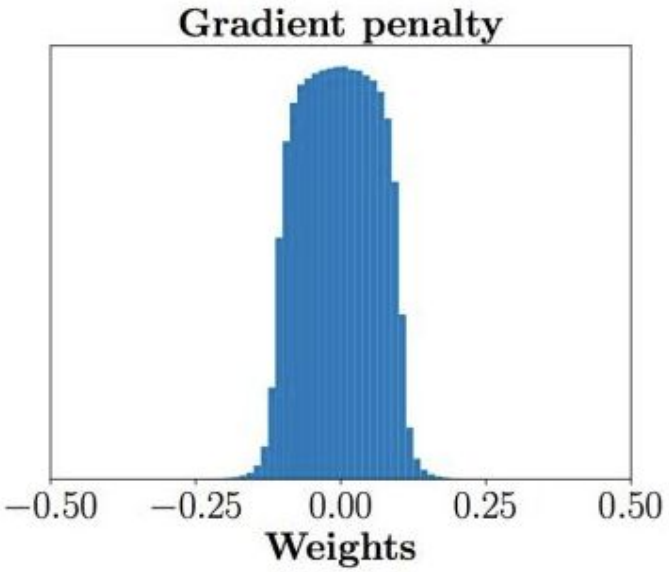
WGAN-GP

WGAN-GP (Wasserstein GAN with gradient penalty)

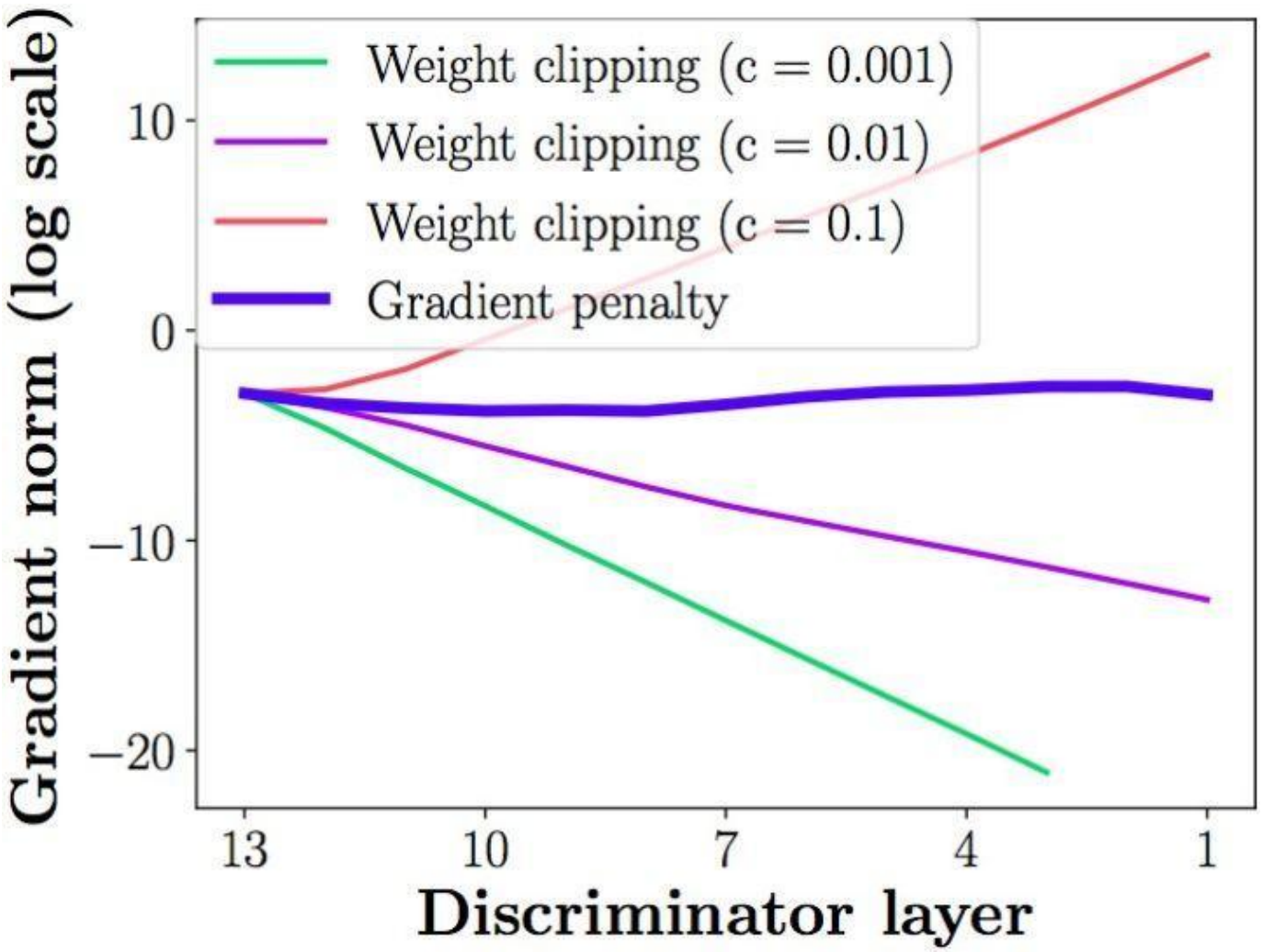
$$L = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Original critic loss

