

# Wasserstein GANs

# Wasserstein GANs

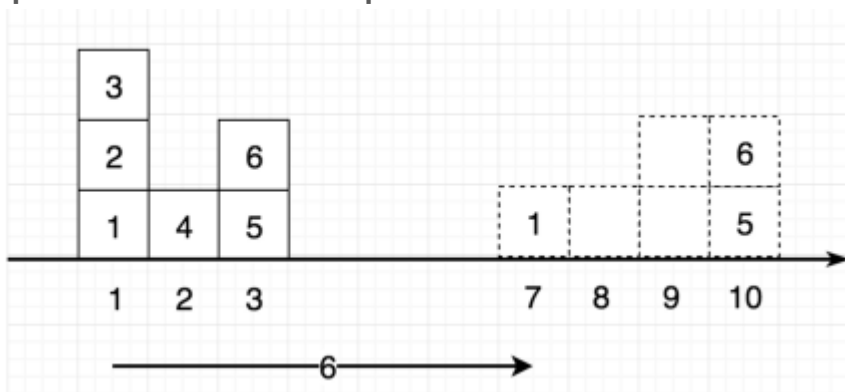
- Motivadas por la complejidad del entrenamiento tradicional de las GANs (juego de suma cero con equilibrio inestable).
- La función de pérdida penaliza el modelo proporcionalmente a la distancia entre la distribución de probabilidad predicha y la distribución de probabilidad esperada para un dato determinado.
- Las WGAN proponen un modelo alternativo para el entrenamiento del generador, intentando lograr una mejor aproximación de la distribución de datos observada en cada conjunto de entrenamiento.

# Wasserstein GANs

- En lugar de utilizar un discriminador para clasificar o predecir la probabilidad de que los datos generados sean reales o falsos, una WGAN reemplaza el discriminador con un **crítico**, una ANN que puntúa la realidad (o falsedad) de un dato determinado.
- Modifica la métrica de distancia original de las GANs (divergencia de Jensen-Shannon) por la **distancia de Wasserstein**.
- La diferencia fundamental de la distancia de Wasserstein es su impacto en la convergencia de las series de distribuciones de probabilidad de los datos generados.

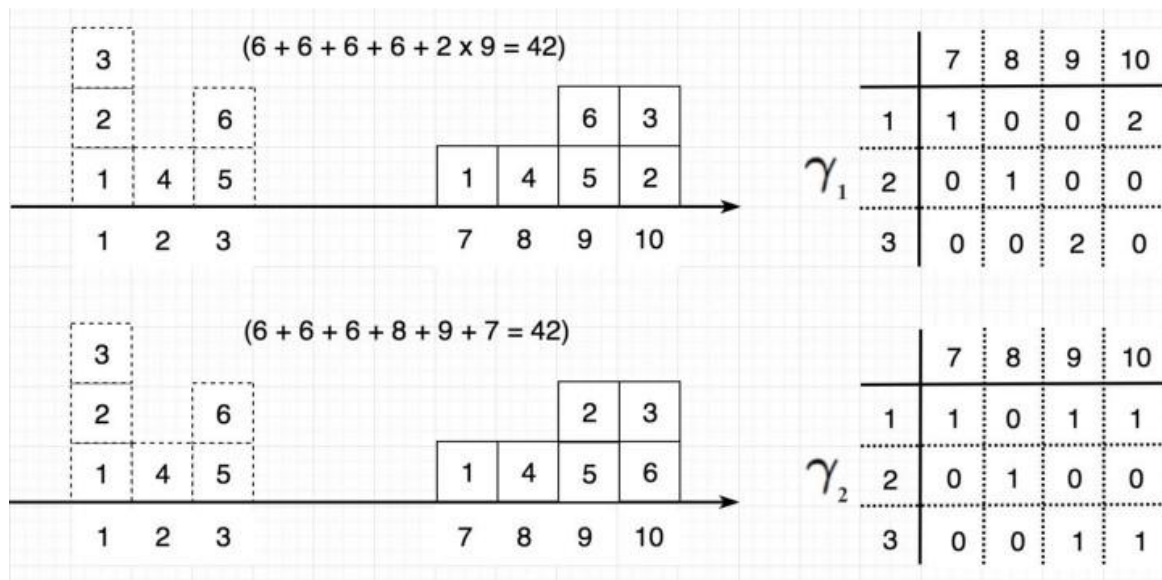
# Wasserstein GANs

- Idea: utilizar una métrica específica (distancia de Wasserstein) para evaluar las diferencias entre distribuciones de probabilidad.
- La métrica también se conoce como “la distancia del movimiento de tierra” (Earth mover's distance) o distancia de Kantorovich-Rubinstein.
- Mínimo costo de transformar una pila en otra: el costo es la cantidad (de tierra) movida por la distancia que se mueve.



