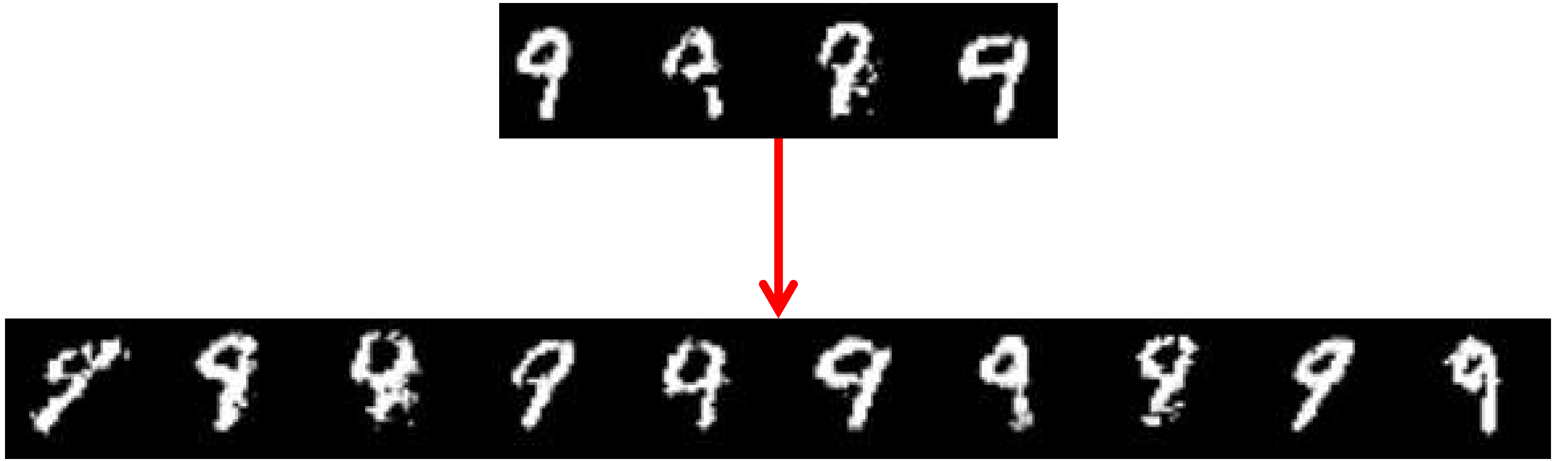
Few-Shot Generative Model

- Goal:

  Given few samples, generating many similar figures



Few-Shot Generative Model

Motivation:

- GANs require several orders of magnitude more data points than humans in order to generate comprehensible images.
- if the data is abundant enough to successfully train a GAN, there is little purpose to generating more of this data.

Few-Shot Generative Model

- Meta Learning (MAML or Reptile)

  learning a parameter initialization that can be fine-tuned quickly on a new task

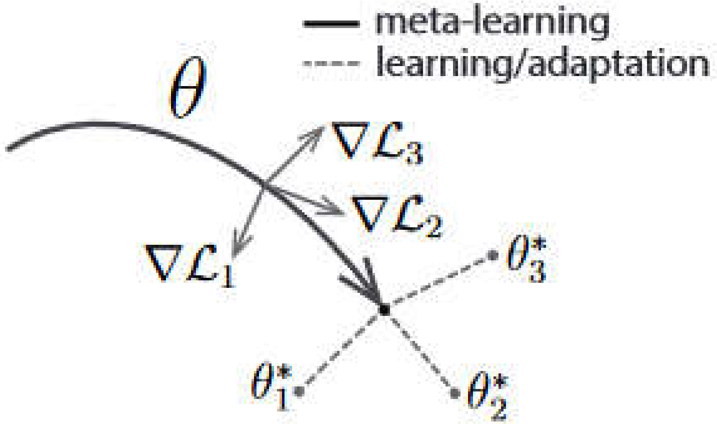


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Few-Shot Generative Model

- One obvious way:
  - Think of GAN as a black box, then put it into meta-learning structure directly.

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

The part of GAN

Few-Shot Generative Model

- Issues: It's difficult to train entire GAN with meta-learning and the result is not good (mode collapse).