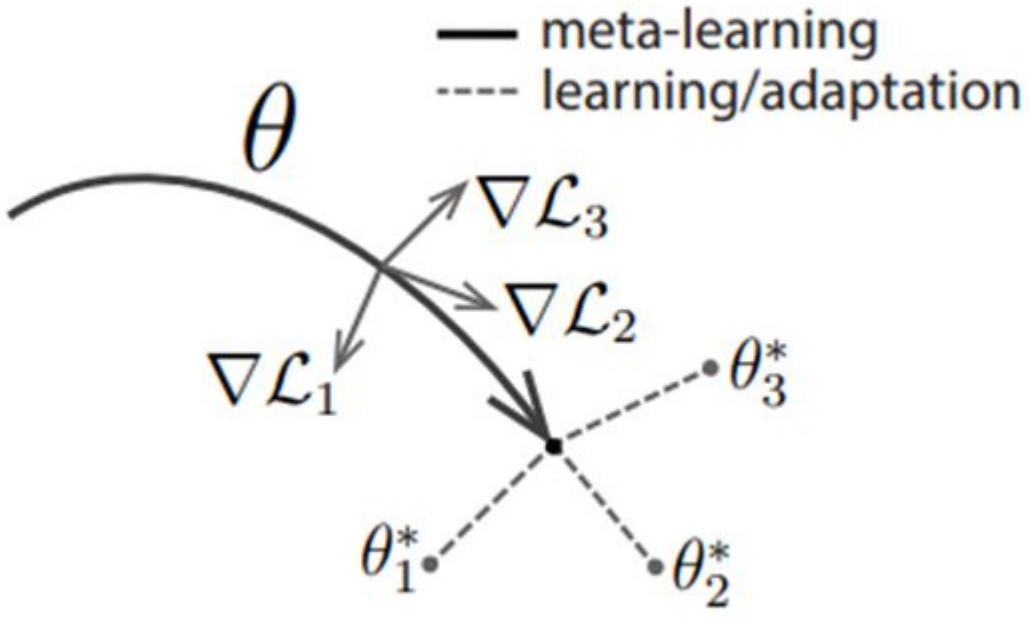
Gradient penalty



WGAN-GP


Weight clipping (c = 0.001)
Weight clipping (c = 0.01)
Weight clipping (c = 0.1)
Gradient penalty
Gradient norm (log scale)
10
0
−10
−20
13
10
7
4
1
Discriminator layer

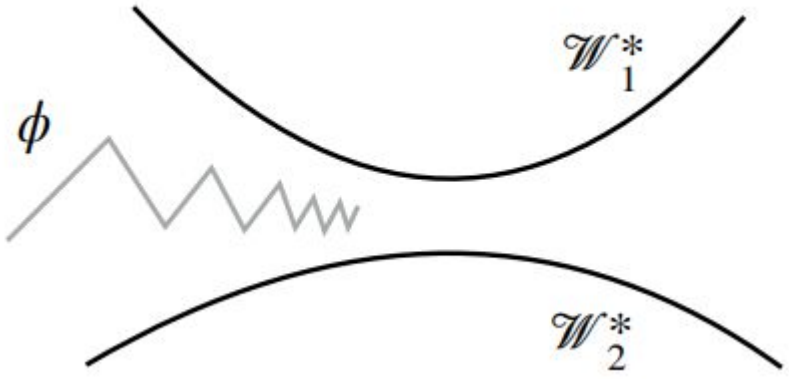
3 Reptile



Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
end while

3 Reptile



Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

3 Reptile

$$\underset{\emptyset}{minimize}\ \mathbb{E}_T[L_T\left(U_T^K(\emptyset)\right)]$$

$$\underset{\emptyset}{minimize}\ \mathbb{E}_T[L_{T,B}\left(U_{T,A}(\emptyset)\right)]$$