# Earth mover's distance

- Hay muchos planes posibles para la transformación.
- No todos los planes tienen el mismo costo.
- La distancia de Wasserstein es el mínimo costo de todos los planes



# Earth mover's distance

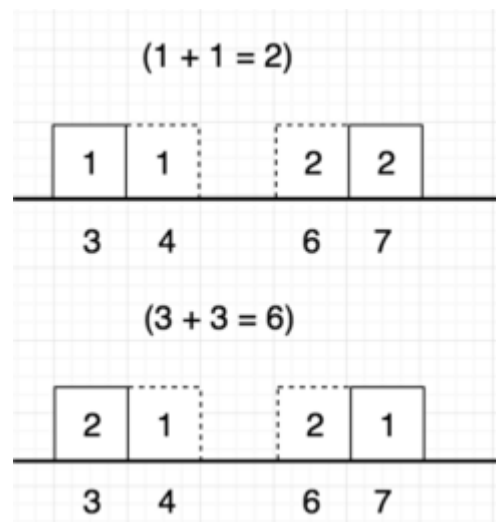
- No todos los planes tienen el mismo costo.
- La distancia de Wasserstein es el mínimo costo de todos los planes

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

- En las WGAN se utiliza para evaluar la distancia entre la distribución y la generada

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

- $f$  debe ser Lipschitziana
- $\mathbb{P}_\theta$  es una densidad paramétrica



# Distancia de Wasserstein

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

- $\mathbb{P}_\theta$  es una densidad paramétrica
- En lugar de estimar la densidad de  $\mathbb{P}_r$  se trabaja con una variable aleatoria  $Z$  con distribución  $p(z)$  que se pasa a una función paramétrica  $g_\theta: Z \rightarrow X$  que genera muestras de la distribución  $\mathbb{P}_\theta$
- La idea es variar el parámetro  $\theta$  para intentar aproximarse lo más posible a  $\mathbb{P}_r$
- Típicamente  $g_\theta$  es una ANN.

# Distancia de Wasserstein

- Es una función continua y casi diferenciable (en todos los puntos del dominio), que permite realizar entrenamientos más cercanos al óptimo.
- Es una métrica relevante: converge a 0 a medida que las distribuciones se acercan entre sí y diverge a medida que se alejan.
- Es más estable como función objetivo que la divergencia tradicional (Jensen-Shannon), que se satura localmente a medida que mejora el discriminador, causando la desaparición de gradientes.
- También ayuda a mitigar problemas de colapso de modo.