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

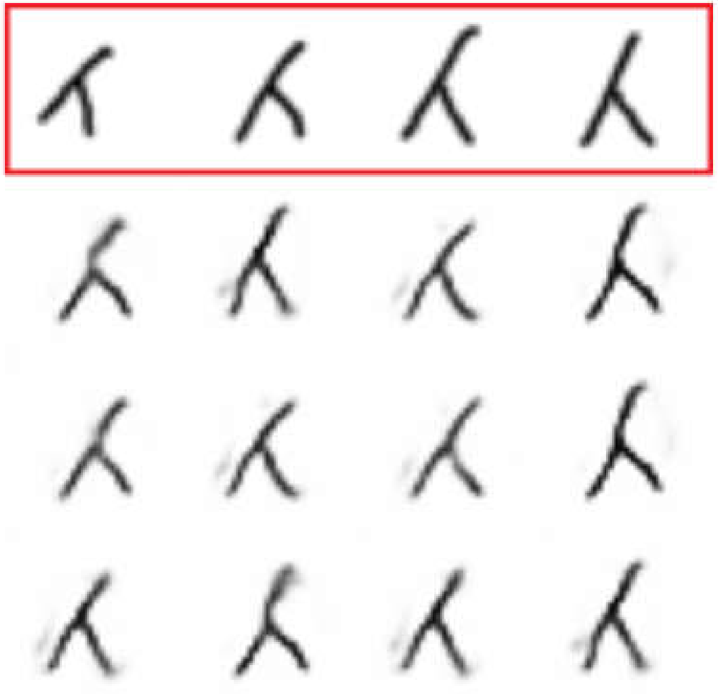


Figure 4: Omniglot; 140,000 update; 10 gradient steps

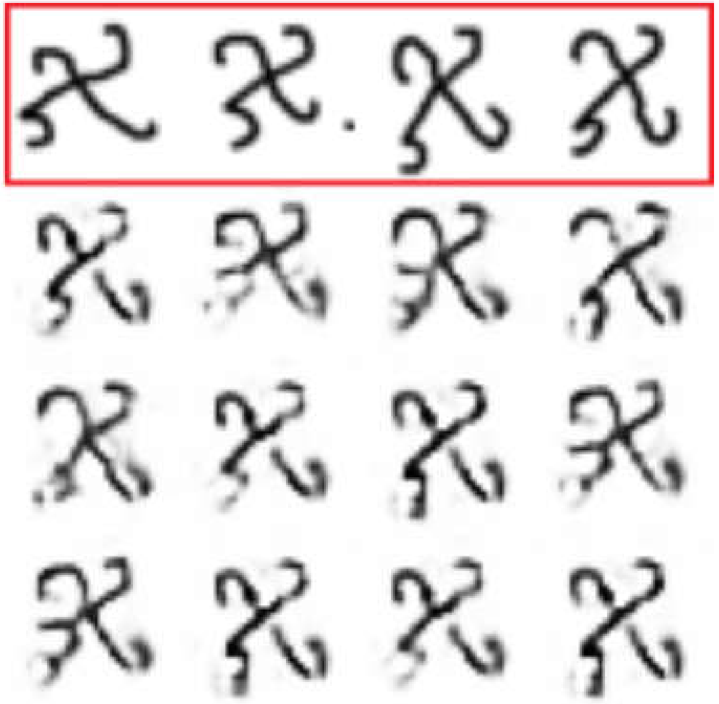


Figure 5: Omniglot; 230,000 update; 10 gradient steps

Few-Shot Generative Model

- our method:

  original GAN: Disriminator D(x),   Generator G(z)

  Add input x (images) and divide G into three parts:

$$h_{\theta_h}(g_{\theta_g}(z), f_{\theta_f}(x_i))$$

  g: the variety of images.
  f: the basic structure of images.
  h: the combination of varity and basic structure.

Algorithm 3 Meta-GAN-beta2.0: training phase

Initialize ϕ_w, the discriminator (D) parameter vector
Initialize θ_h, θ_g, θ_f, the generator (G) parameter vector
for iteration 1,2,3 ... **do**
  Make a copy of ϕ_w resulting in ϕ_W (D_{meta})
  Make a copy of θ_g resulting in θ_G (G_{meta})
  Sample task T and P
  Sample n images resulting x_T, x_P
  for k < 10 (inner loop) **do**
    """"""the part of Discriminator (D_{meta})""""""
    Generate latent vector z
    Generate fake images y with z and θ_h, θ_G, θ_f, ($y = h_{\theta_h}(g_{\theta_G}(z), f_{\theta_f}(x_T))$)
    Generate fake images y_ℓ with z and θ_h, θ_G, θ_f, ($y_\ell = h_{\theta_h}(0, f_{\theta_f}(x_T))$)
    Combine real and fake images $x = torch.cat(x_T, y, y_\ell)$
    Compute real probability $\phi_W(x)$ and intermediate layer ℓ_d of real images x_T
    Perform step of SGD update on ϕ_W with Wasserstein dual loss of x_T, y, y_ℓ and gradient penalty
    """"""the part of Generator (G_{meta})""""""
    Generate latent vector z
    Generate fake images $x = h_{\theta_h}(g_{\theta_G}(z), f_{\theta_f}(x_P))$
    Compute the Wasserstein loss L_1 between $concat(g_{\theta_G}(z), f_{\theta_f}(x_P))$ and ℓ_d
    Compute the Wasserstein dual loss (i.e. Expectation) $L_3 = h_{\theta_h}(g_{\theta_G}(z), f_{\theta_f}(x_P))$
    Perform step of SGD update on $\theta = [\theta_G]$ with $L_1 + \lambda_2 L_3$
  end for
  Set ϕ_w gradient to be $\phi_w - \phi_W$
  Perform step of Adam update on ϕ_w (D)
  Set θ_g gradient to be $\theta_g - \theta_G$
  Perform step of Adam update on θ_g
  Generate latent vector z
  Get intermediate layer ℓ_d from $\phi_w(x_T)$
  Compute the Wasserstein loss L_1 between $concat(g_{\theta_g}(z), f_{\theta_f}(x_P))$ and ℓ_d
  Compute the Wasserstein dual loss (i.e. Expectation) $L_2 = h_{\theta_h}(0, f_{\theta_f}(x_P))$
  Compute the Wasserstein dual loss (i.e. Expectation) $L_3 = h_{\theta_h}(g_{\theta_g}(z), f_{\theta_f}(x_P))$
  Perform step of Adam update on θ_h, θ_f with $L_1 + \lambda_1 L_2 + \lambda_2 L_3$
end for