$\emptyset$: *initialized parameters to find*
T : *Ramdomly sampled task*
L_T : *loss of T after k update*
U_T^K : *The operator than updates* $\emptyset$ *k time using data sampled from T*
A : *Training samples (inner loop)*
B : *Test samples (outer loop)*

FIGR

Algorithm 1: FIGR training

```
Initialize Φ_d, the discriminator parameter vector
Initialize Φ_g, the generator parameter vector
for iteration 1, 2, 3 ... do
    Make a copy of Φ_d resulting in W_d
    Make a copy of Φ_g resulting in W_g
    Sample task τ
    Sample n images from X_τ resulting x_τ
    for K > 1 iterations do
        Generate latent vector z
        Generate fake images y with z and W_g
        Perform step of SGD update on W_d with
         Wasserstein GP loss and x_τ and y
        Generate latent vector z
        Perform step of SGD update on W_g with
         Wasserstein loss and z
    end for
    Set Φ_d gradient to be Φ_d - W_d
    Perform step of Adam update on Φ_d
    Set Φ_g gradient to be Φ_g - W_g
    Perform step of Adam update on Φ_g
end for
```

Φ_d : *discriminator parameter vector (meta − t*

Φ_g : *generator parameter vector (meta − train*

W_d : *discriminator parameter vector*

W_g : *generator parameter vector*

K : *inner loop step size*

FIGR

Algorithm 1: FIGR training

Initialize Φ_d, the discriminator parameter vector
Initialize Φ_g, the generator parameter vector
for iteration 1, 2, 3 ... **do**
  Make a copy of Φ_d resulting in W_d
  Make a copy of Φ_g resulting in W_g
  Sample task τ
  Sample n images from X_τ resulting x_τ
  for $K > 1$ iterations **do**
    Generate latent vector z
    Generate fake images y with z and W_g
    Perform step of SGD update on W_d with
     Wasserstein GP loss and x_τ and y
    Generate latent vector z
    Perform step of SGD update on W_g with
     Wasserstein loss and z
  end for
  Set Φ_d gradient to be Φ_d - W_d
  Perform step of Adam update on Φ_d
  Set Φ_g gradient to be Φ_g - W_g
  Perform step of Adam update on Φ_g
end for

Φ_d : *discriminator parameter vector (meta − t*
Φ_g : *generator parameter vector (meta − train*
W_d : *discriminator parameter vector*
W_g : *generator parameter vector*
K : *inner loop step size*

FIGR

Algorithm 1: FIGR training

Initialize Φ_d, the discriminator parameter vector
Initialize Φ_g, the generator parameter vector
for iteration 1, 2, 3 ... **do**
  Make a copy of Φ_d resulting in W_d
  Make a copy of Φ_g resulting in W_g
  Sample task τ
  Sample n images from X_τ resulting x_τ
  for $K > 1$ iterations **do**
    Generate latent vector z
    Generate fake images y with z and W_g
    Perform step of SGD update on W_d with
     Wasserstein GP loss and x_τ and y
    Generate latent vector z
    Perform step of SGD update on W_g with
     Wasserstein loss and z
  end for
  Set Φ_d gradient to be Φ_d - W_d
  Perform step of Adam update on Φ_d
  Set Φ_g gradient to be Φ_g - W_g
  Perform step of Adam update on Φ_g
end for

Φ_d : *discriminator parameter vector (meta − t*
Φ_g : *generator parameter vector (meta − train*
W_d : *discriminator parameter vector*
W_g : *generator parameter vector*
K : *inner loop step size*

FIGR

Algorithm 2: FIGR generation

Using W_d, a copy of the meta-trained Φ_d
Using W_g, a copy of the meta-trained Φ_g
Sample test task τ
Sample n images as x_τ from X_τ
for $K >= 1$ iterations **do**
    Generate latent vector z
    Generate fake images y with z and W_g
    Perform step of SGD update on W_d with
      Wasserstein GP loss and x_τ and y
    Generate latent vector z
    Perform step of SGD update on W_g with
      Wasserstein loss and z
end for
Generate latent vector z
Generate fake images y