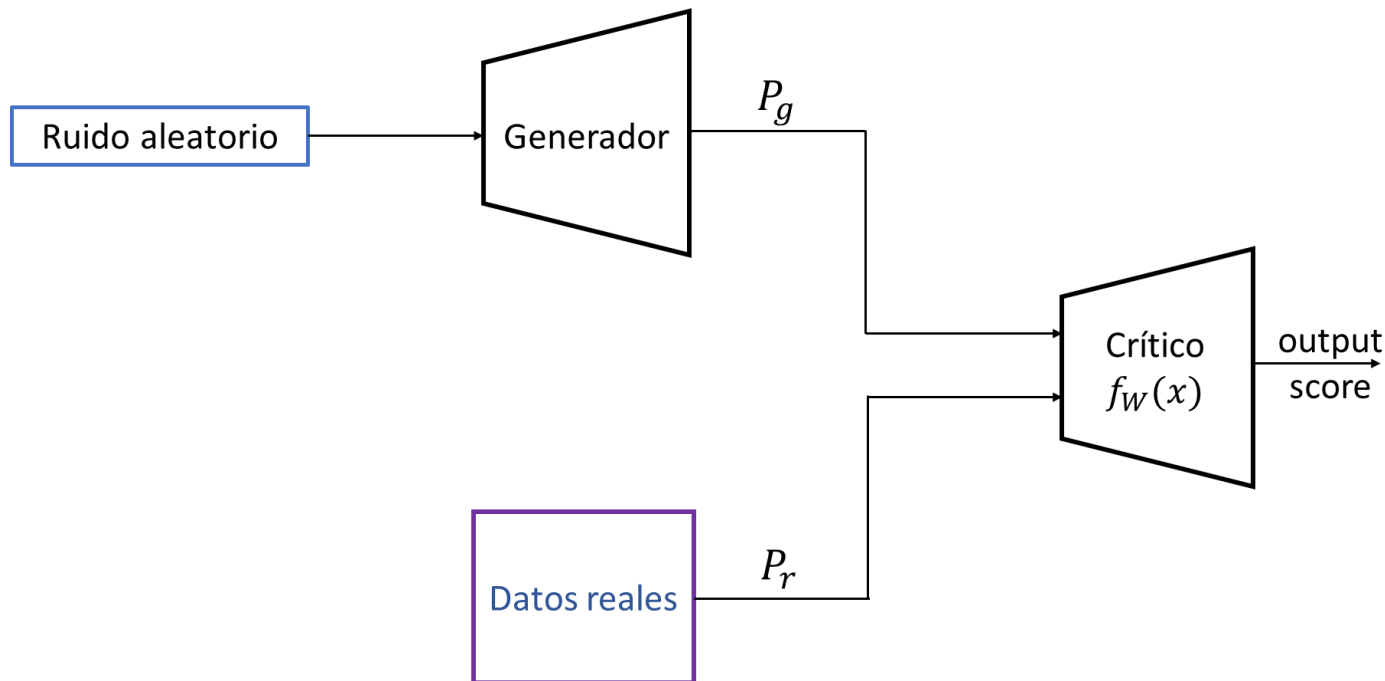
# Wasserstein GANs

- **Crítico** (critical) en lugar de discriminador
- El crítico trata de determinar una función de ajuste (fitting)  $f_w(x)$  que permite aproximar la distribución de los datos reales (valor esperado del ‘output score’)
  - $f_w(x)$  se aplica a batches de datos reales, para calcular el ‘output score’ de cada muestra. Con los valores obtenidos se estima la distribución de datos reales
  - El mismo procedimiento se aplica para los datos sintéticos generados
- El objetivo es minimizar la distancia de Wasserstein entre las distribuciones empíricas calculadas

$$L(P_r, P_g) = \mathbb{E}_{x \in P_r} [f_w(x)] - \mathbb{E}_{z \in P_g(z)} [f_w(G_\theta(z))]$$

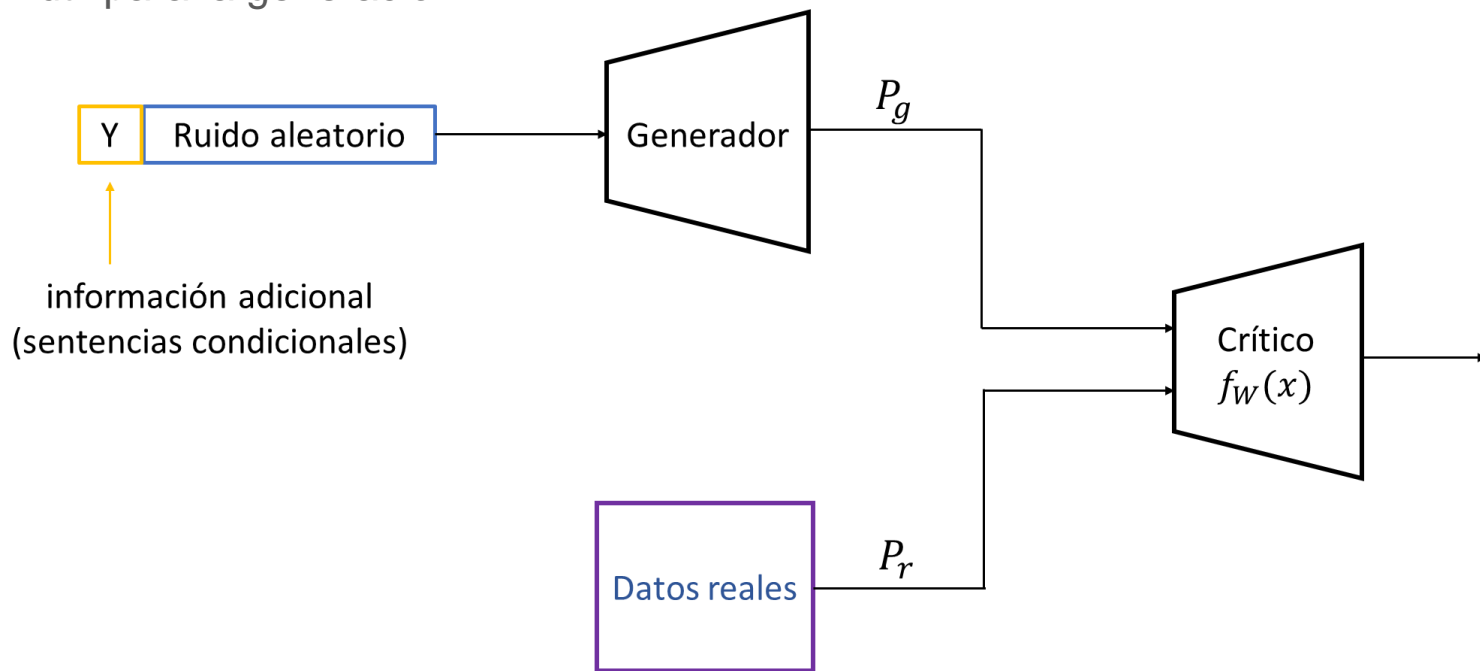
# Wasserstein GANs

- Si  $W(P_g - P_r) < \varepsilon$ , las distribuciones pueden considerarse iguales y los datos generados seguirán la misma distribución que los reales.

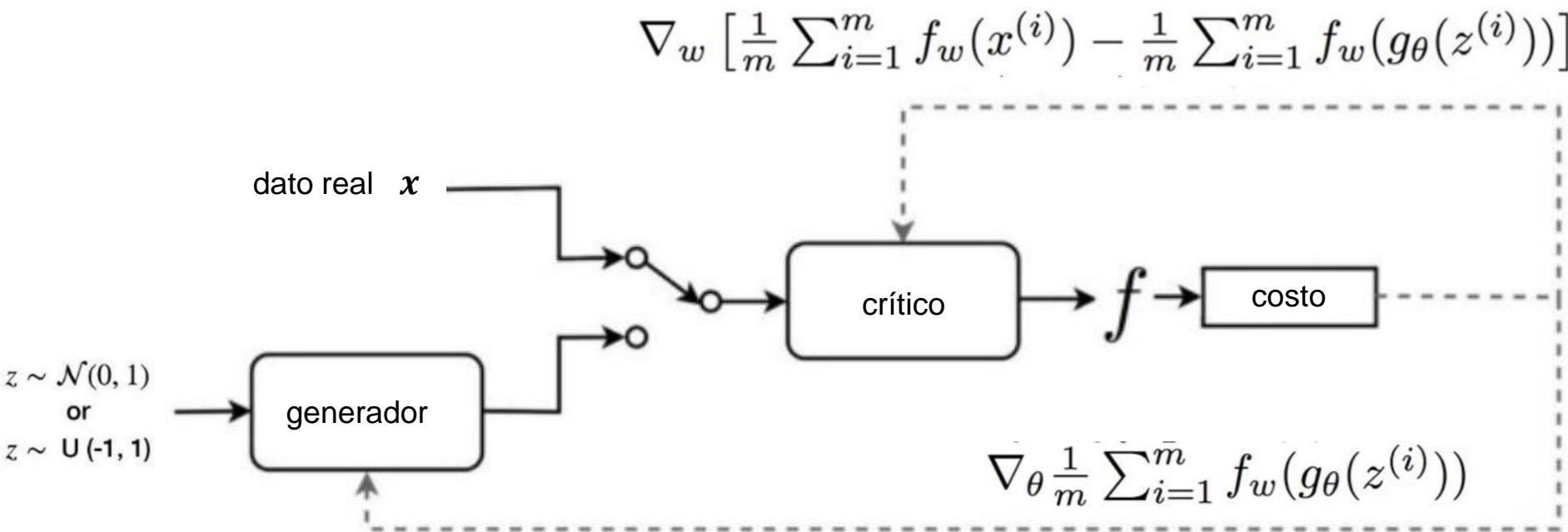


# Wasserstein GANs

- El modelo admite incluir condiciones, para trabajar con GANs condicionales
  - Por ejemplo, puntos con determinadas propiedades que proporcionen información útil para la generación