Φ_d : *discriminator parameter vector (meta − t*
Φ_g : *generator parameter vector (meta − train*
W_d : *discriminator parameter vector*
W_g : *generator parameter vector*

FIGR

Algorithm 2: FIGR generation

Using W_d, a copy of the meta-trained Φ_d
Using W_g, a copy of the meta-trained Φ_g
Sample test task τ
Sample n images as x_τ from X_τ
for $K >= 1$ iterations **do**
    Generate latent vector z
    Generate fake images y with z and W_g
    Perform step of SGD update on W_d with
      Wasserstein GP loss and x_τ and y
    Generate latent vector z
    Perform step of SGD update on W_g with
      Wasserstein loss and z
end for
Generate latent vector z
Generate fake images y

Φ_d : *discriminator parameter vector (meta − t*
Φ_g : *generator parameter vector (meta − train*
W_d : *discriminator parameter vector*
W_g : *generator parameter vector*

Fast adaptation

FIGR

$$minimize_{\Phi}\mathbb{E}_T[L_T\left(U_T^k(\Phi)\right)]$$

$$minimize\sum_T(\Phi_d - W_{d_T}) + (\Phi_g - W_{g_T})$$

$$\Phi_1 - W_{T1} < \Phi_2 - W_{T2}$$

$U_T^k(\Phi)$ is the operator that updates Φ k times using x_n
A total of n data points sampled from X_T