# Esquema general del entrenamiento de una Wasserstein GAN



# WGAN: entrenamiento del crítico

```
for i in n_critic_steps:
    opt_critic.zero_grad()
    real_images = data[0].float().to(device)

    # Generar imágenes
    noise = sample_noise()
    fake_images = netG(noise)

    # Evaluarlas con el crítico
    real_log = netCritic(real_images)
    fake_log = netCritic(fake_images)

    # Calcular  $L = E_{\{x \sim P_X\}}[f_W(x)] - E_{\{Z \sim P_Z\}}[f(G(z))]$ 
    loss = -(real_log.mean() - fake_log.mean())

    loss.backward(retain_graph=True)
    opt_critic.step()

    # Clipping del gradiente
    for p in netCritic.parameters():
        p.data.clamp_(-self.c, self.c)
```

# WGAN: entrenamiento del generador

- El objetivo del generador es minimizar la distancia de Wasserstein entre  $P_g$  y  $P_r$ .
- El generador intenta encontrar  $\theta^*$  que minimice la distancia de Wasserstein entre  $P_g$  y  $P_r$

$$L_{gen}(w) = \min_{\theta} \mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{z \sim Z} [f_w(g_{\theta}(z))] = \min_{\theta} -\mathbb{E}_{z \sim Z} [f_w(g_{\theta}(z))]$$

- La principal diferencia entre el generador de una WGAN y el generador estándar es que el generador WGAN intenta minimizar la distancia de Wasserstein entre  $P_g$  y  $P_r$ , mientras que el generador estándar intenta engañar al discriminador con las imágenes generadas.

$$- \mathbb{E}_{z \in P_y(z)} [f_w(G_{\theta}(z))]$$

# WGAN: entrenamiento del generador

```
opt_gen.zero_grad()

noise = sample_noise()

fake_images = netG(noise)

# Evaluar las imágenes generadas
fake_log = netCritic(fake_images)

# * -  $E_{Z \sim P_Z}[f_W(g(z))]$ 
loss = -fake_log.mean().view(-1)

loss.backward()

opt_gen.step()
```

# Wasserstein GAN: algoritmo

---

**Require:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.

$n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

---



# WGAN-GP: penalización de gradientes en Wasserstein GANs

- Para asegurar continuidad de Lipschitz de la función de pérdida, se limitan (podan) los pesos a un rango  $[-c, c]$ 
  - La poda puede generar fallos de gradiente en el entrenamiento y limita la clase de funciones  $f_w$  que se pueden aprender.
- En una WGAN-GP se suma una penalización del gradiente a la función de pérdida, para asegurar que entrene apropiadamente en un espacio continuo

$$L(P_r, P_g) = \mathbb{E}_{x \in P_r} [f_w(x)] - \mathbb{E}_{z \in P_g} [f_w(G_\theta(z))] + k \cdot \mathbb{E}_{x \in P_g} [(\|\nabla f_w(x)\|_2 - 1)^2]$$

- $P_g$  es la distribución obtenida al muestrear uniformemente a lo largo de líneas rectas entre puntos de las distribuciones real ( $P_r$ ) y generada ( $P_g$ )

# WGAN-GP: penalización de gradientes en Wasserstein GANs

- WGAN-GP converge más rápido y puede generar muestras de mayor calidad que al aplicar la estrategia de poda de pesos
- En este caso Se evita la normalización de batches para el crítico, porque crea correlaciones entre las muestras y reduce la eficacia de la penalización del gradiente.

# Wasserstein GAN-GP: algoritmo

**Require:** The gradient penalty coefficient  $\lambda$ , the number of critic iterations per generator iteration  $n_{\text{critic}}$ , the batch size  $m$ , Adam hyperparameters  $\alpha, \beta_1, \beta_2$ .

**Require:** initial critic parameters  $w_0$ , initial generator parameters  $\theta_0$ .

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_{\theta}(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:  end for
11:  Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:   $\theta \leftarrow \text{Adam}(\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m -D_w(G_{\theta}(\mathbf{z})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

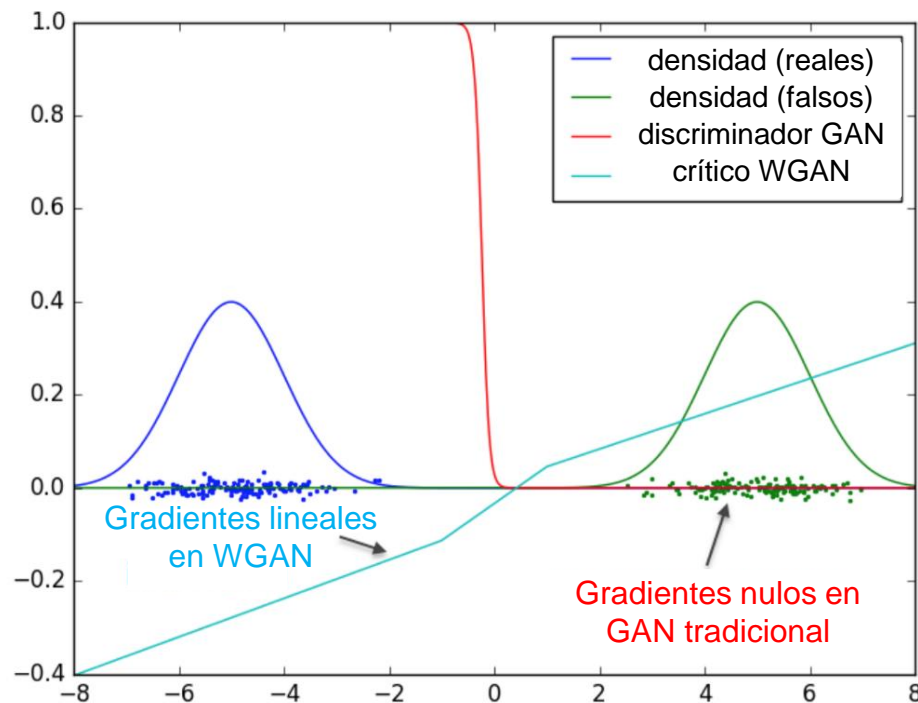
---

# Wasserstein GANs: ventajas

- El crítico es capaz de aportar información valiosa para el entrenamiento del generador, aún cuando esté muy entrenado (idealmente, hasta el óptimo).
  - Es una ventaja fundamental sobre las GAN tradicionales, en las cuales el discriminador, una vez optimizado, no es capaz de proporcionar información útil (del gradiente) para el entrenamiento del generador.
- El crítico no se satura: converge a una función lineal con gradientes definidos en todos los puntos.
- El entrenamiento es más estable, menos sensible a la configuración de la arquitectura y de los hiperparámetros.
- El entrenamiento no requiere mantener el balance entre discriminador y generador.

# Wasserstein GANs: ventajas

- Utilizar la distancia de Wasserstein permite obtener mejores resultados, incluso si las distribuciones  $P_r$  y  $P_g$  son muy diferentes.
- El proceso de aprendizaje es más estable, evitando el mode collapse y mejorando el aprendizaje de clases.
- Una WGAN puede seguir aprendiendo aún cuando el crítico no lo haga.



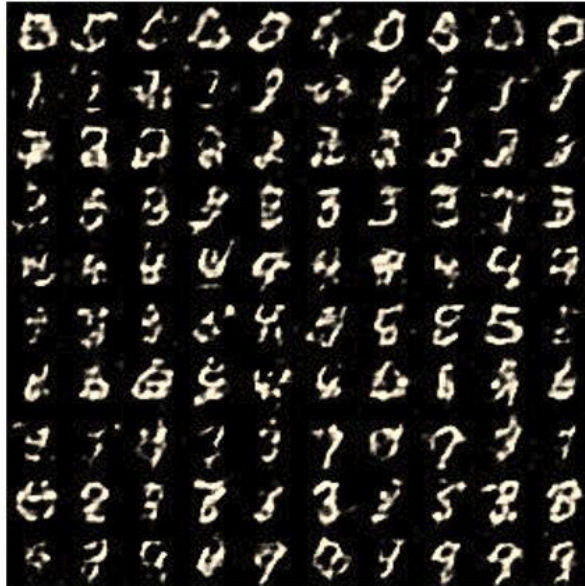
# Wasserstein GANs: implementación

Cambios menores sobre una implementación de GAN tradicional.