$T : task$

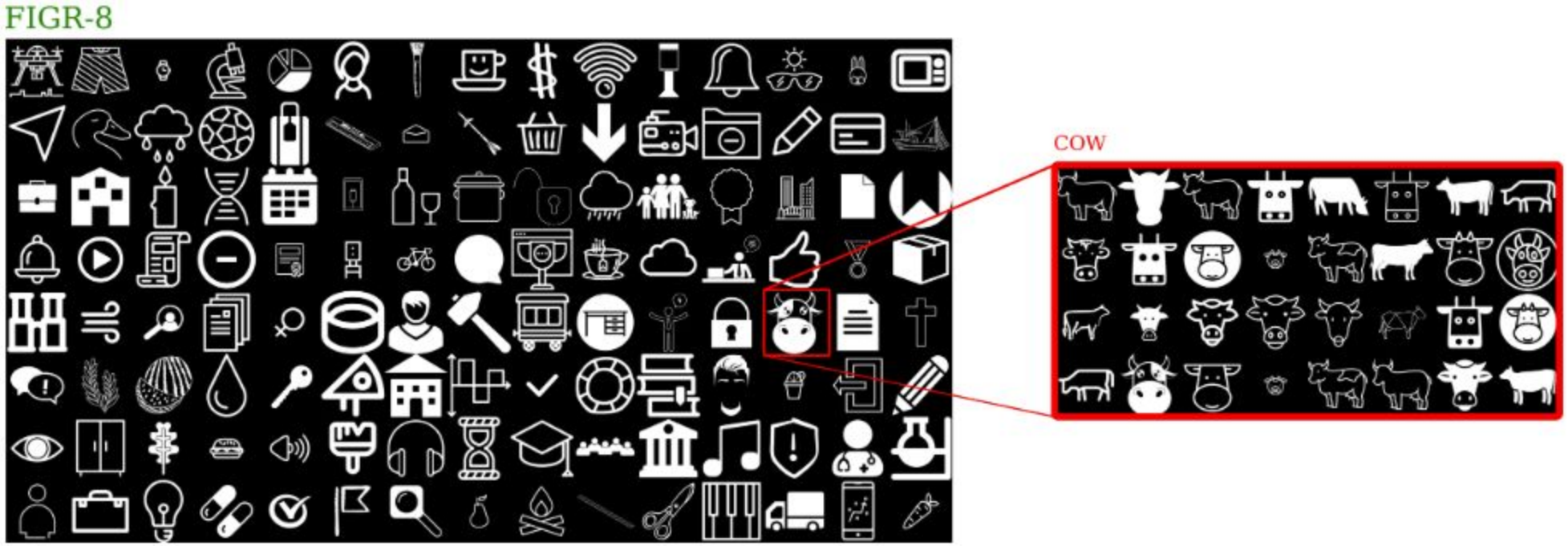
$W_{d_T} : Discriminator\ weight$

$W_{g_T} : Generator\ weight$

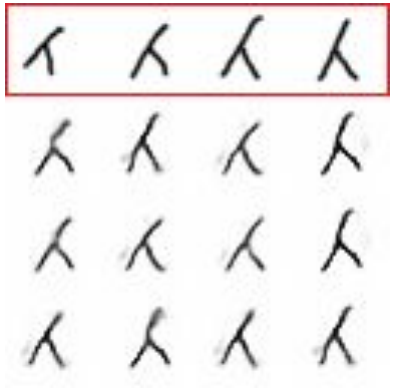
$\Phi_d, \Phi_g :\ Reptile\ initialize\ the\ weights$

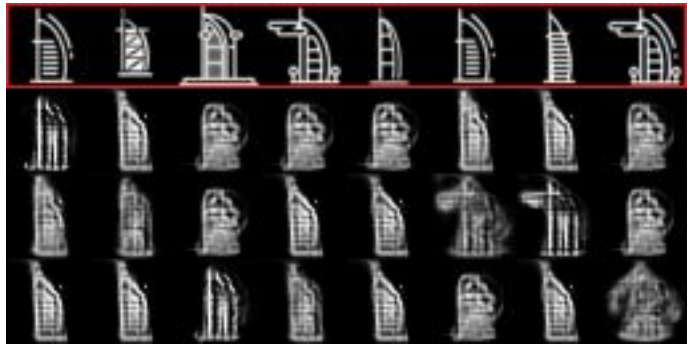
FIGR

17,375 classes of 1,548,256 grayscale images (192x192)



4 FIGR





Omniglot : 1623 classes
Meta-train : randomly sampled 1603
Meta-test : randomly sampled 20

FIGR-8 : 18,409 classes
Meta-train : randomly sampled 18,359
Meta-test : randomly sampled 50