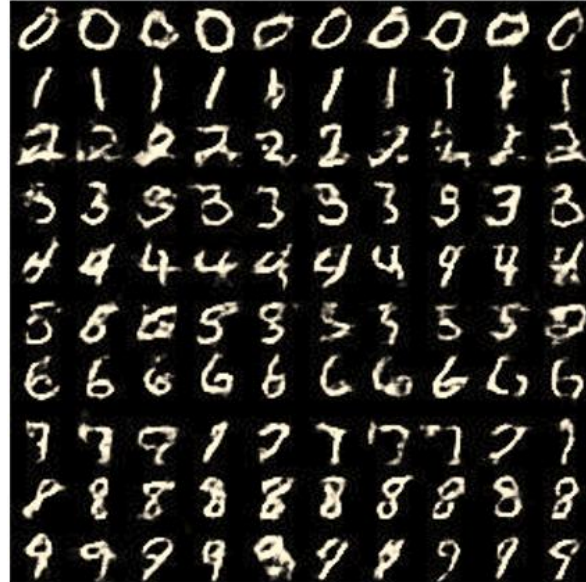
1. Se debe usar la distancia de Wasserstein como pérdida para entrenar el crítico y el generador.
2. Se debe usar una función de activación lineal en la capa de salida del crítico (en lugar de sigmoide en el discriminador).
3. Se debe aplicar algún criterio para asegurar que  $f$  es Liptchitziana: restringir pesos luego de actualizar cada batch:
  - clamp, por ejemplo al rango  $[-0.01, 0.01]$ .
  - Penalización de gradientes, en la función de pérdida
4. Actualizar el crítico más veces que el generador (e.g. 5 veces).
5. Usar la variante RMSProp de SGD con learning rate baja y sin momento.

# Wasserstein GAN: ejemplo MNIST

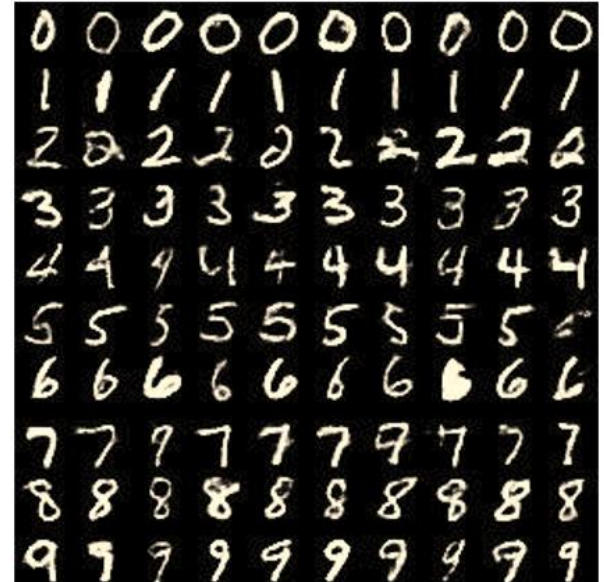
6 épocas de entrenamiento



20 épocas de entrenamiento



40 épocas de entrenamiento





# Wasserstein GAN: ejemplos vs. GAN

MLP, cuatro capas: WGAN mejor diversidad, GAN colapsa a pocos modos



DCGAN, cuatro capas: WGAN imágenes de mejor calidad



GAN sin normalización de batches y número fijo de filtros en cada capa: GAN falla





# Wasserstein GAN: códigos de ejemplo

Disponibles en <https://github.com/nesmachnow/Curso-GANs> (Clase 5)

- Wasserstein GAN  
[colab.research.google.com/drive/1ilRzKzb5WbzHE\\_Da2z1Nzd-bAPt9d9R2](https://colab.research.google.com/drive/1ilRzKzb5WbzHE_Da2z1Nzd-bAPt9d9R2)
- Wasserstein GAN con penalización de gradientes  
[colab.research.google.com/drive/1Lrk6urqFjX3YEnLkCwipx7\\_9gz7QBABo](https://colab.research.google.com/drive/1Lrk6urqFjX3YEnLkCwipx7_9gz7QBABo)
- Wasserstein GAN para modelar una serie temporal financiera  
[github.com/CasperHogenboom/WGAN\\_financial\\_time-series](https://github.com/CasperHogenboom/WGAN_financial_time-series)