5 Result & Discussion

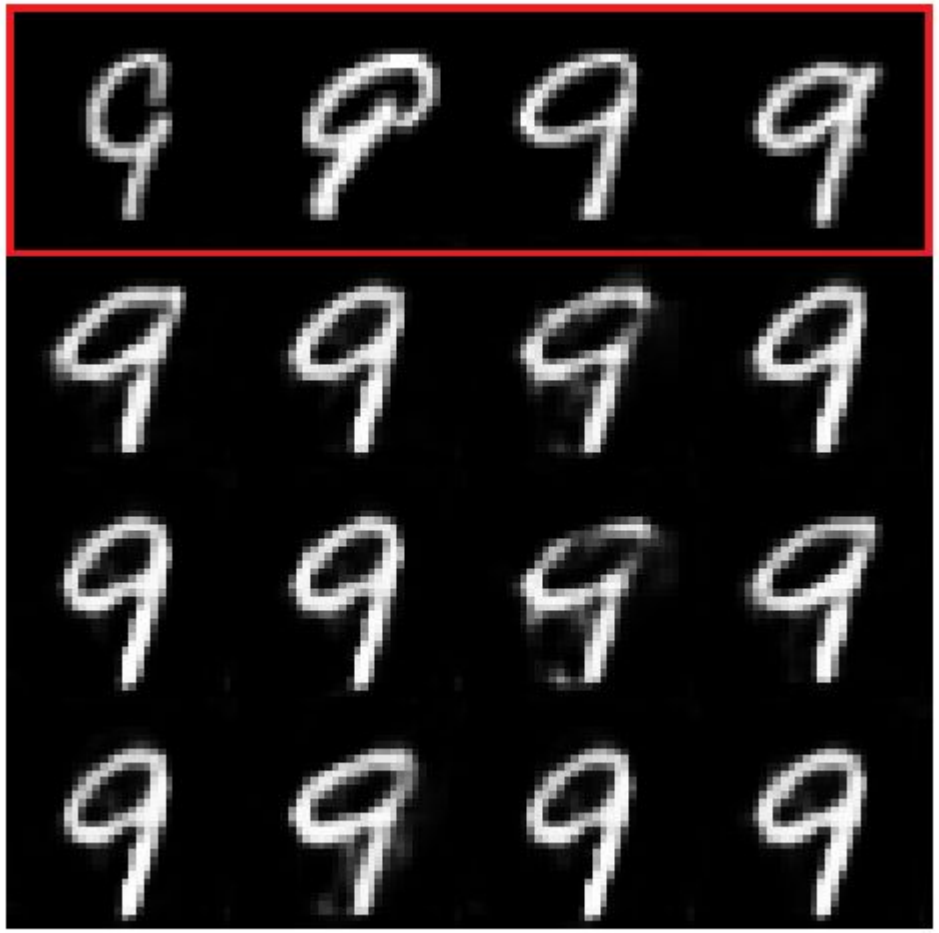


Figure 3: MNIST; 50,000 update; 10 gradient steps

5 Result & Discussion

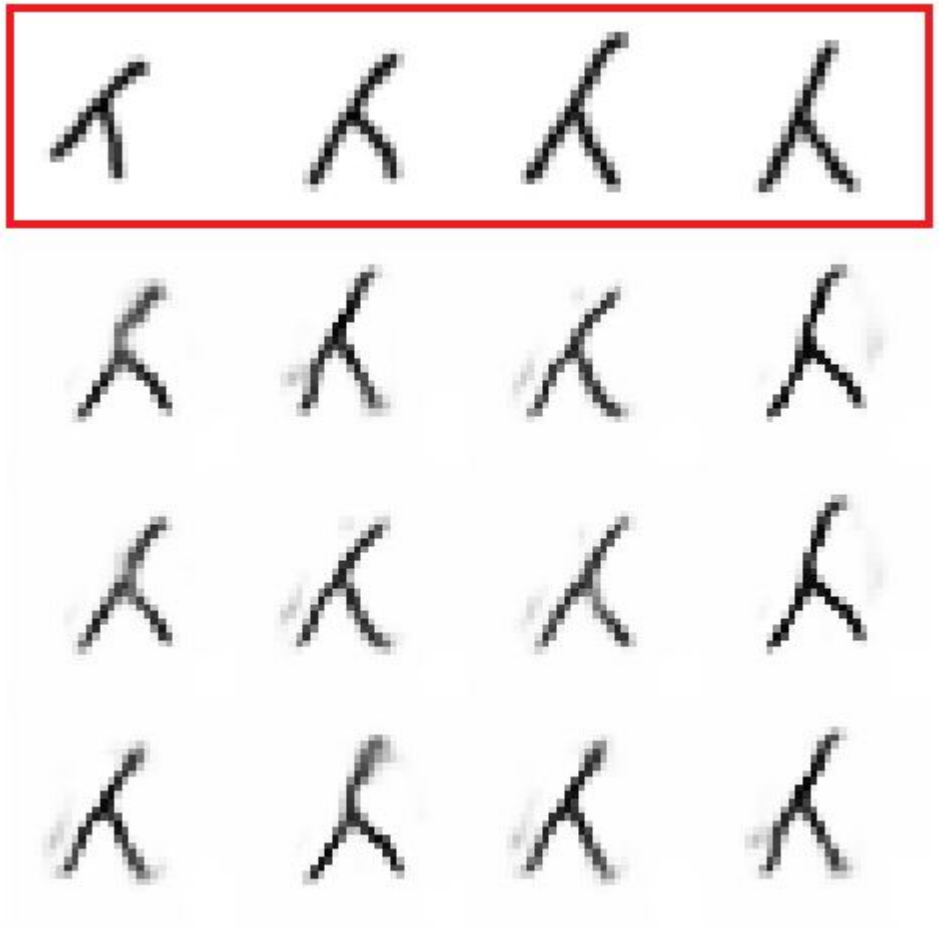


Figure 4: Omniglot; 140,000 update; 10 gradient steps

Result & Discussion

Figure 8: FIGR-8; 90,000 update; 10 gradient steps; $n = 8$

Reference

- https://talkingaboutme.tistory.com/entry/DL-Meta-Learning-Learning-to-Learn-Fast
- https://haawron.tistory.com/21
- https://www.tinymind.cn/articles/193
- https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490
- https://arxiv.org/pdf/1701.07875.pdf
- https://arxiv.org/pdf/1704.00028.pdf
- https://arxiv.org/pdf/1703.03400.pdf
- https://arxiv.org/pdf/1803.02999.pdf
- https://arxiv.org/pdf/1901.02199.pdf